DEPARTMENT OF COMPUTER SCIENCE

Exploring Frequent Batch Auctions

[redacted]

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Thursday 4th May, 2023

Abstract

This dissertation investigates the emergence of the Frequent Batch Auction (FBA) as a new auction mechanism for financial exchanges. It aims to explore the performance of academically established algorithms, previously examined in Continuous Double Auctions, when simulated in an FBA. To achieve this, a first-of-its-kind simulator for FBAs, BFBSE, was created to evaluate different trading algorithms under the specific constraints of discrete time intervals and batch processing. By conducting over a million simulations, the research examines the pairwise relationships and dominance of 6 trading algorithms across 19 different ratios.

> My research hypothesis is that SHVR, 'a tongue-in-cheek model of contemporary high-frequency trading (HFT)' [9], would decline significantly due to FBA's design to curb the advantage of HFTs.

The results of the simulations conducted using BFBSE reveal that SHVR was able to maintain profitability in short batch intervals by adjusting the prices it shaves off. In fact, SHVR dominates both AA and GDX, indicating that it is a robust trading algorithm that can perform well in an FBA, despite the enforcing of discrete time intervals that minimise the speed advantages SHVR and other HFT algorithms possess.

Further to this the hierarchy of trading algorithms established in TBSE, a multi-threaded simulator of CDAs from which BFBSE was extended, was disrupted when tested in BFBSE. Surprisingly, the algorithm GVWY rose to the top of the hierarchy,demonstrating its effectiveness in the unique FBA setting.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

[redacted] Thursday 4th May, 2023

Contents

List of Figures

List of Tables

Ethics Statement

This project did not require an ethical review, as determined by my supervisor, Professor Dave Cliff.

Supporting Technologies

- BFBSE (Bristol Frequent Batch Stock Exchange) is a further extension of Threaded Bristol Stock Exchange[15], which itself is an extension of Dave Cliffs Bristol Stock Exchange [8].
- Amazon Web Services (AWS) was used extensively as a platform for cloud computing. This hosted a range of experiments which would have otherwise taken around a month running on a single machine

Notation and Acronyms

BSE	:	Bristol Stock Exchange
TBSE	:	Threaded Bristol Stock Exchange
LOB	:	Limit Order Book
FBA	:	Frequent Batch Auction
HFT	:	High Frequency Trading
ZIC	:	Zero Intelligence Constrained
ZIP	:	Zero Intelligence Plus
GDX	:	Gjerstad Dickhaut eXtended
AA	:	Adaptive Aggressive
GVWY	:	Giveway
SHVR	:	Shaver

Chapter 1

Introduction

1.1 Context

Financial exchanges' vast impact on everyday life is undeniable. In 2021, the financial services sector contributed £173.6 billion to the UK economy, 8.3% of total economic output [14]. A considerable part of this sector is comprised of traders who actively seek to capitalize on market movements to maximise their profits. Due to the development of the internet and the considerable money these exchanges produce, automated traders are now ubiquitous. What formerly comprised of verbal exchanges between individuals has changed into a battle between trading algorithms whereby agents are able to analyse large quantities of data and make a decision in a significantly quicker than a human trader would be able to. This new landscape has given rise to many academically validated algorithms, which have had their performance analysed with much scrutiny across varying market conditions.

Issues such as high-frequency trading (HFT) and a shifting focus towards short-term profits have caused widespread volatility and distortion in markets. In an effort to mitigate this, a new auction mechanism known as the Frequent Batch Auction (FBA) has become more popular. As a result, orders are processed in batches at discrete intervals and a clearing price is determined for all qualifying orders. With this, the incremental speed advantage possessed by traders is less advantageous than in the de-facto standard Continuous Double Auction (CDA); "the auction transforms competition on speed into competition on price" [5].

1.2 Motivation

The rise of the FBA as an alternative to the de facto standard continuous double auction mechanism has left a gap which needs to be explored and raises many questions on the dynamics of this novel auction. Using laboratory experiments, Aldrich and Vargas investigated the demand for a costly financial technology that reduced communication overhead and explored the difference in behaviour between traders in a CDA and a FBA [3]. They concluded that traders in a FBA displayed less predatory trading behaviour than the CDA, made fewer investments in low-latency communication technologies, had lower transaction costs, and saw less fluctuation in market spreads and liquidity. These findings show that FBAs may be able to mitigate some of the detrimental consequences of high-frequency trading and profit-driven trading strategies.

However, there exists a gap in literature which explores how established trading perform within this new auction mechanism. For example, it is not yet clear how well trading algorithms, which were developed for use in other market structures, will perform when operating within the context of FBAs. Additionally, it is still unclear whether the benefits of FBAs, such as reduced volatility and predatory behaviour, will be experienced equally by all types of traders, or whether some may still be able to gain an advantage over others. To address this gap in the literature, further empirical research is needed to assess the performance of trading agents within the context of FBAs, and to explore the implications of these findings for market design and regulation.

1.3 Aims and Objectives

The overall objective of this project is to undertake a comprehensive empirical analysis of existing trading algorithms. BFBSE (Bristol Frequent Batch Stock Exchange) is an extension of Dave Cliff's Bristol Stock Exchange [8] and aims to act as a minimal simulation of a limit-order-book financial exchange which uses FBAs. By using a first of its kind simulator of a FBA, I can evaluate the effectiveness of various trading algorithms under the constraints of discrete time intervals and batch processing, which they have not previously been tested with.

The concrete aims are:

1. Research the literature around Frequent Batch Auctions and their design.
2. Create a first-of-its-kind simulator for Frequent Batch Auctions.
3. Adapt existing algorithms so that they can perform effectively.
4. Perform an extensive analysis on the profitability of individual trading algorithms in this new environment.

Chapter 2

Background

2.1 Financial Markets

Traditionally financial markets acted as meeting places for buyers and sellers to agree on the price of a transaction. Traders would communicate verbally in order to barter with each other and determine the value of which their good could be sold at. The digitisation of this process in the 1980s brought forth a new platform whereby transactions could be executed through automated exchange systems. These were widely adopted by banks and so a new era of trading began. One landmark paper that demonstrated the superiority of automated trading strategies was the 2001 IBM IJCAI paper [10]. In this paper it was shown that automated trading strategies would "consistently obtain significantly larger gains from trade than their human counterparts". Experiments were conducted for the first time using a combination of both human and automated traders, rather than exclusively focusing on one or the other, and so conclusions could be drawn that showed trading agents ability in automated trading systems. It was stated that the "successful demonstration of machine superiority in the CDA and other common auctions could have a much more direct and powerful impact – one that might be measured in billions of dollars annually".

2.1.1 Microeconomics

The principles of microeconomics, which focus on how individual decisions impact the allocation of scarce resources, are also relevant to understanding the dynamics of financial markets. The competitive market for a simple good can be depicted graphically via a microeconomic model of supply and demand. This is shown in Figure 2.1.

The supply curve represents the relationship between the quantity of a good that producers are willing and able to supply and the price of the good. Generally, the supply curve slopes upwards, indicating that as the price of the good increases, producers are willing to supply more of it. On the other hand, the demand curve represents the relationship between the quantity of a good that consumers are willing and able to buy and the price of the good. The demand curve slopes downwards, indicating that as the price of the good increases, consumers are willing to buy less of it. The intersection of these curves represents the market equilibrium price, which is the point at which the quantity supplied and quantity demanded are equal. However, the volatility of financial markets means that this equilibrium price is subject to frequent and often significant changes due to market shocks, changes in the number of participants, and other external factors. As a result, the equilibrium price is in a constant state of change, and so investors must adapt their strategies accordingly.

2.1.2 Continuous Double Auction (CDA)

Auctions are a popular price discovery mechanism employed around the world. With auctions, buyers and sellers compete to reveal their valuations for an asset in an incentive compatible environment where competition is achieved through price. This mechanism is most useful when the value of the asset being exchanged may not be immediately obvious and is subject to change. The two widely used open auction mechanisms are the English auction and the Dutch auction. In an English auction, the auctioneer starts with a low price and it gradually increases through competition between bidders until only one bidder remains. At this point the auction is ended and the asset is then sold at the highest quoted price. In

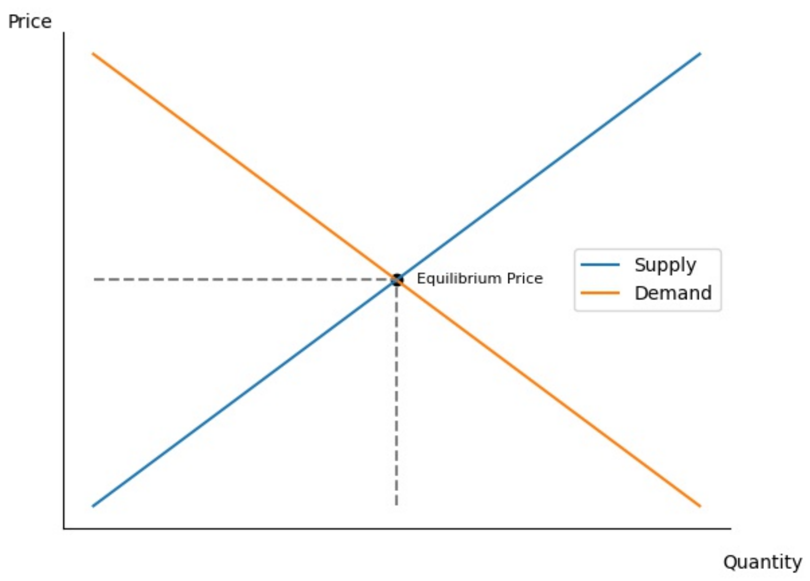

Figure 2.1: Supply and demand diagram

a Dutch auction, the auctioneer sets a high price and then progressively lowers it until a single bidder agrees to the quoted price, signaling the end of the auction and the sale of the asset at that price.

The Continuous Double Auction (CDA) is the most popular mechanism used by exchanges to determine the price of an asset and can be thought of as a combination of an English and Dutch auction.

In a CDA, buyers and sellers continuously submit orders to buy (bids) or sell (asks), and the exchange processes these orders serially based off time-priority. Each bid/ask processed is either matched to a counter party or added to the exchanges record so that it can be matched to incoming orders. Exchanges use a limit order book (LOB) to display all the buy and sell orders that are currently open for a particular asset. With this, traders can see the market depth and liquidity at different price levels, which can help inform their trading decisions. If a new bid is higher than any existing asks, or a new ask is lower than any existing bids, the market is cleared at the best price available on the limit order book. This is known as crossing the spread which results in the buyer and seller being matched and a trade being executed.

2.1.3 Limit Order Book (LOB)

Financial markets differ from traditional auctions such as English or Dutch auctions, which typically operate for brief periods of time. Instead, financial markets operate continuously, with trades occurring throughout the trading day and for a variety of assets. For this reason, a record of all outstanding bids and asks must be used in the absence of a central physical auctioneer.

The Limit Order Book (LOB) shows the ascending bid prices, where the highest quote price represents the best bids, as well as the descending ask prices, where the lowest quote price represents the best asks. From this, traders are able to infer qualities of the market. When the exchange receives an order that crosses the spread, meaning a bid with a price higher than the current best ask ("lifting the ask") or an ask with a price lower than the current best bid ("hitting the bid"), records are consumed from the top of each side of the limit order book. The specified quantity of the spread-crossing order is then taken off the top of the LOB, and the trade is executed at the price of the record consumed.

To illustrate this point, take the limit order book shown in Table 2.1. In this case, the spread is defined as the difference between the highest bid and the highest ask price listed on the limit order book (LOB). This value is currently 8. One measure of the current tradable price of an asset is the midprice. This is calculated as as the midpoint between the best bid and ask, which comes to 98.5 in our example. Further to this, if we wish to include the quantity of orders on each side of the limit order book in our calculation of the asset price we can use the micro price which has the formula expressed in equation 2.1. For our limit order book in Table 2.1 this has a value of 96.7.

$$Microprice = \frac{(Best\ Bid\ Quantity \times Best\ Ask\ Price + Best\ Ask\ Quantity \times Best\ Bid\ Price)}{(Best\ Bid\ Quantity + Best\ Ask\ Quantity)} \tag{2.1}$$

Bid		**Ask**	
Quantity	**Price**	**Price**	**Quantity**
5	95.5	101.5	20
30	92.0	102.5	50

Table 2.1: Example Limit Order Book

2.1.4 Vernon Smith

The study of CDAs was Pioneered by Vernon Smith whereby participants were split into groups of buyers and sellers and each given a limit price drawn from a distribution. Buyers were instructed to quote bid prices whereas sellers were told to quote offers into the market. In the seminal paper 'An Experimental Study Of Competitive Market Behaviour', Smith declared that rapid equilibration could be seen with only a very small numbers of traders in a CDA [18].

The subsequent studies presented in this paper are akin to those performed by Smith under the banner of Experimental Economics. Smith's research established the groundwork for this field and served as inspiration for numerous influential papers in the area of algorithmic trading, such as [13] and [10]. Analysis of human versus robot traders is central to these papers, and it is conducted using a methodology similar to that developed by Smith, but with the aid of computer simulations rather than laboratory environments.

