DEPARTMENT OF COMPUTER SCIENCE

Improving the Strategies of Algorithmic Traders and Investigating Further Realism in their Market Environment

Steven James Alexander Stotter

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

May 2012 | CSMENG-12

Declaration

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. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Steven James Alexander Stotter, May 2012

Executive Summary

I. Abstract

Just over 10 years ago, a seminal paper published by a team of researchers at IBM concluded that two types of autonomous, adaptive algorithmic traders (‘*agents*’) were able to consistently beat human traders in a continuous double auction (CDA) [6]. What made this a major finding is that this type of market mechanism is used in a wide variety of exchanges, including all major financial exchanges, with the team concluding that the impact of these results could “be measured in billions of dollars annually”. Since then, it seems that the anticipated impact of this study has indeed been realised, with algorithmic traders being responsible for over 50% of all equity trades in US markets alone [1]. But in academia, the pace of research in this area has since been typically slow, and it wasn’t until last year that a repeat of this study was completed using a more realistic *continuous replenishment* CDA market environment [14]. This new study, commissioned by the UK Government as part of a study into the future of the financial markets, repeated the aims of the original IBM study, using the current best algorithmic trading strategies developed in academic research. However, that study’s findings were somewhat in contrast to the IBM experiment; finding that the computerised trader-agents were in fact less efficient compared to humans when tested in these more realistic experimental conditions.

In this dissertation, I extend this recent study further by taking two of the agent strategies used, and scrutinise and improve their performance and adaptivity. Due to open interpretation about how they’ve been implemented previously, I have found that the implementation of the strategies used in the latest study were not optimal. By testing them in an agent-only environment, using the more-realistic continuous replenishment CDA market mechanism, and adjusting the speed of the market and the speed of each agent, I accurately determine agent performance. Then, by introducing market shocks, I test these agents’ naive performance in dynamic markets, and provide additional modifications and thoughts of how to further improve their adaptivity when a shock occurs.

II. Summary of Achievements

- In this dissertation, I am the first person to use the ExPo trading platform in research; automating, testing and editing the platform (in Ruby & Bash) to provide stability. Due to this platform still being in development, I had to spend time bug tracking and setting up for appropriate use.
- Converted and implemented versions of both ZIP and AA algorithmic trading strategies (in C), scrutinising their previous implementations in research and improving them further; testing any assumptions made previously and adapting them to work in more realistic market conditions in ExPo.
- Analysed previous research, indicating differences in approaches, explaining the approach I have taken in this paper, and recommendations for future research.
- Replicated a recent important study, finding flaws and introducing fixes, and comparing prior results to new results.
- Introduced market shocks into this new variation of CDA, analysing the ability to reach a natural equilibrium naively, and adapted an agent to react to shocks.
- Conducted over 200 hours of experiments used in this paper, conducting an additional 400 hours in testing.

Contents

Chapter 1

Introduction

1.1 Algorithmic Trading

In 2001, a team of researchers at IBM decided to conduct an experiment to test how effective two adaptive, algorithmic trading strategies (known as ZIP and GD) were against human traders [6]. Previous studies using homogeneous trader populations of all-humans or all-agents (algorithmic trading strategies) had indicated that both populations approached theoretically-perfect efficiencies using the same market conditions being used in the IBM study. The new experiment attracted international coverage as they found that these software agents consistently beat human traders in the market; achieving greater efficiency by making more profitable transactions. The paper concluded by predicting a huge impact on future trading in markets: *"We suspect that, in many real marketplaces, agents of sufficient quality will be developed such that most agents beat most humans. [...] then the competition between agents and humans will evolve into a competition among agents"* [6].

And it seems they may have been right. The rise of this technology usage in financial markets has been exponential in the last decade or so, thanks in-part to the deregulation of markets, transforming the trading floor. Trades that used to change hands between human traders not so long ago are now being mostly fulfilled electronically, at super-human speeds, by software agents. The practice is called High Frequency Trading (HFT), with trading firms striving to increase their strategies by fractions of a second to outperform each other. But how much do we know about how these strategies act in real financial markets? The advancement of this field in commercial deployments has been extraordinarily fast, but coupled with slow academic advancement and outdated experimental conditions, the gap between what is happening in practice and what is understood in academia is growing ever wider. A common thought, now more prominent after the wake up call of the Flash Crash on May 6th 2010, which saw the biggest one-day decline in the Dow Jones Industrial Average ever recorded during a 20 minute period [2], is questioning the sustainability and future of HFT; so much so that it's now become the focus of a new initiative set up by the UK Government.