2.2 Algorithmic Trading

The rise of automation in financial markets brought forth a new era of algorithmic trading whereby trading agents would work in a role of a traditional sales trader. Sales traders work for investment banks or brokerage firms and execute trades on clients behalf. The act as an intermediary between the clients and the markets so that a client can take advantage of market movements without extensive knowledge of how financial markets work. A typical customer order will be given to a sales trader and will specify how many units they want to buy or sell at a specific price. This price is known as the limit price. This states either the price that the customer is not willing to exceed when buying an asset, or the price the customer is not willing to go below when selling an asset.

Sales traders profit by generating an order for a customer that maximises their personal margin from executing the order. That is, traders who sell assets for customers want to maximise their margin M so that the price set $P = (1.0 + M)L$ is still attractive to buyers in the market. Similarly traders who buy an asset want to maximise their margin so that the price $P = (1.0 - M)L$ still attracts sellers to the market. The performance of algorithmic traders can be quantified by the value of the margin they achieve from trades over the course of a trading period.

2.3 Trading Agents

2.3.1 The Academic History

Gode and Sunder's 1993 paper titled "Allocative Efficiency of Markets with Zero-Intelligence Traders: Market as a Partial Substitute for Individual Rationality" [13] was one of the first seminal papers in algorithmic trading. Two trading agents, ZIU and ZIC, were developed by Gode and Sunder, which were referred to as 'zero intelligence' agents. These agents operated purely on randomly generated bid and offer prices, without relying on any information about the market conditions. ZIU, generated prices regardless of a customers limit price, leading to losses, whereas ZIC was constrained to never generate an order which would result in a loss for the customer. A surprising result of this paper was the similarity in performance between the 'zero intelligence' ZIC traders, and human traders.

Building on the work of Gode and Sunder, in 1997 Dave Cliff produced another 'zero intelligence' algorithm named Zero Intelligence Plus (ZIP) [6]. Similarly to ZIC, ZIP is constrained to never make a loss-making bid/ask but instead of using a random process to generate quotes, it instead uses machine learning heuristics to adjust it's quote price based off the state of the market. Shortly after in 1998, Gjerstad and Dickhaut produced an algorithm, later coined GD, in which buyers/sellers produced quotes to the market based on their belief that their price would be accepted [12]. The probabilistic belief function used by the traders was grounded using previous historical transactions. In 2001, a IBM's IJCAI paper found that both ZIP and MGD, a modified version of GD designed to operate within a limit order book, surpassed the performance of human traders.

In 2002, IBM continued its development of automated traders with the release of GDX, claiming that "this algorithm may offer the best performance of any published CDA bidding strategy" [20]. GDX, like its predecessor GD, utilizes historical belief functions to inform its trading decisions. However, GDX uses dynamic programming to achieve this, unlike GD.

In 2006 Vytelingum created Adaptive Aggressive (AA) which was part of a comprehensive analysis of AA,ZIP and GDX [22]. AA estimates the equilibrium price and uses this, along with an aggressiveness function, to make quotes into the market, with more aggressive agent placing bids/asks more likely to be accepted. Although Vytelingum first published that AA always dominated both ZIP and GDX, it was later discovered in 2015 that this is not the case when the ratios of other traders in the market compared to AA were adjusted [21].

The relevant history of trading algorithms concludes with a comprehensive analysis of the performance of established trading algorithms in academic literature, utilizing a multi-threaded approach [16]. Section 2.5.1 discusses TBSE in more detail, which was the simulator chosen to perform this analysis. From these studies a new hierarchy was published. At the top of the hierarchy is AA, which dominates all of the other algorithms and is considered the most successful in this multi threaded approach. Following closely behind are SHVR and GVWY, which each dominate 3 other trading algorithms. Following this, ZIP and GDX dominate 2 algorithms each. Lastly, the algorithm ZIC ranks at the bottom of the hierarchy, as it fails to dominate any algorithms in the comparison [9].

2.3.2 Trading Algorithms

The technical details of the trading algorithms examined in this paper are detailed below with Python3 code given from their implementation in BSE[8] where necessary.

Zero Intelligence Constrained (ZIC)

ZIC is the first of the zero-intelligence algorithms explored in this section. If it has received a new order from a customer, it generates a quote based off the result of a random probabilistic process and submits it to the market. For bids, the quote price is a random number between the lowest bid on the limit order book and a customers limit price. For asks, the quote price is a random number between a customer's limit price and the highest ask on the limit order book. The code for ZIC's `getorder()` function is shown in Listing 2.1

Zero Intelligence Plus (ZIP)

ZIP is the first adaptive algorithm and the second zero-intelligence method we investigate in this section. The structure of the ZIP version implemented in BSE closely mirrors that of the C reference implementation discussed in [6]. ZIP works by adapting its profit margin using a learning rule based off other traders actions in the market. This is seen in the `getorder()` function in Listing 2.2 . This is the same equation discussed in Section 2.2 whereby a profit margin M is maximised in a way that still quotes attractive prices to buyers/sellers in the market. In the case where the last order was accepted, the trader will adjust their quote price with the executed trade price. Conversely, if the trader's current quote price is more ambitious than an unaccepted price, they will decrease their margin.

First, ZIP estimates a target price using stochastic functions based off the activity in the market. The target price is calculated using Equation 2.2 where $q(t)$ is the latest quote, and $A_i(t)$ and $R_i(t)$ denote the absolute and relative stochastic components respectively.

$$t_i(t) = A_i(t) + R_i(t) \cdot q(t) \tag{2.2}$$

After this, the Widrow Hoff update equation is used to adjust the traders margin. This is defined in Equation 2.3. Where $\Delta(t)$ is defined as the difference between the desired output D and the actual output A, multiplied by the rate β.

```
def getorder(self, time, countdown, lob):
            if len(self.orders) < 1:
                    order = None
            else:
                    minprice = lob['bids'].['best']
                    maxprice = lob['asks'].['worst']
                    limit = self.orders[0].price
                    otype = self.orders[0].otype
                    if otype == 'Bid':
                            quoteprice = random.randint(minprice,limit)
                    else:
                            quoteprice = random.randint(limit,maxprice)

                    order = Order(self.tid, otype, quoteprice,
                                  self.orders[0].qty, time)
            return order
```
Listing 2.1: The ZIC trading algorithm's getorder() function.

$$A(t+1) = A(t) + \Delta(t) \tag{2.3}$$

By incorporating momentum, ZIP improves upon the Widrow-Hoff learning rule. In particular, the momentum factor reduces reactions to frequent changes in target price that may otherwise lead to instability in the learning process. With this factor, the algorithm is better able to track the underlying trend in the data and improve its overall performance which is particularly useful in volatile environments. The equation with momentum is described in Equation 2.4

$$A(t+1) = \gamma \cdot A(t) + ((1-\gamma) \cdot D_i(t)); 0 \leq \gamma_i \leq 1 \tag{2.4}$$

Gjerstad Dickhaut eXtended (GDX)

The algorithm GDX, which is examined in this paper, is the result of the algorithmic progression from GD and MGD.

GD's most notable feature was the inclusion of a belief function which is "formed on the basis of observed market data, including frequencies of asks, bids, accepted asks, and accepted bids" [12]. This is expressed in Equation 2.5. From this traders " traders choose an action that maximizes their own expected surplus" whereby the most profitable price is quoted into the market that is still expected to be matched to a counter party.

$$f(p) = \frac{able(p) + ole(p)}{able(p) + ole(p) + rbge(p)} \tag{2.5}$$

where:

- able(p) is the number of accepted bids/asks that are less than or equal to p in the set H
- ole(p) is the number of asks/bids made that are less than or equal to p in H
- rbge(p) is the number of rejected bids/asks that are greater than or equal to p in H

IBM modified the GD algorithm and renamed it MGD to conduct a comparative analysis between human traders and automated traders in a CDA. It was noted that the " cyclical behavior" of GD "was associated with high trade price volatility" when the recent market history contained no rejected bids/asks. MGD addressed this problem by incorporating weighted terms into the Equation 2.5. This allowed the system to respond not only to recent events like GD, but also to use knowledge from past events in circumstances when data from newer events was skewed. As a result, MGD improved the algorithm's ability to react to shifting market conditions while reducing the influence of irregular market data.

```
def getorder(self, time, countdown, lob):
            if len(self.orders) < 1:
                    self.active = False
                    order = None
            else:
                    self.active = True
                    self.limit = self.orders[0].price
                    self.job = self.orders[0].otype
                    if self.job == 'Bid':
                            # currently a buyer (working a bid order)
                            self.margin = self.margin_buy
                    else:
                            # currently a seller (working a sell order)
                            self.margin = self.margin_sell
                    quoteprice = int(self.limit * (1 + self.margin))
                    self.price = quoteprice
                    order=Order(self.tid, self.job, quoteprice,
                                self.orders[0].qty, time)

            return order
```

Listing 2.2: The ZIP trading algorithm's getorder() function.

GDX was the final evolution of the belief function oriented algorithms and at the time it was stated that "that this algorithm may offer the best performance of any published CDA bidding strategy" [20]. Algorithm 2.1 expresses the evolution of the state space for GDX. The algorithm starts with the initial state $V(\vec{x}, 0)$ and proceeds backwards over the bidding opportunities to compute the value of the state $V(\vec{x}, n)$ in terms of $V(\vec{x}, n-1)$.

```
for $n = 1$ to $N$ do
    for all reachable states $\vec{x}(n)$ do
        $V(\vec{x}, n) = \max_p \ (f(p, t_n)[s(p) + \gamma V(\vec{x*}, n-1)] + (1 - f(p, t_n))\gamma V(\vec{x}, n-1)$
    end
end
```

Algorithm 2.1: State Evolution of GDX

where:

- $\vec{x*}$ represents a traders state after trading
- γ is the discount factor
- $s(p)$ is the profit generated at price p
- $f(p, t_n)$ is the belief of a trade at price p and time remaining t_n
- N is the bidding opportunities

The dynamic state space evolution, along with the dynamic market history makes up the components of GDX. Finally, the value table $V(\vec{x}, n)$, is used to calculate the optimal action for a trader.

Adaptive Aggressive (AA)

AA is the most recent trading algorithm investigated in this paper and works by adapting its aggressiveness in a similar way that ZIP updates its profit margin. AA alters it aggressiveness based off a calculation of the equilibrium price of the market at a given time.

In order to generate the equilibrium price p^*, a moving average is calculated from from a list of the N most recent transactions. This is shown in Equation 2.6. With this, large price fluctuations are smoothed out and larger weights are given to the most recent transactions.

$$p^* = \frac{\sum_{i=T-N+1}^{w} w_i p_i}{\sum_{i=T-N+1}^{w} w_i} \text{where } w_T = 1 \text{ and } w_{i-1} = \lambda w_i \tag{2.6}$$

A trader's position is categorized as intramarginal if their limit price is more favorable than the equilibrium price, or extramarginal if their price is less favorable than the equilibrium price. The equations 2.7,2.8,2.9,2.10 are given in [22] and show how the quote price of AA traders is generated given an aggressiveness level r. Values of $r < 0$ represent aggressive traders who quote prices that are better than the equilibrium price and so are likely to be accepted, whereas values of $r > 0$ represent passive traders who quote prices that are less likely to be matched to a counter party since.

For an intramarginal buyer i,

$$\tau = \begin{cases} p^*(1 - re^{\theta(r-1)}) & \text{if } r \in (-1, 0) \\ (l_i - p^*)(1 - (r+1)e^{r\theta}) + p^* & \text{if } r \in (0, 1) \end{cases} \tag{2.7}$$

For an intramarginal seller j,

$$\tau = \begin{cases} p^* + (p_{max} - p^*)re^{(r-1)\theta} & \text{if } r \in (-1, 0) \\ p^* + (\hat{p}^* - c_j)re^{(r+1)\theta} & \text{if } r \in (0, 1) \end{cases} \tag{2.8}$$

For an extramarginal buyer i,

$$\tau = \begin{cases} l_i(1 - re^{\theta(r-1)}) & \text{if } r \in (-1, 0) \\ l_i & \text{if } r \in (0, 1) \end{cases} \tag{2.9}$$

For an extramarginal seller j,

$$\tau = \begin{cases} c_j + (p_{max} - c_j)re^{(r-1)\theta} & \text{if } r \in (-1, 0) \\ c_j & \text{if } r \in (0, 1) \end{cases} \tag{2.10}$$

where

- $r \in [-1, 1]$
- c_j denotes a seller j's limit price
- l_i denotes a buyer i's limit price

Variables r and θ are updated upon every new transaction in a market. The aggressiveness variable r is updated towards 'the desired aggressiveness' via a Widrow-Hoff rule similar to to that specified in Equation 2.3 that ZIP uses. The additional parameter θ, determines the rate at which the algorithm adjusts its target price based on the difference in the best bid and best ask. When θ is small, target prices are updated faster which is best suited to volatile markets. Larger values of θ are necessary when it is not beneficial to deviate too much from the equilibrium price p^*.

Giveway (GVWY)

GVWY agents are one of the two strategies bespoke to BSE explored in this paper. GVWY simply quotes its limit price to the market when a new customer order comes in. For this reason, its only chance of making a profit is if it is matched with a quote from another trader. The code for GVWY's `getorder()` function is shown in Listing 2.3.

Shaver (SHVR)

The SHVR algorithm is the second of the BSE-specific algorithms investigated in this paper. The goal of SHVR trading agents is to maintain the most competitive bid on the Limit Order Book (LOB) by altering their bid/ask price by a penny, as long as it does not result in a loss-making trade. If a customer order comes in which is a bid it will generate an order with a quote price of the best bid on the LOB plus a penny whereas if a customer order comes in which is an ask it will generate an order with a quote price of the best ask minus a penny. The code for SHVR's `getorder()` function is shown in Listing 2.4

```
def getorder(self, time, countdown, lob):
        if len(self.orders) < 1:
                order = None
        else:
                quoteprice = self.orders[0].price
                self.lastquote = quoteprice
                order = Order(self.tid, self.orders[0].otype,
                              quoteprice, self.orders[0].qty, time)
        return order
```

Listing 2.3: The GVWY trading algorithm's getorder() function.

```
def getorder(self, time, countdown, lob):
        if len(self.orders) < 1:
                order = None
        else:
                limitprice = self.orders[0].price
                otype = self.orders[0].otype
                if otype == 'Bid':
                        if lob['bids']['n'] > 0:
                                quoteprice = lob['bids']['best'] + 1
                                if quoteprice > limitprice :
                                        quoteprice = limitprice
                        else:
                                quoteprice = lob['bids']['worst']
                else:
                        if lob['asks']['n'] > 0:
                                quoteprice = lob['asks']['best'] - 1
                                if quoteprice < limitprice:
                                        quoteprice = limitprice
                        else:
                                quoteprice = lob['asks']['worst']
                self.lastquote = quoteprice
                order = Order(self.tid, otype, quoteprice,
                              self.orders[0].qty, time)

        return order
```

Listing 2.4: The SHVR trading algorithm's getorder() function.

2.4 Frequent Batch Auction

The emergence of high-frequency trading has led to concerns about market liquidity, price manipulation, and volatility. A High Frequency Trader (HFT) is an automated trader that is able to work a profit, purely from the speed at which it can carry out transactions. One of the ways in which HFTs achieve this is by crossing the spread when stale quotes exist on an exchange. Stale quotes are old quotes that are no longer valid or reflective of the current market conditions. This can occur when market prices change rapidly due to market shocks. Due to the serial nature of order processing in CDAs, it is common for high frequency traders to be able too react to changes in market conditions before these stales quotes can be cancelled, thereby crossing the spread and transacting these quotes at a beneficial price.

Over the past decade, an alternative mechanism named the Frequent Batch Auction (FBA) has gathered momentum [5] in order to combat the rise of HFTs. FBAs, in contrast to CDAs, match orders intermittently rather than continuously, with pre-defined batch periods used to match orders.

Buyers and sellers can freely submit, modify or cancel from the exchange, in the same way they can do in a CDA. Once the order submission period has ended, the equilibrium price is determined based on the orders placed during that batch and the state of the LOB at the time of execution.

There are two possible outcomes during a batch period in an FBA. If the spread is crossed, meaning the highest bid is greater than the lowest ask, trades occur at the equilibrium price, otherwise there is no trades. Although this change may seem trivial, it can have huge effects on a market and “modifying the market design from continuous-time to discrete-time substantially reduces the value of tiny speed advantages” that are exhibited by HFTs [4]. A notable example of an FBA inspired exchange is The London Stock Exchange’s Turquoise Plato, in which Plato clients had traded more than €1.1 trillion in equities by its 5 year anniversary [17]. In general, the implementation of FBAs by exchanges like the Turquoise Plato of the London Stock Exchange underscores the need for alternative trading methods that can level the playing field for all traders.

2.5 Bristol Stock Exchange

Bristol Stock Exchange (BSE) is a minimal simulation of a centralised financial market and was developed by Dave Cliff as an educational tool in 2012 [8]. It works by maintaining a copy of a LOB for a single unnamed stock in the same way that an exchange would. Trading agents are polled at random and tasked with dealing with new market information, updating internal records, and producing new market orders. Unlike simulators based purely solely on replaying historic data, traders in BSE may exhibit a market impact when placing orders. A market impact might cause the price of the asset traded on the exchange to increase/decrease significantly if a trader places a sufficiently large bid/ask. BSE’s main advantage over other simulators is its ability to account for market impact, which is a crucial factor in financial markets.

2.5.1 Threaded Bristol Stock Exchange

Although BSE is a powerful tool it is not a true representation of a modern-day exchange as it abstracts away some of the complexities that are common across financial markets. Due to the single threaded nature of BSE, the random polling phase of each trading agent gives them an arbitrarily long amount of time to make their decision on which action to take next and so no other trading agents have the opportunity to ‘steal the deal’. Threaded Bristol Stock Exchange (TBSE) [15] is a parallel extension of BSE that models the multi-threaded execution of traders so that all traders can perform their calculations concurrently. To achieve this, it employs threads, enabling simultaneous execution of the exchange and each participating trader in the simulation.

TBSE’s multi-threaded approach was selected as a more suitable fit for my simulator because it can replicate the quote sniping behavior that is a characteristic of the CDA mechanism that FBAs seek to address.

Chapter 3

Project Execution

3.1 Understanding Threaded Bristol Stock Exchange

The first step towards extending TBSE to provide the functionality of Frequent Batch Auctions was to gain a comprehensive understanding of the simulator and its implementation. Compared to BSE, TBSE's structure is relatively straightforward as it comprises 7 modular files, whereas BSE, being designed for simplicity, only occupies a single file. However, a great deal of time was still needed to understand the details of a 2500+ line codebase. The use of the wiki on the BSE GitHub page [8], along with the papers written by Rollins & Cliff, helped provided a good provisional understanding [16][9]. I then furthered my understanding by running the simulations with a small number of traders in `verbose` mode, which produced extensive and detailed output regarding the state of the limit order book and traders throughout the simulation.

The next step taken was to understand the implementation of the trading algorithms in TBSE. This was a critical stage as it would allow me to validate that the results of a batch auction were consistent with the expected outcomes and examine how they differed from the established academic hierarchy, detailed in Section 2.3.1. For instance, by understanding the implementation of the SHVR trading algorithm, which acts as a simple model for a High-Frequency Trader, I was able to understand that possible implication of it relying heavily on the state of the limit order book when determining its quote price to the market. This process was particularly challenging for complex algorithms such as GDX, AA, and ZIP, which spanned hundreds of lines of code. Nonetheless, with the aid of the wiki and academic papers that were relevant to the trading algorithms, I was able to acquire a sufficient understanding of them.

3.2 Bristol Frequent Batch Stock Exchange (BFBSE)

Once the intricacies of TBSE were understood, I was able to begin modifying the code in order to create a platform to simulate batch auctions. While many implementations of Frequent Batch Auctions have been discussed and implemented in real-life, my simulator reproduces the behaviour specified in Implementation Details for Frequent Batch Auctions: Slowing Down Markets to the Blink of an Eye by Budish et al. [4].

BFBSE maintains much of the same code that is used in TBSE but a number of key changes are made to give the required performance of Frequent Batch Auction. Budish et al. define the process flow of FBAs as having three components: order submission, auction and reporting.

The order submission is not dissimilar to CDA and orders can be freely submitted, modified, or withdrawn. BFBSE uses the same queue structure as TBSE whereby orders are submitted to a order queue, and the result of trades submitted by a trader, along with all the transactions processed by the exchange, are sent to traders via another queue. At any point during the trading day, traders are able to cancel their orders, in the same way that they can in a continuous double auction. Orders are cancelled by traders sending a request to the kill queue. As a result, the communication between traders and the exchange follows the same channels as those created in TBSE.

The auction phase occurs at regular batch intervals and is the stage at which a clearing price is determined. This is achieved by examining the state of remaining orders on the exchange from previous batches, along with the orders submitted during the most recent batch. This phase is significantly different from TBSE and required the most thought when extending TBSE.

The reporting phase relates to the announcement of the result from processing of the orders on the exchange. This includes the aggregate supply and demand curves from the batch, along with the announcement of the equilibrium price and quantity. While this may seem like a significant change from the LOB whereby traders can see the price and quantity of orders submitted by buyers, along with the spread and micro price as detailed in Section 2.1.3, this is analogous to how the LOB in the continuous market represents liquidity provision.

Any further arbitrary design decisions made in the implementation of BFBSE were made to be the most similar to that of the CDA implemented in TBSE.

The key modifications to TBSE are detailed further throughout this section.

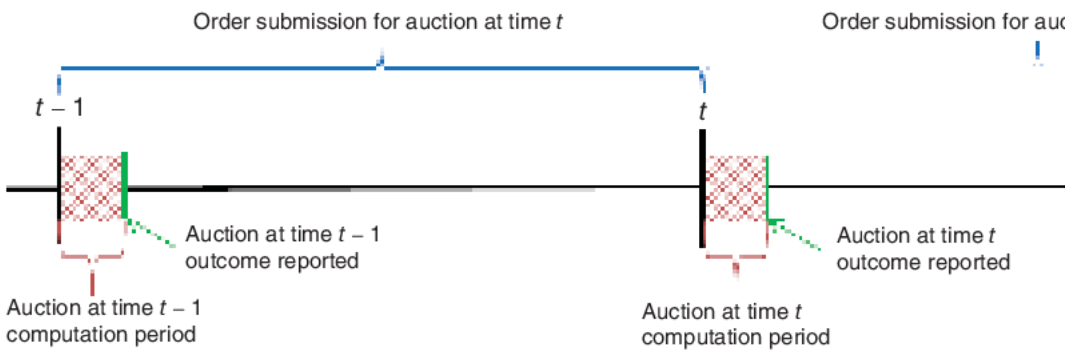
Order submission for auction at time *t*
Order submission for auc
t – 1
t
Auction at time *t* – 1 outcome reported
Auction at time *t* outcome reported
Auction at time *t* – 1 computation period
Auction at time *t* computation period

Figure 3.1: Frequent Batch Auction Stages [4]

3.2.1 Batching Mechanism

The batching mechanism differs widely from that implemented in TBSE. In TBSE, orders are processed serially by the exchange where they are examined to see if they cross the spread and can be matched with a counter party on the limit order book. BFBSE, instead works by storing orders as they arrive into the exchange via a queue. When a trader submits an order, a check is performed to determine if the trader has already submitted an order to the exchange or within the current batch period. If this is confirmed, the new order from the trader replaces the old order. This is inline with the simplification included in both TBSE and BSE where traders can only have one order on the exchange at a single time. The time elapsed since the last batch process is then calculated each time the exchange thread is run. If the time elapsed is greater than or equal to the batch interval, the auction process begins and the outstanding orders are processed by the exchange.

Orders are categorised into orders from the most recent batch and longstanding orders on the exchange. Orders are then sorted by price and demand and supply curves are produced. From this an equilibrium price can be calculated which is used as the clearing price. Orders are then characterised using this price. If the intersection of the supply and demand curves is horizontal, all orders to buy with a price greater than the clearing price and all orders to sell with a price less than the clearing price transact their full quantity. In the case where there are orders with the same price as the equilibrium price, it will only be possible to fill one side of the market. In this case, Buddish et al. suggest “time priority across batch intervals but not within batch intervals” so that long standing orders are encouraged, as in exchanges which use a CDA. This means that orders from earlier batches are given priority over the latest batch, but orders are not given time priority within a batch. If the intersection is vertical, then no rationing is needed. The price is simply set as the midpoint of the intersection.

The completed trades from the auction are then sent to a queue so traders can react to this information. Remaining orders are then carried through to the next batch period. The steps of the batch process are shown in Figure 3.1.

3.2.2 Limit Order Book State

In TBSE, the limit order book structure is used to store all bids and asks on the exchange during the course of the trading day. Orders are removed from the LOB when they can be matched to a quote that crosses the spread. Each time this occurs, the best bid/ask is removed from the opposite side of the limit order book and the transaction is recorded and sent to the traders so that they can update their respective internal values that influence their order submission.

In order to implement the time priority across batch intervals, discussed in Section 3.2.1, I used TBSE's limit order book to store remaining bids from each batch. With this, it was possible to use the existing TBSE infrastructure in the implementation of a FBA.

Furthermore the use of the LOB, is comparable to the aggregate supply and demand curve used in the reporting stage of a frequent batch auction and so this was used as an intermediary before the demand and supply curves could be reproduced programmatically.