The *Foresight* Programme, set up in 1994 and headed by the Government's Chief Scientific Adviser, advises the UK Government on future uncertainties, reporting directly to the Prime Minister and Cabinet. One of their current projects, *The Future of Computer Trading in Financial Markets*, is looking at the effect of this mass change in financial markets. This project is not due to conclude until later this year, but supporting research has recently been published. One supporting evidence report produced by this project in particular attempted to repeat the famous IBM study from over 10 years ago, but bringing it up to date with new developments [14]. First, improved adaptive algorithmic trading strategies were chosen to study, and second, a new more realistic 'drip-feed' (or *continuous replenishment*) market property was used. They found that, using this more realistic experimental model of a trading market, closely imitating such exchanges as the equities market, that agents were in fact less efficient than human traders during experiments. They concluded that speed was a big factor in the results, with slow markets hindering agent performance but enhancing human performance.

In this paper, I will study and replicate the continuous replenishment market property proposed in this new paper on an open-source platform currently still in development, called *ExPo* ('Exchange Portal'), which focuses solely on all-agent interaction. I will take two leading adaptive trading strategies which were also used in this new study of agent-human dominance in markets - Cliff's Zero-Intelligence-Plus (ZIP) strategy [4] and Vytelingum's Aggressive-Adaptive (AA) strategy [23] - and examine and improve upon their implementation. As there has been something of open interpretation previously about how these strategies should be implemented in continuous double auctions (CDAs), I will take a closer look at the performance of a couple of implementations. As no humans are participating in these experiments, there are no participants in the market limited by their speed of execution, so the market experiments can afford to be a lot quicker. By altering the speed of the market and the update speed of each agent, I will test the effect of speed on agent performance and dominance.

To further increase realism, I will introduce market shocks in this new continuous replenishment market and research how the different agent strategies perform *naively* (i.e. with no additional adaptation) in dynamic markets. To the best of my knowledge, this has never been studied before. I will then present ideas about how these agents can be adapted to notice and take advantage of market changes, effectively trying to beat the market to provide the least profit spread between buyers and sellers. I will analyse how an adapted strategy works with others in the market, and the effectiveness of adapting to market shocks as a single individual trader.

This research aims to provide a much better insight into how different HFT agents actually operate in real-to-life markets compared to previous research, and provide a concise platform on which to extend work. I will analyse the suitability of ExPo as a future platform for academic research in this field of experimental economics. I will also prepare and contribute to the platform so that it can be immediately used by others in the research field.

1.2 Why Is This Study Important?

A recent report from the Securities and Exchange Commission and Boston Consulting Group reported that trades by software agents in US equity markets reached 56% last year, up 21% in 6 years, and predicted to rise to around 70% by 2015 in both Europe and the US [1]. But with little relevant or current academic research into how these agents interact with each other in real-life markets, we lack true understanding of how markets are operating behind the scenes.

In 1962, Economics Nobel Prize winner Vernon Smith paved the way for experimental studies of competitive market behaviour, re-enacting a simple continuous double auction (CDA) with human traders [18]. His pioneering approach has since given rise to several further experiments into the competitiveness and efficiency of traders in a CDA, with researchers testing both adaptive algorithmic traders and humans in varying experiments. But with all the changes made to the participants in the market, what has remained unchanged throughout is the use of Smith's original market model. Although Smith's model has served behavioural economics well until now, being analytically tractable and practical to implement, we now have the ability to introduce more realism, enabling us to detect any experimental artifacts that were in Smith's original CDA design.

By simulating a more realistic experimental environment for leading adaptive trading strategies to compete in, I have provided a better picture of agents' true dynamics and market-based behaviour in modern day markets. With a significant new adaptive agent strategy published in the last few years - Vytelingum's AA strategy [23] - the question of agent dominance in markets lingers somewhat, as although it has been found to be dominant in previous heterogeneous and homogeneous CDA studies [23] [13] [14], it has never been tested in homogeneous continuous replenishment markets to date. This paper tests variations of two leading trading agent strategies, ZIP and AA, in various realistic market situations to help enhance theoretical understanding and comprehensively determine the dominance hierarchy of agent strategies. Where possible, I will increase the adaptive nature of these algorithms making them less reliant on a particular market and improving their overall performance.