3.2.3 Equilibrium Price Calculation

In order to determine the equilibrium price used for the clearing price for each auction batch, an addition had to be made to the `Exchange` class used in TBSE. As mentioned in Section 2.1.1, the equilibrium price is determined by the price at which demand is equivalent to supply. However, in financial markets the demand and supply curves are stepped as they represent demand at discrete levels, rather than a continuous price. An illustration for how the equilibrium price is calculated in FBAs is shown in Figure 3.2 where the supply curve represents all sellers with orders at the exchange, and the demand curve represents all buyers with orders at the exchange.

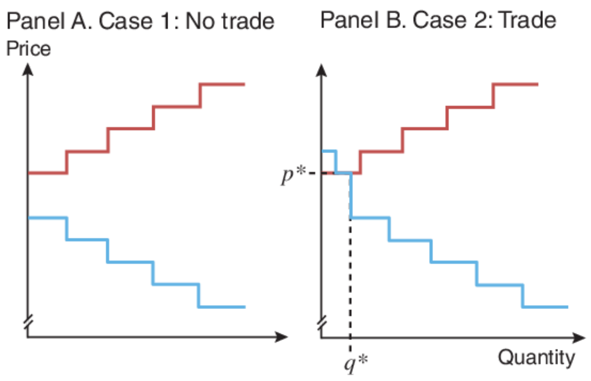

Figure 3.2: Frequent Batch Mechanism [4]

Due to the frequency of batches, the equilibrium price is calculated regularly with only a small number of orders on the exchange. Furthermore, at the start of a trading day, the bid-ask spread is large and so it can be harder to pin down an equilibrium price for a given bath of orders. As a result, the calculated equilibrium price may not correspond to an actual market price, but instead may represent the midpoint of two intersecting prices. In keeping with the portable style of BSE and TBSE, the function used to calculate the equilibrium price was written by myself and not imported from another module, such as QuantLib [2]. This continues the portability and simplicity of BSE and TBSE.

In order to calculate the equilibrium a function was written which loops over the prices in the supply and demand curves, calculating the consumer and producer surpluses at each price. It then compares the net surplus to the smallest net surplus seen so far, and updates the best supply and demand prices if the current net surplus is smaller. Finally, the method calculates the equilibrium price as the midpoint of the best supply and demand prices.

3.2.4 Adapting Trading Agents

In TBSE, trading agents are assigned there own thread and which executes continuously until the end of a trading day. The first method executed is the `bookeep()` method. This is responsible for updating a traders internal records of transactions and calculates the total profit and the number of trades made, remains unchanged from the code in TBSE.

Following this, the `respond()` method is used to update a traders internal values based on the most recent market activity and trades executed in the market. This is where much of the complexity is encoded into trading algorithms. For example AA, GDX and ZIP notably use this function to update their aggressiveness, belief functions and margin respectively as detailed in Section 2.2. Finally, the `getorder()` function is used which determines if an order should be generated in the market and at what price the trader should issue the order at. Due to the bespoke nature of the `getorder()` and `respond()` functions, these had to be changed to take into account the new information broadcasted to traders.

```
def get_order(self,time,p_eq ,q_eq,
    demand_curve,supply_curve,countdown,lob):
          if len(self.orders) < 1:
               order = None
          else:

               coid = max(self.orders.keys())
               limit_price = self.orders[coid].price
               otype = self.orders[coid].otype

               if demand_curve!=[]:
                    best_bid = max(demand_curve, key=lambda x:
                       x[0])[0]+1

               if supply_curve!=[]:
                    best_ask = min(supply_curve, key=lambda x:
                       x[0])[0]-1

               if otype == 'Bid':
                    quote_price= best_bid
                    quote_price = min(quote_price, limit_price)
               else:
                    quote_price = best_ask
                    quote_price = max(quote_price, limit_price)

               order = Order(self.tid, otype,
               quote_price, self.orders[coid].qty, time,
                   self.orders[coid].coid,self.orders[coid].toid)
               self.last_quote = order

          return order
```

Listing 3.1: The SHVR trading algorithm's adapted getorder() function in BFBSE.

As shown in Figure 3.1, Frequent Batch Auctions have three stages: order submission, auction and reporting. The reporting stage differs from that in CDAs by the state of the market being reported using aggregate supply and demand curves, along with the equilibrium price and quantity.

The original implementation of SHVR, seen in Listing 2.4, aims to have the best quote on its side of the LOB. It achieves this by 'shaving' one unit off the best bid or best ask at the time of issue. However, BFBSE uses the LOB structure to store remaining bids and so in its existing implementation SHVR will only examine quotes that haven't crossed the spread, and hence do not represent the top bids or offers. Using the demand/supply curve for it's appropriate order type, I redesigned SHVR to 'shave' a penny off the best bid/ask from the most recent batch. This implementation is seen in Listing 3.1.

In a continuous double auction, the mechanism implemented by TBSE and it's predecessor BSE, orders are matched continuously. Orders can be matched within milliseconds of traders making a bid and so the most attractive bids in the market, the lowest asks and the highest bids, exist only for a short time before the spread is crossed and they are matched to a counter party. For this reason, traders can understand a market quickly by observing the state of the LOB for a short period of time. However, in a FBA orders are matched in intervals rather than continuously and so even the most attractive bids/asked cannot be executed faster than the batch frequency used at the exchange. A consequence of this is that, traders cannot gauge the attractiveness of a bid strictly by the time it takes to be matched, unless that time is longer than a batch period. To combat this, each traders implementation were adapted so that they would only update there internal values via the `respond` function when the reporting stage of each batch was finished.

3.3 Workflow

Over the course of the project, a number of decisions were made that impacted the production and testing of BFBSE.

Firstly, Git was used extensively as a version control system in order to keep track of incremental changes made to the codebase. For example, by using Git it became possible to simulate batch auctions with slightly different implementations from the same repositories. An example of this is the implementation of SHVR which was tested extensively in order to examine the difference in performance between shaving a penny off the best order at a given time and the best remaining order. An additional example is the use of Git in order to test implementations with varying batch periods in order to observe the effect on the academic hierarchy of published algorithms.

Another decisions that was notable was the cautious approach to the validity of the codebase on which my extension was built. Similarly to BFBSE, TBSE was built by a student over a short duration as an extension to a existing program. TBSE is highly successfully and has yielded publishable results, however due to the time constraints on which it was built, it was highly probable that bugs or hidden side effects were introduced which could have an affect on the performance of BFBSE. By reading through the paper published by Michael Rollins (the author of TBSE), along with the extensive comments throughout the program, I was able to gain an understanding of the structure of the project and identifying bugs became significantly easier.

For example, in the latest version of TBSE, there was an issue with the SHVR trading agent's implementation when dealing with customer sell orders. Specifically, the agent was programmed to select the lower value between its limit price and the quote price generated by shaving a penny off the best quote. While this approach is reasonable for traders placing bids, as they should not bid above their limit price to avoid losses, it is not suitable for customer sell orders. This is because, if the quote price generated is less than the price on the limit order book, a loss-making price will be quoted to the market.

3.4 Experimental Methodology

3.4.1 Design

A large component of the execution of the project was the experimental design used when comparing the performance of trading algorithms. Since FBAs transform the dynamics of the market for an asset from an ascending auction to a uniform-price sealed-bid double auction, the main objective of the tests performed throughout the project were to examine how these changes affect the performance of academically established traders.

Given that trading algorithms are often created using complex mathematical models and that their behaviour is strongly influenced by a variety of external circumstances, gaining a thorough analytical knowledge of how these algorithms function can be difficult. Even if simplifying assumptions are used to make their study more manageable, these assumptions can distort or oversimplify the reality of real-world financial markets. As a result, empirical studies, such as those detailed in Section 2.1.4, were the chosen method of choice for examining trading strategy performance.

During the early stages of development, the performance of the trading algorithms could be briefly examined by running a small number of simulations and observing the profit made by each trading agent type in an individual simulations. However, a large component of the profit made by traders is not the just in the bidding strategy it implements but also the limit price it receives from customer orders. If a trader's limit price is too high and it is required to create a bid for a customer, it may miss out on the opportunity to quote prices that are appealing enough to be matched by a counter party.

Customer Orders

In order to carry out empirical studies of trading algorithms, a decision must be made about how limit order prices are distributed to traders. In IBM's seminal 2001 paper[10], limit order prices are generated through a uniform random distribution. Each trader is assigned 10 limit prices from a random distribution between between 100 and 200. These are then refreshed at the end of each trading period. Similarly, in [11], traders are evaluated over 10 consecutive rounds, with each trader receiving a new supply of 13 orders to sell in the market during that particular round. Although, both [10], and [11] simulate supply shocks by manipulating the limit price periodically, this was ignored in order to pin down the basic change in dynamics for FBAs.

In both BSE, TBSE and BFBSE, the limit price a customer receives is specified in the `config.py` file. The distribution of order prices can be defined using 3 possible values: ‘fixed’, where prices have a constant difference between them; ‘jittered’, where prices are generated in a similar way to ‘fixed’ but with a small amount of random noise; and ‘random’, where prices are drawn at random from a uniform distribution [8]. From the list of limit order prices in the market it is possible to draw supply and demand curves.

For my project, the supply and demand curves are symmetric and so both the bid limit prices and the ask limit prices are drawn from an identical distribution. For a given run, a maximum value is generated between 100 and 200, and minimum value is generated between 1 and 100. The limit order prices are then drawn from a uniform random distribution between the maximum and minimum values. Each session simulated a virtual time of 600 seconds, with replenishment of orders happening periodically every 30 seconds. This totals 19 unique orders per trader agent per market session. This follows the style of experimental economics in [18] and so is a good framework for comparing results to existing papers.

Conventional Testing Methodologies

However, because of the inherent randomness of limit orders in a given run, in order to produce accurate results a statistical experiment had to be run which was able to average out the distribution of limit orders and produce results which analyse the performance of traders based only on their strategic decisions.

IBM’s 2001 IJCAI paper [10] first explored the use of two types of testing methodologies, one-in-many tests and balanced-group tests, to evaluate the agents’ performance. This was a novel approach which differed from the conventional method of testing trading agents in homogeneously populated markets, where only one type of trader was present in a market experiment. One-in-many tests involve running a single instance of a trading agent against multiple instances of other trading agents. This method allows researchers to test how well the trading agent performs against a variety of other agents that have different bidding strategies and decision-making processes. Balanced-group tests, on the other hand, involved creating groups of buyers and sellers that are evenly split between two types trading agents. Every trader has a counterpart who has an identical limit price but uses the other trading strategy.

While one-in-many tests and balanced-group tests have their advantages in evaluating trading agents’ performance in strict pre-defined scenarios, they are limited when trying to evaluate a traders performance in a real-world financial market. For example, a real-world financial market would be populated with trading agents running various strategies. Moreover, traders would have to operate with a wide range of variables and not just predefined market scenarios with similarly distributed limit orders. The following section presents two experiments used to evaluate the performance of traders in a FBA.

Varying ratio

As an alternative to one-in-many tests or balanced group tests, exhaustive testing across a wide range of market scenarios was used, as explored in [21] [7][19]. The availability of large-scale cloud computing facilities for consumer use has made it possible to perform thorough testing, which wasn’t possible during the onset of algorithmic trading. Snashall and Cliff observed that by varying the ratio of 4 traders in the market, academically established results do not always hold. Trading strategies which seemingly dominate others do not always dominate across varying ratios.

With this in mind, I performed an analysis of the results of varying the ratios of traders, so that the performance of trading algorithms could be examined across varying market levels. The trader ratios were changed to give each trader type a different number of agents on each side of the market, ranging from 0 to 5. A market with 10 ZIC agents and 2 GVWY trading agents would be represented with a ratio of 5:0:0:0:1. An experiment of this form requires 6^6 distinct trading days to represent all possible combinations of trading agents in the market. Additionally, in order to produce accurate results, it is preferable to perform analysis on an average of runs over a distinct trading day. A particularly conservative estimate of 10 runs per trading day was used which resulted in 466,560 simulated trading days, each lasting a second, totalling roughly 5 and a half days of simulations. After gathering the results from this experiment I began plotting the average revenue generated per trader by the number of agents it had in each trading day simulation. However, from this it was hard to draw conclusions as the results from each of the 10 runs differed substantially.

An avenue for future work might include averaging over a larger number of simulations for each trading day, but this would have taken order of magnitudes longer and so would have cost significantly more when simulated on the cloud, which is discussed in Section 3.4.2. For this reason, I decided to use another experimental design for comparing the performance of trading strategies. With this I was able to reduce the variability of results and focus mainly on the change in dynamics resulting from switching from a CDA to a FBA.

Pairwise Tests

Inspired from the tests run in [16], I chose to instead use pairwise comparisons of the 6 trading algorithms as a method for comparing the performance of different trading strategies in a FBA. Tests were conducted by varying the ratio of trading strategies used across a fixed number of total traders in each simulation. Each trading day was populated with a total of 20 buyers and 20 sellers, totalling 40 traders, which was the recommended limit of the number of threads operating simultaneously without causing potentially unstable behaviour. For a test of two algorithms, say ZIC and ZIP, all possible ratios were explored that gave a total of 40 traders in the market. A ratio of 19:1 would represent a total of 38 ZIC traders and 2 ZIP traders. This ratio was varied in increments of 1, resulting in 19 different ratios ranging from 19:1 to 10:10 to 1:19, all of which were examined for each pair of trading algorithms.

Due to the smaller number of combinations explored in pairwise comparisons, I was able to perform 1000 simulations for each distinct ratio of traders. Each pair of the 6 algorithms totalled 15 pairs, with 19 ratios, were explored, totalling just over 3 days worth of simulation time for each pairwise sweep.

The use of pairwise comparisons to analyze the performance of trading algorithms offers several advantages over an exhaustive analysis. One major advantage is that it allows for a large number of results to be generated over a set of ratios, without incurring the high computational cost of an exhaustive test. By comparing each pair of agents directly, I was able to focus on the most relevant and interesting combinations, rather than having to simulate all possible combinations. Moreover, pairwise comparisons provide a more direct and targeted approach to investigating the differences in performance between different trading algorithms. By limiting each simulation to having only two types of traders, the interactions between trading agents can be more directly examined and possible market phenomena can be broken down more easily.

Furthermore, by using the same experiments described in [16], I was able to compare the results from TBSE and BFBSE and make conclusions about the new market dynamics induced from a FBA. This involved using the same trader specification and order schedule, which is defined in a configuration file `config.py`, and noting the differences in results.

3.4.2 Execution

To ensure the reliability of the conclusions made, a large number of simulations had to be run for each market scenario. Fortunately, the task of running simulations for financial market analysis is embarrassingly parallel, meaning it can be parallelised without any communication between processes. In performing the 285000 required simulations for each accurate pairwise sweep, I was able to utilise cloud computing services in order to generate results faster than would be possible on a single machine.

Amazon Web Services is the most widely adopted cloud platform and offers the functionality to parallelise the simulations in a cost efficient manner. Using the free tier [1], I was able to utilise 750 hours of virtual machines and access to 5GB of storage on a limited budget.

One of the key services provided by AWS is Elastic Compute Cloud (EC2), which allows users to rent virtual machines in the cloud. EC2 instances with minimal specifications (‘t3.micro’) were a sufficient to run the number of simulations needed in an appropriate time as TBSE is designed to be run on a personal computer and does not require specialist hardware. By renting a separate machine for each of the 15 pairs of algorithms, I was able to reduce the time needed for an extensive pairwise comparison of algorithms from just over 3 days to a matter of hours.

Simple Storage Service (S3) is another valuable AWS service, offering scalable object storage in the cloud. By utilizing the command-line tools, I was able to store data from a wide range of simulations in a single ‘bucket’. This provided a centralized storage location for the results that were yet to be analyzed, allowing for easier organization and accessibility.

A big motivator for the use of cloud computing in the testing process was [19] where it is stated that “Surely then the broader methodological lesson here is that we should not allow ourselves to be seduced by results from small-scale studies in minimally simple approximations to real-world markets. [Vernon] Smith developed his experimental methods in the late 1950’s when there were no realistic alternative ways of doing things. Running experiments with human subjects is laborious and slow, but experiments in electronic markets populated entirely by robot traders can proceed in appropriate simulators at speeds much faster than real-time”..“ surely trading-agent researchers should collectively abandon the simple minimal test-environments that worked well for Vernon Smith in the middle of the 20th Century and instead start to tolerate the minor inconvenience of running very large numbers of trials on reasonably accurate simulations of realistic market situations".

3.4.3 Analysis

After the results were generated and stored with the aid of cloud computing I was able to perform an analysis. In order to measure the performance of traders in a FBA, a number of metrics were considered.

Profit Dispersion (PD), first defined in [13], is a metric used to measure the performance of a single trader and was first used to discriminate between the profit seeking human traders, and the ZIC trading agents who generate random prices from a distribution. PD is calculated by dividing the profit made by a single trader by the profit made if every transaction was executed at the equilibrium price p^*. Lower value of PD represent traders who are able to extract the most profit from a market relative to the theoretical maximum profit.

However, I felt that the best measure of performance was the average profit per trader over the course of a trading day. With this, traders are not analysed based off the profit they can extract from the market, which is irrelevant to us, but instead the focus is on how the performance of traders is changed in a FBA. It should also be mentioned that a higher average profit per day can be brought about from executing a larger number of trades, which is relevant to real-world financial markets where the main objective of sales traders is to maximise total profit. On the other hand, the total trades executed by a trader is not relevant to its PD, and performing an additional trade, which is less profitable than the last trade made, would actually harm the PD scored by an algorithm.

Using the average profit per trader, I calculated the number of wins that each algorithm scored in order to access the hierarchy of algorithms in an FBA. This is explored in Chapter 4.

Chapter 4

Critical Evaluation

4.1 Assessing Performance

The following results were generated from the experiments described in Section 3.4. Each experiment consists of a sweep across all possible pairs of the 6 algorithms discussed in Section 2.3.2. Each of the 15 pairs is examined across all possible ratios in a market which has a total of 40 traders, 20 buyers and 20 sellers. For each ratio, 1000 trials are run, and from each trial the winning algorithm is declared as the one with the greatest profit per trader. The algorithm that wins more trials than its counterpart across the 19,000 trials generated is said to dominate it in a pairwise comparison. Through 15 different pairwise tests across 6 algorithms, we conduct a total of 285,000 trials per experiment. Using wins as a success metric for algorithm comparison, we can establish two hierarchies: one by ordering algorithms based on the number of algorithms they dominate, and the other by comparing their total wins against every other algorithm.

4.1.1 Delta Curves

From a single pairwise test, we can gain a further understanding of the dynamics between the two trading algorithms by using delta curves. These are produced by plotting the difference in the number of wins between 'Algorithm A' vs 'Algorithm B' for each ratio examined. If the difference of wins is positive then 'Algorithm A' scores more wins, whereas if the difference of wins is negative then 'Algorithm B' scores more wins for a given ratio. Analysing the difference in the number of wins between two algorithms at different ratios allows us to identify points of peak performance where one algorithm performs significantly better than the other. These points are defined as the moments where the difference in wins between the two algorithms is greatest. This approach enables us to gain a more detailed understanding of the results beyond just identifying which algorithm dominates another across a set of trials. This was notable in [16], where by using delta curves it was easier to examine the changes in results of trading algorithms when switching between the single threaded BSE and the multi threaded TBSE simulator.

4.1.2 Varying Batch Interval and Equilibrium Price

Each experiment consists of the results of each trading strategy generated from a pairwise sweep across a different batch interval. By changing the batch interval to be progressively more frequent, we can observe which agents perform better with longer batch intervals, and which agents perform better with relatively frequent batch intervals, where the state of the market is reported to traders more frequently. With a smaller batch interval we expect the market to become more similar to a CDA and so results may be more similar to those recorded in [9]. By performing experiments with varying batch intervals it also becomes possible to deduce the best batch interval to prevent certain strategies. Such strategies are employed by HFTs, which often increase volatility in the market. .

It is also important to observe the effects of varying the equilibrium price. As discussed in Section 3.4, for a given trial, limit prices are drawn from a random distribution between a minimum value between 1 and 100 and a maximum value between 200 and 100, giving an average equilibrium price of 100. This is maintained throughout the trading day so that customer orders are distributed where the average transaction price should be 100. However, in order to produce results from a simulation which accurately reflects a real-world financial market it is preferred to use a varying equilibrium price. To alter the equilibrium price throughout a trading day, an offset function was used in a separate experiment for each batch period. This function adjusts limit prices based on time, resulting in the equilibrium price varying during the trading day.

In the case of experiments with the longest batch intervals (10 seconds), I made a deliberate decision not to examine the impact of varying the equilibrium price. This is due to the fact that the FBA mechanism already offers traders a restricted view of the market, since the reporting mechanism provides them with information less frequently than in a continuous double auction. Consequently, I chose to avoid introducing any further complexity that could interfere with the results obtained from less frequent batch intervals.

4.2 Results

The following sections will detail the findings of the experiments and clarify the justification for conducting further experiments. The results of each experiment will be summarised by stating the number of wins and algorithms beaten by each trading strategy. I will then plot delta curves to explain the interaction between traders where necessary.

4.2.1 10 Second Batch interval

Initially, I generated results using a longer batch interval as I was interested in observing the distinct outcomes when the new testing environment for evaluating trader performance was the most dissimilar to a CDA. This is because an FBA with large batch intervals has more extended periods of inactivity, where the perceived market state from the previous reporting stage can significantly differ from the actual market state. As a result, there are significant price fluctuations in transaction prices between batches, which is a characteristic feature of FBAs compared to CDAs. The set of results for the first experiment are shown in Table 4.1.

Old SHVR Implementation

Trading Algorithm	Total Wins	Algorithms Beaten
GVWY	94537	5
ZIP	66684	4
GDX	60162	3
AA	37466	2
ZIC	22797	1
SHVR	3354	0

Table 4.1: Wins and algorithms beaten by each trading algorithm in a 10 second batch interval FBA with the old SHVR implementation

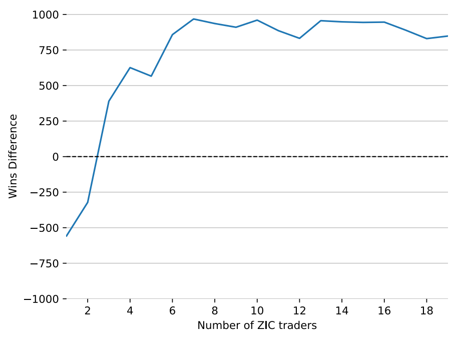

Figure 4.1: ZIC vs SHVR Delta Curve for results from a 10 second batch interval

One particular result that revealed itself was the poor performance of SHVR. This is seen in Figure 4.1 where SHVR is only able to out perform the ZIC trading strategy with a small number of agents on each side of the market. This is a surprising result as in both TBSE and BSE, SHVR is known to dominate ZIC with both a static and dynamic equilibrium price [9]. As SHVR reacts to the state of the market it is surprising that it is outperformed by a strategy which generates quote prices from a random distribution such that it doesn't make a loss. In order to address this result, subsequent experiments were run with a modification of SHVR where it aims to 'shave' a penny off the best quote on its side of the market rather than the best quote remaining on its side of the market after all qualifying orders have been matched.

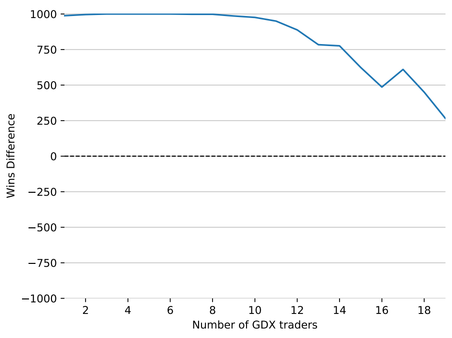

Figure 4.2: GDX vs ZIC Delta Curve for results from a 10 second batch interval

Furthermore ZIC's surprisingly good performance can be seen from Figure 4.2 where when it makes up a small population of the market, it is able compete somewhat closely with GDX. Specifically, when there is only 1 ZIC trader on each side of the market, compared to 19 GDX agents on each side of the market, GDX scores 625 wins compared to ZIC's 375 wins.

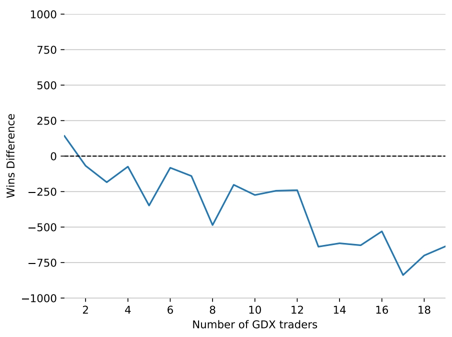

Figure 4.3: GDX vs ZIP Delta Curve for results from a 10 second batch interval

New SHVR Implementation

Trading Algorithm	Total Wins	Algorithms Beaten
GVWY	94330	5
ZIP	62071	4
GDX	56833	3
SHVR	40299	2
AA	22922	1
ZIC	8545	0

Table 4.2: Wins and algorithms beaten by each trading algorithm in a 10 second batch interval FBA with the new SHVR implementation

From the new set of results produced in Table 4.2 it is possible to draw further conclusions about the performance of certain trading algorithms. The high performance of GVWY is notable, considering its straight forward strategy of simply quoting its limit order price into the market. The reasons for its success is further detailed in Section 4.4 but this primarily comes down to a number of reasons. Firstly due to GVWY quoting attractive prices into the market it is able to fulfill a large number of customer orders at a profit, and so generates a larger profit per trader. In addition to this, the significant difference in performance between GVWY and the adaptive algorithms, ZIP, GDX, and AA, is unlikely due to GVWY implementing a more intelligent strategy suited for batch auctions. Instead, it is likely a result of other trading algorithms being unable to perform as profitably when the state of the market is not reported as frequently as in a CDA.