This research is pertinent and timely for researchers in the field of experimental economics and agent-based computational economics, as it expands upon some of the latest research into algorithmic trading. By concentrating on modelling an environment for autonomous agents to compete in, I test high speed, continuous trading in both static and dynamic markets. It will benefit both past and future influencers of the field, as I will be taking existing algorithms and adapting them for use in these new markets, summarising and scrutinising previous implementations. I will then conclude the study with recommendations of future research that can be immediately investigated as an extension to the research that I present here. A strategy for development in this field has been long overdue, and I hope to encourage others to participate in research by providing the groundwork and some basic novel new implementations.

Finally this study also introduces and explores the new open-source Exchange Portal ('*ExPo*') trading platform and its capabilities for research in this area. In this respect, the eventual end users of ExPo will benefit from the development of the platform in this research; including full automation scripts and ready-implementations of a couple of the leading adaptive trading agents. As the platform is still in development under the LSCITS (Large Scale Complex IT Systems) Initiative, the stability testing and bug tracking I have performed to use this platform in my research will prove invaluable to the software developers in charge of its future development. Each stage of the proposed research has never been done before and is now only possible because of the existence of this highly customisable, deployable and suitable test harness.

1.3 Challenges Faced

There is not one central challenge in this piece of work; rather there are multiple challenges in the areas of design, implementation and testing. These include working with a development platform, implementing and scrutinising existing trading algorithms, and further adapting these algorithms to deal with dynamic markets.

The open-source trading platform ExPo has been used to carry out market simulations throughout this paper. ExPo is a unique web-based financial trading platform for conducting agent-human trading experiments, and something that I helped to develop as a front-end developer with a couple of other interns over Summer 2011 during an LSCITS internship. It is still very much in development and has never been used before for research, so adapting, testing and developing the system further was a major technical challenge. To get it working for this research study, I had to build automation scripts, fix any prevalent bugs, and carefully analyse its suitability and stability for this type of research. Having not been involved with back-end development, I needed to become completely familiar with the system to be able to make changes and build automation scripts; made difficult by the lack of documentation at this point in time.

The algorithmic traders implemented in this paper are the Adaptive-Aggressive (AA) strategy [23] and Zero-Intelligence-Plus (ZIP) strategy [4]. ZIP has been around for about a decade and a half, and has been cited in nearly 100 different research papers in this field and other broader fields of AI and economics. The main technical challenge with ZIP is the confusion about how it should be best implemented. This paper will aim to pit two different implementations of ZIP against each other to determine the most optimal version of ZIP. In contrast, AA is a fairly recent strategy, published about 5 years ago. AA is seen as a fairly comprehensive extension of ZIP, which uses a similar learning rule as part of its implementation. Because it is new, AA has only been replicated in a couple of published papers [13] [14]. However, neither paper has scrutinised or adapted the algorithm, other than to transfer it to a continuous replenishment environment, and therefore the AA strategy needs to be treated as new - requiring some thought on optimal implementation. In addition to the technical challenge of implementing optimal versions of these adaptive strategies, I will need to adapt them both to the ExPo platform, deciding when and how to best use event callbacks in the continuous market environment.

The final stage of this research will be based on the solid theoretical grounding gained from the previous stages, but it will be more exploratory and technical in nature. I will adapt one variation of the ZIP algorithm to a dynamic market more effectively, capitalising on market shocks where possible. I will discuss a method of detecting future market movement and produce an example of an adaptive strategy that takes advantage of this. With this strategy integrated into ZIP, I will explore the results of this compared to the naive strategy of just reacting to a market shock.

The entire project will encompass the use of many different programming languages to deal with the different parts of development and testing needed to complete the study. ExPo was written primarily in Ruby on Rails, meaning that I will need to code most of the automation scripts in Ruby and use it when delving into the code of the ExPo platform. I will also need to code some of the automation and stability processes in Bash script. The algorithmic trading strategies used in ExPo will be coded in C, but I will also need to understand C# as I delve into the implementation of these algorithms in OpEx [11] [14] which was written entirely in this language. Finally, to extract useful data from ExPo, I will need to develop a variety of complex SQL queries.