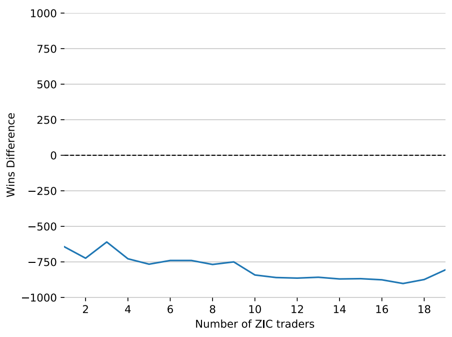

Figure 4.4: ZIC vs SHVR Delta Curve for results from a 10 second batch interval with the new SHVR implementation

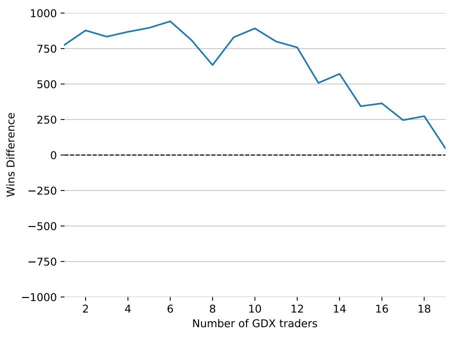

Figure 4.5: GDX vs SHVR Delta Curve for results from a 10 second batch interval with the new SHVR implementation

With the new implementation SHVR produced much more profitable trades. This is seen in Figure 4.4 where SHVR is now able to record more wins at all ratios over ZIC. However it should be mentioned that given a longer batch interval, the auction process of FBAs is able to reduce the effectiveness of SHVR. This is seen in Table 4.2 where SHVR is only dominant over 2 algorithms whereas in TBSE it has been recorded as dominating 3 algorithms when a static equilibrium price is used [9].This is seen in Figure 4.5 where SHVR does not record more wins than GDX at any ratio whereas in TBSE it scores more total wins than GDX across a pairwise comparison.

4.2.2 2 Second Batch Interval

The results from the 10 second batch interval prompted further experiments to see how quickly the FBA dynamics broke down when the batch interval was shortened to more closely mimic a CDA.

Without Varying Equilibrium

The performance of SHVR improves significantly due to the increase in batch frequency resulting from the update of the batch interval from 10 seconds to 2 seconds. This is seen in the hierarchy in Table 4.3 and 4.4 where SHVR scores a significantly higher number of wins and now dominates GDX. This is seen

Trading Algorithm	**Total Wins**	**Algorithms Beaten**
GVWY	91698	5
ZIP	63676	4
SHVR	52080	3
GDX	38439	2
AA	30923	1
ZIC	8184	0

Table 4.3: Wins and algorithms beaten by each trading algorithm in a 2 second batch interval FBA

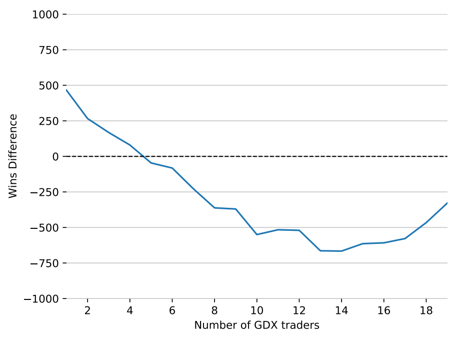

Figure 4.6: GDX vs SHVR Delta Curve for results from a 2 second batch interval

in Figure 4.6 where SHVR now scores more wins at every ratio compared to the 10 second implementation shown in Figure 4.5. Furthermore, the dominance of GDX has shifted such that it only dominates SHVR when it has 4 or less agents on each side of the market; as the number of GDX agents increase, the performance of SHVR increases.

Furthermore the decline in the performance of GDX can be seen in the difference in Figures 4.3 & 4.7 where the delta curve is shifted down as the batch size increases. This represents additional wins of ZIP across all ratios against GDX.

It should also be noted that the increase in performance from ZIC from varying the equilibrium price, as seen in Table 4.4, likely represents ZIC capitalising on the inability of other traders to produce quotes that accurately reflect the market state at a given time, and does not represent ZIC's quotes being more intelligent than other traders.

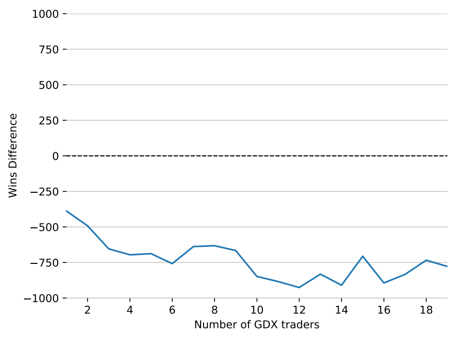

Figure 4.7: GDXvsZIP Delta Curve for results from a 2 second batch interval

With Varying Equilibrium

The delta curves from the implementation of a varying equilibrium price are omitted as they are not dissimilar to those produced with a static equilibrium price. The results from a 2 second batch interval with a varying equilibrium price are shown in Table 4.4.

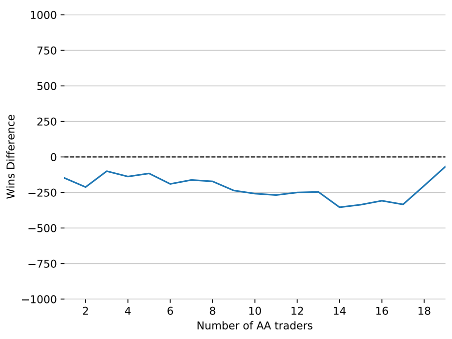

Figure 4.8: AA vs SHVR Delta Curve for results from a 2 second batch interval

Trading Algorithm	Total Wins	Algorithms Beaten
GVWY	92826	5
ZIP	67403	4
SHVR	47353	3
GDX	34908	2
AA	31211	1
ZIC	11299	0

Table 4.4: Wins and algorithms beaten by each trading algorithm in a 2 second batch interval FBA with a varying equilibrium price

4.2.3 Half Second Batch Interval

I could have stopped here but the results of GVVY meant I wanted to explore even less frequent batch period in order to determine if the success of GVWY was still observed in more frequent batches.

Due to the results of GVVY, I decided to investigate even less frequent batch periods. My goal was to determine whether the success of GVWY could still be observed in simulations with more frequent batches, as prescribed in [5].

Without Varying Equilibrium

Trading Algorithm	Total Wins	Algorithms Beaten
GVWY	73306	5
ZIP	65574	4
SHVR	54512	3
GDX	48096	2
AA	35759	1
ZIC	7753	0

Table 4.5: Wins and algorithms beaten by each trading algorithm in half second batch interval FBA

With Varying Equilibrium

The results seen in Table 4.5 and 4.6 show that as the batch interval is shortened GVWY agents are less profitable but still remain top of the hierarchy of traders in BFBSE. A consequence of this is that the number of wins scored by all the adaptive algorithms, AA, ZIP and GDX, increases as they are able to steal more profit from GVWY.

Another notable result is the increase in performance of AA as the batch frequency increases. From Figures 4.8, to 4.9, to 4.10, it is clear to see that AA performs considerably better as the environment becomes closer to that of a CDA. This is more similar to the hierarchy given in [16] where AA is at the top in TBSE with both a static and dynamic equilibrium price.

Another notable result is that in the final experiment, ZIP is able to score more wins than GVWY in ratios where it has many more agents on both sides of the market. This is seen in Figure 4.11 where

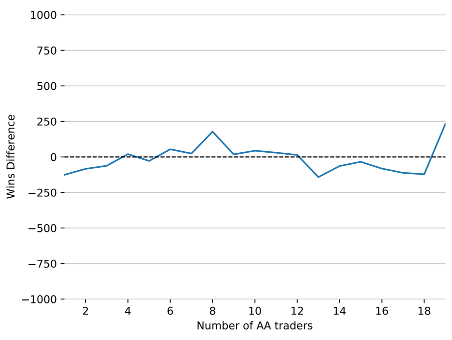

Figure 4.9: AA vs SHVR Delta Curve for results from a half second batch interval

Trading Algorithm	Total Wins	Algorithms Beaten
GVWY	76876	5
ZIP	70482	4
SHVR	49489	2
AA	39176	2
GDX	38967	2
ZIC	10010	0

Table 4.6: Wins and algorithms beaten by each trading algorithm in a half second batch interval FBA with varying equilibrium

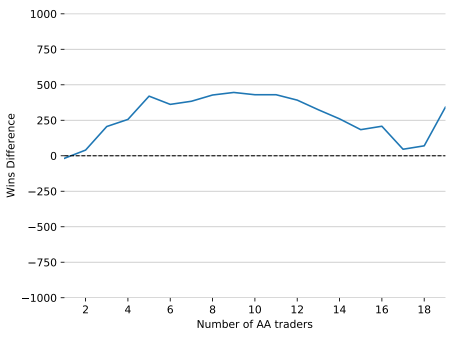

Figure 4.10: AA vs SHVR Delta Curve for results from a half second batch interval with a varying equilibrium price

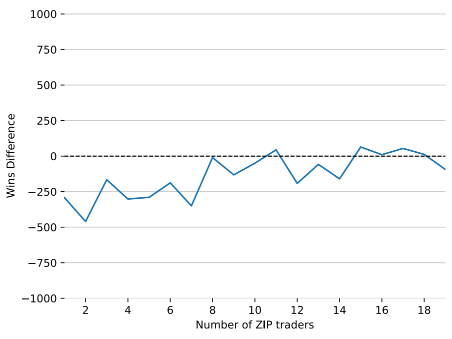

Figure 4.11: ZIP vs GVWY Delta Curve for results from a half second batch interval with a varying equilibrium price

ZIP has a greater number of wins than GVWY when it has between 14 and 18 trades on each side of the market.

4.3 Final Hierarchy and Comparison to TBSE

A final comparison can be performed from the most realistic simulations of a FBA, one in which the batch periods are relatively frequent and the equilibrium price varies. For example, in [5] they propose a trading day that is divided into frequent intervals and suggest around 100 milliseconds for the length between each auction. This is relatively close to the batch interval used in our most frequent FBA experiment, which stands at 0.5 seconds or 500 milliseconds. Although a smaller batch interval could have been used for experiments, 0.5 seconds was chosen as the limit to allow the exchange enough time to perform all necessary calculations for one auction process before starting another. Comparing the performance of trading strategies in a FBA and a CDA is possible by using results produced from TBSE as a benchmark. As discussed in Section 2.5.1, TBSE models a CDA and was the foundation for constructing BFBSE. Consequently, experiments conducted using TBSE can produce insightful findings to analyze the differences between the two auction types.

Algorithm A	Algorithm B	TBSE		BFBSE	
		A wins	B wins	A wins	B wins
ZIC	AA	2178	16822	5601	13399
ZIP	AA	6559	12441	14847	4153
GDX	ZIC	10674	8326	17709	1291
GDX	ZIP	8628	10372	5923	13077
ZIP	ZIC	12452	6548	18984	16
GDX	AA	9401	9599	12032	6968
ZIC	SHVR	4275	14725	841	18159
ZIP	SHVR	5226	13774	10906	8094
AA	SHVR	10637	8363	9379	9621
GDX	SHVR	15073	3927	9258	9742
ZIC	GVWY	4802	14198	4	18996
ZIP	GVWY	5995	13005	7760	11240
AA	GVWY	12296	6704	1860	17140
GDX	GVWY	7708	11292	3174	15826
GVWY	SHVR	7817	11183	10104	8896

Table 4.7: Comparison of Algorithms on TBSE and BFBSE with Static p^*

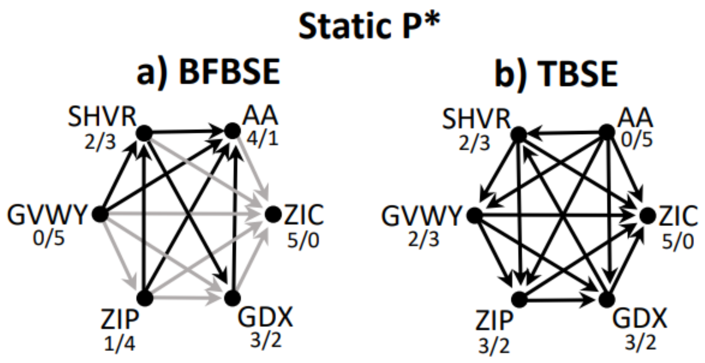

Figure 4.12: Fully-connect pairwise dominance graph for a Static p^*

The results of trading strategies in the most realistic FBA were selected from Section 4.2.3, where a FBA was simulated without a change of equilibrium price and with a change of equilibrium price throughout the trading day. These are referred to as Static p^* and Dynamic p^* respectively. The comparison between the number of wins scored by each algorithm across a pairwise sweep with both a Static p^* and a Dynamic p^* is seen in Tables 4.7 and 4.8 . From these tables, a dominance graph is generated to show the evolution of algorithm dominance when moving from TBSE to BFBSE. This represents the performance of different algorithms by showing arrows from one algorithm to another when it consistently outperforms it in a pairwise comparison. Indegree/outdegree values for each strategy are shown below its name. In the dominance graph for BFBSE simulations, pale grey arrows represent unchanged dominance relationships from simulations run in TBSE, while black arrows represent switches in dominance between algorithm pairs. The dominance graphs from experiments run with both a Static p^* and a Dynamic p^* are shown in Figure 4.12 and 4.14.

The conclusions made from simulations of CDAs performed in TBSE can be simplified into a number of points. Firstly in a CDA, being a fairly simple but quick trading algorithm can be profitable. This is seen in the result of ZIC which is able to dominate GDX across a number of trials when a Dynamic p^* is used. This is simply a consequence of ZIC being able to produce a quote into the market quicker than GDX because of GDX's complexity. With this, ZIC is often able to 'steal' deals from GDX simply by being quick. In an FBA, speed advantages become nullified if the batch interval is longer than the time taken for a trader to produce a quote. This is demonstrated in Figure 4.14, where the transition from TBSE to BFBSE results in ZIC no longer dominating GDX, as it fails to score more total wins than GDX across different ratios. In addition to this, from the experiments conducted in TBSE, it can be seen trading strategies which are not adaptive are less impacted by the noisy LOB in a CDA. In a CDA, orders to an exchange are processed serially and so the state of the LOB when an order is submitted may not have the same state when the order is processed. This is one of the reasons ZIC scores so many of its wins. For example from Table 4.3, we can observe that ZIC is able to score a sizeable number of wins (6548) over ZIP, which is also a fairly speedy algorithm. This partly because of ZIP's reliance on the LOB which will be noisy at some stages when an order is processed. However in BFBSE, ZIC no longer has this advantage over ZIP and so records significantly fewer wins are recorded (16).

From Figures 4.12 and 4.14, it is clear to see GVWY's dominance in BFBSE. Although in TBSE, the simplicity of GVWY enables it to perform profitably, dominating ZIP,GDX and ZIC with both a static and dynamic equilibrium price, it's ranking at the top of the hierarchy for BFBSE is notable due to its simplicity. GVWY's position as a 'source' node, a node with maximum outdegree and minimal indegree, represents the algorithms ability to score more wins across all ratios than all other algorithms it was tested against. This is promising for FBAs as it back up the claims of Buddish et al. which state that FBAs reduce "the value of tiny speed advantages, and the auction transforms competition on speed into competition on price"" [5]. GVWY's profitability lies in its design to always quote the most attractive price it can on the market, which is the limit price assigned from a customer order. By always offering the most competitive price, it consistently outperforms other algorithms.

Another notable change can be observed is the performance of AA vs ZIP with a static equilibrium, where the domination is drastically reversed when the pairwise comparison is done in BFBSE vs TBSE. In TBSE AA dominates ZIP, scoring nearly twice as many wins. However, when BFBSE is used, AA fails to compete with ZIP and scores just 4153 wins. This is seen in Figure 4.13, where AA fails to score more wins than ZIP across any ratio. The possible reasons for this are discussed in Section 4.4.

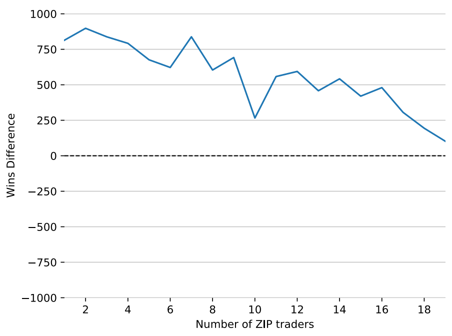

Figure 4.13: ZIP vs AA Delta Curve for a Static p^*

Since academic history of trading algorithms is routed in their performance in CDAs, there are no results we can compare to those generated in BFBSE. However, some of the results recorded in BFBSE are similar to results that have been supported by previous work such as: ZIP beating both ZIC [6], and GDX [16] and GVWY beating GDX [9]. This is promising as it suggests that FBAs can be a viable alternative to CDAs, and that with minimal adaptations, algorithms designed for CDAs can be deployed in FBA environments.

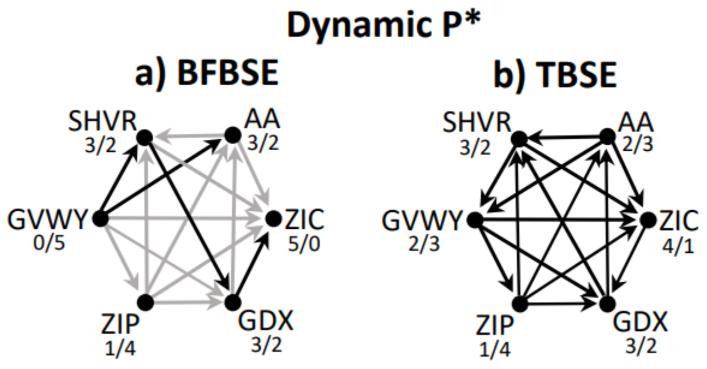

Figure 4.14: Fully-connect pairwise dominance graph for a Dynamic p^*

Algorithm A	Algorithm B	TBSE		BFBSE	
		A wins	B wins	A wins	B wins
ZIC	AA	2724	16276	6375	12625
ZIP	AA	9769	9231	13503	5497
GDX	ZIC	8650	10350	16321	2679
GDX	ZIP	5007	13993	1518	17482
ZIP	ZIC	17025	1975	18965	35
GDX	AA	10303	8697	12147	6853
ZIC	SHVR	4713	14287	918	18082
ZIP	SHVR	10283	8717	12312	6688
AA	SHVR	11779	7221	12105	6895
GDX	SHVR	16375	2625	8312	10688
ZIC	GVWY	5999	13001	3	18997
ZIP	GVWY	9165	9835	8220	10780
AA	GVWY	10227	8773	2096	16904
GDX	GVWY	5462	13538	669	18331
GVWY	SHVR	8430	10570	11864	7136

Table 4.8: Comparison of Algorithms on TBSE and BFBSE with dynamic p^*

4.4 Overview of Performance of Algorithms

From the 1,710,000 trials, generated through simulating 90 pairwise comparisons, a number of judgements can be made about how the 6 algorithms discussed in this paper perform in BFBSE.

4.4.1 Zero Intelligence Constrained (ZIC)

The poor performance of ZIC in BFBSE is unsurprising due to it's main advantages being removed by the structure of a FBA. Although performing well across experiments performed in TBSE, due to its speed advantage over more complex algorithms, this advantage is stripped away due to the discrete time of a FBA which causes orders to be processed at intervals.

ZIC is most profitable when batch intervals are larger as adaptive algorithms have a less clear view of the market and so ZIC is able to profit off the quotes of these traders. However, with this being said, ZIC records the least amount of wins in every experiment performed in BFBSE, bar the first experiment with the old implementation of SHVR. While other non-adaptive algorithms are able to perform more profitably in BFBSE, ZIC's quote prices drawn from a random distribution do not yield a high number of customers orders fulfilled and so it fails to record a significant profit per trader in most trials.

4.4.2 Zero Intelligence Plus (ZIP)

ZIP's machine learning heuristics means that it performs exceptionally compared to other adaptive trading algorithms. ZIP works by adapting its profit margin using a learning rule based off other traders actions in the market. This mechanism causes ZIP to prosper in BFBSE where it is able to consistently quote profitable prices into the market. ZIP's performance with both longer batch intervals and shorter batch intervals are significant as it always remains 2nd in the hierarchy, behind only GVWY. In fact, across all experiments in this paper, ZIP is only ever dominated by GVWY and never scores less wins across a pairwise sweep than any other algorithm. This is unlikely due to GVWY being a more intelligent trading strategy, which is able to accurately capture the future state of a market, but simply a product of GVWY's ability to perform a higher number of trades.

A notable result is that even with more frequent batches, where the market behaviour of an FBA is more similar to a CDA, ZIP is able to dominate algorithms such as AA, which has had papers published supporting it's superior performance to ZIP in a CDA [22],[21]. An interesting avenue for future work might include observing how small the batch interval can be configured, such that ZIP still out performances AA.

4.4.3 Gjerstad Dickhaut eXtended (GDX)

In BFBSE, GDX records similar success to that reported from tests done in TBSE [16]. One of the main motivators for TBSEs development was to re-evaluate the dominance hierarchy of trading algorithms by taking into consideration their response time. This resulted in GDX being placed lower in the hierarchy due to its slower processing time compared to other algorithms.

Given that order matching in an FBA occurs through an auction at discrete time intervals, my initial assumption was that GDX could be one of the most dominant strategies in BFBSE. This is because GDX would have sufficient time to quote intelligent prices to the market, without concern for simpler traders crossing the spread first, by simply being faster. While this assumption is correct for longer batch intervals, where GDX is placed high up the hierarchy for experiments conducted with batch intervals of 10 seconds, it is not the case when a batch interval closer to that prescribed in [5] is used. For example from Table 4.1 , it can be seen that GDX scores very favourably and is able score a similar amount of wins to ZIP, which is the most successful adaptive algorithm in BFBSE. The results presented in Table 4.5 show that as the batch interval is reduced to half a second, GDX scores fewer wins. This effect is further magnified when an offset function is used to introduce variation into the equilibrium price throughout the trading day, as seen in Table 4.5. This is a result of SHVR's number of wins increasing, leading to GDX losing its dominance over SHVR. Consequently, GDX falls in the hierarchy.

It should also be noted that GDXs strong performance with large batch intervals is surprising as it is formulated to make quotes "in a broad class of auctions characterized by sequential bidding and continuous clearing", which does not fit the properties of a FBA [20]. Despite this, it is still able to dominate both ZIC and AA across all batch intervals tested in this paper.

While the result of GDX for smaller batch intervals is disappointing, GDX's belief function could potentially be modified to enable it to function more effectively with the discrete reporting of a FBA. This could be achieved through hyperparameter tuning, which is detailed in Section 5.3.

4.4.4 Adaptive Aggressiveness (AA)

Vytelingum's Adaptive-Aggressive (AA) trading strategy is designed to maximize profits by adapting to the current market conditions and taking aggressive positions when the opportunity arises. In simple simulations of a CDA, AA is widely regarded as one of the best academically established algorithms and has been shown to dominate all other strategies across a variety of market conditions,[22],[21][11]. However, this strategy may not be well-suited for a FBA.

AA's mechanism relies on estimating the current equilibrium price p^* and then using this in order to determine if its limit order price is less than or greater than p^*. In AA, orders are categorized as intramarginal if their transaction at the equilibrium price results in a profit, and as extramarginal if they won't make a profit at the equilibrium price. It is likely that the measure of the equilibrium price used by AA, is not an ideal metric in a FBA.

AA's strategy of classifying orders as intramarginal and extramarginal can work effectively in a CDA. However, in a FBA, where transactions occur at regular intervals, a large batch of orders can skew AA's perception of the true equilibrium price in the market as they all transact at the same time. Because of this, AA may find it difficult to determine the intramarginality or extramarginality of its limit orders due to the poor accuracy of its perceived equilibrium price. In a CDA a poorly calculated equilibrium price can quickly be corrected by the arrival of new quotes to the exchange, however due to the batching nature of FBAs it is much harder to evaluate the true equilibrium price. This can be seen as AA scores fewer wins in markets simulated with a large batch interval.

Moreover, AA's inability to calculate accurate equilibrium prices is exacerbated by the fact that trading volumes in FBA are lower as a result of the batching process. A consequence of this is that it takes a longer amount of time to produce an estimate of the market equilibrium price and so AA's performance suffers.

Another pitfall of AA in FBAs is its inability to transact with orders on the exchange instantly. When AA's aggressiveness value is above 0, AA's passive strategy of quoting prices that generate large profits but are less attractive to counterparties are unlikely to transact. The reason for this is that, unlike in a CDA, where AA can transact with less informed traders as soon as the quote is submitted, the batch order matching of an FBA elimnates AA's ability to make a profit in a volatile market.

4.4.5 Giveway (GVWY)

The results of GVWYs performance in BFBSE represent the most significant shift in the hierarchy from the results reported from studies with TBSE [9]. GVWY dominates all over algorithms, across all experiments run in BFBSE.

In BFBSE, GVWY is at the top of the hierarchy and dominates all other trading algorithms. This may seem impossible since GVWY never aims to make a profit and instead quotes its limit price into the market. In a CDA it can occasionally make a profit by crossing the spread the moment the order is processed by the exchange.

However, in FBAs, transactions for a given batch occur at the clearing price, which is determined as the point at which total supply is equivalent to total demand. For this reason, if a limit price is quoted to the market at a more attractive price than the equilibrium price, it will make a profit. Given the batch nature of processing in FBAs, the occurrence of GVWY agents making a profit is significantly more likely than in TBSE, as in between batch intervals many orders may cross the spread, which will have an effect on the equilibrium price. This is not the case in continuous double auctions in which due to the serial nature of processing, the spread is only crossed by a single bid/offer at a time. Moreover, with the batch processing of FBAs, the state of the market when orders are processed is not known and so other trading algorithms will find it harder to exploit any bids submitted from GVWY that are much more attractive than previous quotes from other traders in the market.