1.4 Summary of Aims and Objectives

1. Discover whether continuous replenishment can change the dominance hierarchy of traders in an agent-only market.
2. Replicate and scrutinise previous experimental results, using the new ExPo platform, to test validity of previous results.
3. Improving the suitability and adaptability of existing agent strategies (AA and ZIP) to real life markets.
4. Use the in-development ExPo trading platform to perform this study of multi-agent performance by building automation scripts, stress-testing it and demonstrating the suitability and usability of ExPo for future algorithmic trading research.
5. Investigate the effect of dynamic markets (market shocks) on both naive and adapted algorithms to further improve realism.
6. To have developed everything needed for other researchers to further extend the research presented here with the greatest of ease, and to provide direction by recommending important extensions for future research.

Chapter 2

Technical Background

2.1 An Overview of the Economics

2.1.1 The Continuous Double Auction

An auction is a mechanism whereby sellers and buyers can come together and agree on a price for the transaction of a common item or service. There are several different types of auction mechanisms in existence, each governed by a different set of procedures to facilitate transactions. Types of auction include examples such as the *Posted Offer* auction as commonly seen in retail situations where an offer is advertised and buyers are simply able to accept or reject the offer, and the *English* auction, where buyers increasingly bid against each other until one winner is left standing, who achieves the deal at that highest price quoted in the market.

For the purposes of this study, I will be focusing on one particular type of auction – the *Continuous Double Auction*, or CDA. Unlike an English Auction for example, which is often chaired by a central *auctioneer*, a CDA allows buyers and sellers to both freely and independently exchange quotes (known as "shouts") at any time in the market, where a transaction takes place when a seller accepts a buyers "bid", or conversely, a buyer accepts a sellers "ask". Although it is possible in these auctions for any seller to accept any buyer's bid, and any buyer to accept any seller's ask, it is in both of their interests to get the best deal possible at any point in time, thus executing a deal with those offering the most competitive shout each time.

The adaptive, autonomous traders I investigate in this paper are not solely used in financial markets; they can also be used in any market that operates as a Continuous Double Auction. Examples of a CDA can be found in the majority of commodity exchanges, the labour market and market-based control systems, such as cloud resource allocation [17]. The reason why so many markets use CDA is due to its decentralised nature, leaving participants to continuously barter with one another, achieving superior performance in terms of robustness and efficiency compared to other auction types, even with a limited number of participants and varying supply and demand configurations [7]. The reason why I have concentrated on explaining agent research with relation to financial markets in this paper is due to their large and increasing presence in financial exchanges, with every major exchange (NYSE, NASDAQ, LSE) operating as a version of a CDA.

2.1.2 Experimental Economics

Vernon Smith, in 1962, set out to explore the dynamics of CDA markets [18] in what became Nobel Prize winning work. By conceptualising a minimally designed market mechanism that fulfilled certain key criteria of typical real-life CDAs of organised markets, such as organised stock and commodity exchanges, Smith looked at the dynamics of these general markets populated by human traders.

Armed with a recruit of students, and splitting the group evenly into a group of buyers and a group of sellers, Smith handed out a single card to each buyer and seller with a single price written on each, known only to that individual. The price on the card for buyers represented the maximum price they were willing to pay for a fictitious commodity, with strict instructions that they could not bid a price higher than what was on their card in the market. They were encouraged to bid lower than this price, regarding any difference between the price on the card and the price achieved in the market as profit. Sellers were given similar instructions for the price on their card, although this price represented the minimum they were willing to sell the fictitious commodity for, restricted from asking a price below this in the market. Again, profit was deemed as the difference between the price they could get in the market and the price on their card. Using the terminology we use throughout the rest of this study, the card is known as an *assignment*, with the price detailed on the card known as that assignment's *limit price.*

The experiment was split up into a number of *trading days*, which would last typically a few minutes. At any point during the trading day, a buyer or seller could put up their hand and announce a quote. This is henceforth known as a *shout*, and when a seller and a buyer agreed on a shout, a transaction (or *deal*) was made. At the end of a trading day, all the stock (sellers assignment cards) and money (buyer assignment cards) was recalled, and reallocated from scratch at the start of each new trading day.