It should be mentioned that part of the reason for GVWYs dominance is that it never tries to make a profit and so executes the most trades compared to any other strategies. A consequence of this is that it can record a high average profit per trader, without recording trades which make a large amount of profit.

When batch intervals are shortened, and so the market becomes more like a CDA, GVWY's scores less wins. This is for a number of reasons. Firstly, with more frequent batches, fewer orders cross the spread at each auction and so the clearing price is more influenced by GVWY quoting its limit price. Another reason for GVWYs worse performance in shorter batch intervals is that the state of the exchange is reported more frequently and so adaptive traders such as AA,ZIP and GDX are able to perform more profitable trades. However, even with experiments run at a batch interval of half a second, GVWY is able to dominate all other algorithms that have been tested against it.

4.4.6 Shaver (SHVR)

SHVR is described in [9] as a “strategy, intended as a tongue-in-cheek model of a contemporary high-frequency trading (HFT) algorithms, which involves no intelligence other than a relentless desire to undercut all of its competitors;“. For this reason, the performance of SHVR in a FBA auction is of great interest to us as it is the trading algorithm that mimics HFT, which is what FBAs claim to address [5]. For this reason, 2 SHVR implementations were trialled in order to examine the dynamics of an FBA.

Firstly, SHVR followed its original implementation by attempting to have the best bid remaining from the supply/demand curves. Since the curves and the LOB are analogous, SHVRs performance from an FBA to a CDA may seem unchanged, but the performance is affected significantly. Since SHVR agents only act on the state of the market at any given time, they rely on this market state accurately reflecting the state of the market when the orders are matched. Since transactions only occur in the auction phase of a given batch period, the state of the market derived from the supply and demand only report the state of the market from the last set of transactions. Because of this, SHVRs ability to gain understanding of the market is significantly stunted and this can be seen from the performance of its first implementation in BFBSE. Results shown in Table 4.1, where it is outperformed by every other algorithm.

While Buddish et al state this difference is similar to the noisy asynchronous information shown on the LOB, whereby the serial nature of orders added to the exchange induce a latency of the actual state of the market being shown on the LOB [5], for longer batch periods the latency in FBA is significantly larger.

This is shown from the results for a shorter batch period, where SHVRs performance is significantly improved and it goes up the hierarchy.

Furthermore, unlike in a CDA, quotes from SHVR are not able to transact immediately due to the batch nature of FBAs. This has a huge affect on the performance of SHVR trading agents as it relinquishes their biggest advantage; they are no longer able to 'steal quotes' from the market at a beneficial price simply by acting the quickest. This advantage is discussed in [15], where the performance of SHVR is impressive. With the FBA mechanism, there is no advantage to being the best bidder in a batch period, so long as your bid qualifies and it transacts at the clearing price and so the first SHVR implementation was adapted.

SHVR's adaption boasted a significant performance increase where it outperforms the adaptive algorithms AA and GDX which use their intelligence, along with a history of previously transacted prices, to submit profitable quotes.

While this may seem to serve as proof that FBA's that have a small batch interval do not mitigate the speed advantages of HFT inspired traders like SHVR, this is likely not the case. SHVR's performance is a consequence of it performing a large number of trades and not necessarily earning the highest profit per trader. Future work might include further analysis into SHVR agents when orders are less frequently replenished and so the profit earned per trade is of more concern to the performance of a sales trader. Moreover, it is likely that with additional hyperparameter tuning, algorithms such as AA and GDX will be able to outperform SHVR agents by using their intelligence to place more profitable quotes in the market.

Finally, since SHVRs number of wins fall as the batch frequency increases, it is likely that a large enough batch period can be chosen to mitigate against SHVRs wins.

Chapter 5

Conclusion

5.1 Summary

This dissertation has explored the emergence of the Frequent Batch Auction (FBA) as a new auction mechanism for financial exchanges. The central goal of this project was explore how the performance of academically established algorithms, which have previously been explored in Continuous Double Auctions, fare when simulated in a FBA.

To address this gap, BFBSE was created which is a first-of-its-kind simulator for FBAs. Through BFBSE, the performance of different trading algorithms were evaluated under the specific constraints of discrete time intervals and batch processing, which are unique to FBAs. Using results generated from BFBSE, a comparison could be made with experiments run in TBSE [9], in order to determine the effect on the published hierarchy.

By conducting more than a million simulations of trading days within a FBA setting, a comprehensive analysis was carried out to examine the dominance and pairwise relationships among various trading algorithms. The results of these simulations led to insight into how trading algorithms designed for CDAs interact with each other, over a pairwise comparisons across 19 different ratios.

Since the design of FBAs is motivated to curb the advantage of high-frequency traders, it was hypothesised that the performance of the SHVR strategy, which is designed "as a tongue-in-cheek model of a contemporary high-frequency trading (HFT)" [9], would decline significantly. Experiments were conducted using a similar interval to that recommended in [5] with the addition of a dynamically varying equilibrium price, in order to better reflect real-world financial markets. The results indicated that SHVR was able to maintain perform profitably in short batch intervals by adjusting the prices it 'shaves' off. Most notably, the results from Section 4.2.3 show that it dominates both AA and GDX.

5.2 Project Status

The status pf the aims and objectives detailed in Chapter 1 are presented below.

1. Although FBAs are a fairly novel mechanism, I performed extensive research into the existing literature. This involved reading research paper and academic journals as well as looking at theoretical analysis used to evaluate their effectiveness.

2. The creation of a first-of-its-kind simulator was successful and allowed the unique dynamics of FBAs to be simulated. This was achieved by extending TBSE [15], which itself is an extension of BSE [8].

3. In order to adapt existing algorithms a great deal of research had to be done into each algorithms decision making so that its implementation could be sufficiently adapted. This was especially difficult for the adaptive algorithms AA,GDX and ZIP and required thorough analysis of their foundational papers.

4. Finally, an extensive analysis was performed on the profitability of individual trading algorithms in FBA. This was achieved using pairwise comparisons as done in [16],[9]. Following this graphical representations such as dominance networks and delta curves could be produced in order to highlight new findings.

5.3 Future Work

In Section 4.2.3, the experiments that closely mimic FBAs yielded surprising results. Despite SHVR being less intelligent than both AA and GDX, the HFT trader model continued to dominate. Algorithms like GDX,AA, and ZIP all quote prices depending on market activity, with the rate at which their bidding behaviour changes determined by hyperparameters such as learning rate and momentum. Experiments in BFBSE were carried out using the default specification of traders in TBSE, hence no conclusions were drawn about the optimal set of hyperparameters. For this reason, if time and budget were not constrained, I would perform hyperparameter tuning on these algorithms and investigate the effect that has on their performance. The optimal set of parameters for a certain algorithm can be identified through hyperparameter tuning by running extensive tests. With the aid of cloud computing this would be possible by running a distinct combination on a rented virtual machine and observing which combination performs best across a set of trials. A result of this may be a significant performance boost for adaptive algorithms as they may be configured operate better with the low amount of trades executed in an FBA.

Furthermore, future work might include utilising cloud technology, as done in [19], in order to capture a wider array of market scenarios and to address the simplifications made in the experiments reported in Chapter 4. By analysing a trading day populated by more than 2 different trading strategies, more accurate conclusions can be made then those produced through pairwise comparisons. The scale of my project did not allow for additional exhaustive testing as it would have necessitated a larger number of tests. Additionally, renting virtual machines to perform the calculations within an acceptable timeframe was not feasible given my budget. Similarly if budget were not constrained, cloud technology could be utilised in order to perform pairwise sweeps, such that the number of traders in a market is not fixed at 20 on each side of the market.

Bibliography

[1] Aws free tier, 2023. URL: https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&amp;all-free-tier.sort-order=asc&amp;awsf.Free+Tier+Types=%2Aall&amp;awsf.Free+Tier+Categories=%2Aall.

[2] Quantlib: A free open-source library for quantitative finance, 2023. URL: https://www.quantlib.org/.

[3] Eric Mark Aldrich and Kristian López Vargas. Experiments in high-frequency trading: Comparing two market institutions, online appendices. *SSRN Electronic Journal*, 2019. doi:10.2139/ssrn.3339584.

[4] Eric Budish, Peter Cramton, and John Shim. Implementation details for frequent batch auctions: Slowing down markets to the blink of an eye. *American Economic Review*, 104(5):418–424, May 2014. doi:10.1257/aer.104.5.418.

[5] Eric Budish, Peter Cramton, and John Shim. The high-frequency trading arms race: Frequent batch auctions as a market design response. *The Quarterly Journal of Economics*, 130(4):1547–1621, 2015. doi:10.1093/qje/qjv027.

[6] Dave Cliff. Zero is not enough: On the lower limit of agent intelligence for continuous double auction markets. 01 1998. URL: http://www-cdr.stanford.edu/~petrie/agents/zero_not_enough.pdf.

[7] Dave Cliff. Exhaustive testing of trader-agents in realistically dynamic continuous double auction markets: Aa does not dominate. In Ana Rocha, Luc Steels, and Jaap van den Herik, editors, *Proceedings of the 11th International Conference on Autonomous Agents and Artificial Intelligence (ICAART 2019)*, volume 2, pages 224–236. SciTePress, March 2019. 11th International Conference on Agents and Artificial Intelligence, ICAART 2019 ; Conference date: 19-02-2019 Through 21-02-2019. doi:10.5220/0007382802240236.

[8] Dave Cliff. Bse: A minimal simulation of a limit-order-book stock exchange, 2023. URL: https://github.com/davecliff/BristolStockExchange.

[9] Dave Cliff and Michael Rollins. Methods matter: A trading agent with no intelligence routinely outperforms ai-based traders, 2020. arXiv:2011.14346.

[10] Rajarshi Das, James Hanson, Jeffrey Kephart, and Gerald Tesauro. Agent-human interactions in the continuous double auction. *IJCAI International Joint Conference on Artificial Intelligence*, 05 2001. URL: http://spider.sci.brooklyn.cuny.edu/~parsons/courses/840-spring-2005/notes/das.pdf.

[11] Marco De Luca and Dave Cliff. Human-agent auction interactions: Adaptive-aggressive agents dominate. pages 178–185, 01 2011. doi:10.5591/978-1-57735-516-8/IJCAI11-041.

[12] Steven Gjerstad and John Dickhaut. Price formation in double auctions. *Games and Economic Behavior*, 22(1):1–29, 1998. URL: https://www.sciencedirect.com/science/article/pii/S0899825697905765, doi:https://doi.org/10.1006/game.1997.0576.

[13] Dhananjay K. Gode and Shyam Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 101(1):119–137, 1993. URL: http://www.jstor.org/stable/2138676.

[14] Georgina Hutton, Ali Shalchi, and Matthew Ward. Financial services: Contribution to the uk economy, Sep 2022. URL: https://commonslibrary.parliament.uk/research-briefings/sn06193/.

[15] Michael Rollins. The threaded bristol stock exchange (tbse), 2021. URL: https://github.com/MichaelRol/Threaded-Bristol-Stock-Exchange.

[16] Michael Rollins and Dave Cliff. Which trading agent is best? using a threaded parallel simulation of a financial market changes the pecking-order, 2020. arXiv:2009.06905.

[17] Annabel Smith. Lseg hits €1 trillion milestone amid five-year turquoise plato anniversary, Sep 2021. URL: https://www.thetradenews.com/lseg-hits-e1-trillion-milestone-amid-five-year-turquoise-plato-anniversary/.

[18] Vernon L. Smith. An experimental study of competitive market behavior. *Journal of Political Economy*, 70(2):111–137, 1962. URL: http://www.jstor.org/stable/1861810.

[19] Daniel Snashall and Dave Cliff. Adaptive-aggressive traders don't dominate. In Jaap van Herik, Ana Paula Rocha, and Luc Steels, editors, *Agents and Artificial Intelligence*, Lecture Notes in Artificial Intelligence, pages 246–269, Germany, December 2019. Springer Berlin Heidelberg. doi:10.1007/978-3-030-37494-5_13.

[20] Gerald Tesauro and Jonathan L. Bredin. Strategic sequential bidding in auctions using dynamic programming. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2*, AAMAS '02, page 591–598, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/544862.544885.

[21] Daniel Vach. Comparison of double auction bidding strategies for automated trading agents. 2015. URL: https://dspace.cuni.cz/bitstream/handle/20.500.11956/77724/DPTX_2013_2_11230_0_415646_0_152184.pdf?sequence=1&isAllowed=y.

[22] Perukrishnen Vytelingum. *The Structure and Behaviour of the Continuous Double Auction*. PhD thesis, 12 2006. URL: http://eprints.soton.ac.uk/id/eprint/263234.

Appendix A

Appendix

Bristol Frequent Batch Stock Exchange is used extensively to perform experiments evaluating the performance of trading algorithms in a Frequent Batch Auction. The code for BFBSE can be found at [redacted]