Due to the way the experiment had been purposefully designed, Smith could effectively control the supply and demand schedules in the market as he had complete control over the prices on the cards (the limit prices in the market). A demand schedule represents the number of items demanded by buyers as a function of price, and the supply schedule represents the quantity of items up for sale as a function of price. At a higher price, there will be less quantity demanded in the market but more supply due to willingness to sell, and at a lower price there will be more quantity demanded in the market but less quantity supplied by sellers as less of them want to sell at this lower price [22]. We therefore have a point of intersection between the supply and demand curves, known as the point of equilibrium (q_0, p_0) – a point at which we say the market is *efficient.* To better understand what is meant by efficiency, we can analyse the notion of *Pareto Efficient* allocations – a situation such that no-one can be made better off without someone being made worse off.

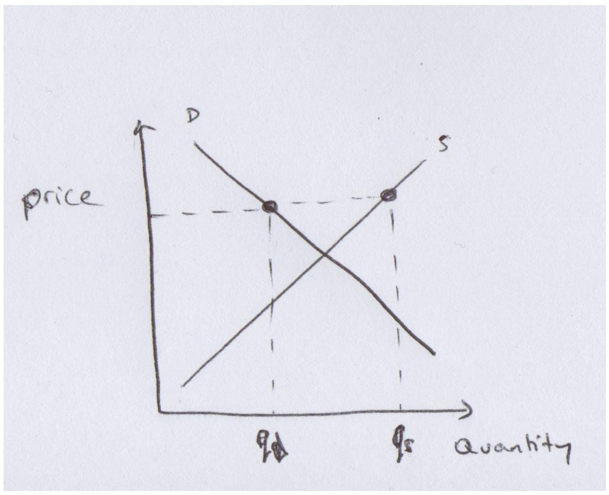


Figure 2.1: Before Equilibrium

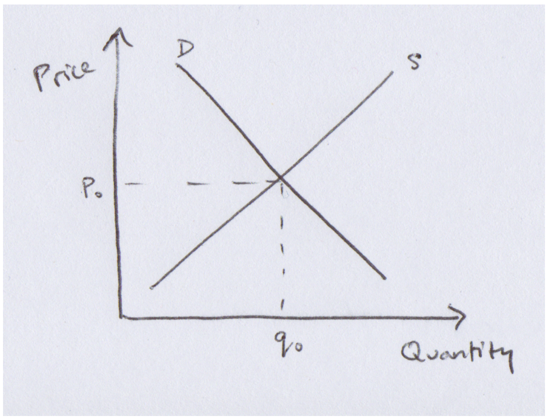


Figure 2.2: Equilibrium

One of the inherent properties of competitive markets, following the notion of Pareto Efficiency, is that they have, *in theory*, the ability to self-equilibrate [22]. At $q_s > q_d$, there is excess supply in the market, with more being supplied than units demanded. Thus, there is more competition between sellers in the market to get their units sold, resulting in a reduction in their prices to attract sales. This increased competition results in q_d rising and q_s falling, resulting in $q_s = q_d$, and more importantly, the market is back to equilibrium point (q_0, p_0). This effect is sometimes referred to as the *Invisible Hand* of competitive markets, but this is a simple theory and in reality markets are a lot more complicated than this example in real life (e.g. due to noise or incorrect equilibration). The research that Smith carried out in his experiments was to establish whether this property of efficiency through market equilibration was realised in his human trader CDA environment.

Smith found that, typically after a couple of 'trading days', human traders achieved very close to 100% allocative efficiency (a measure of the percentage of profit in relation to the maximum theoretical profit available, as detailed in Section 2.1.3). This was a hugely significant result to have been observed, as self-interested, competitive participants were seen to effectively self-equilibrate to the underlying competitive equilibrium point of (q_0, p_0). However, what made this finding even more significant was that the market only consisted of a very small number of inexperienced human traders, and yet still managed to equilibrate.

2.1.3 Market Metrics

The measurements used throughout this study are similar metrics to those used in Smith (1962) [18]. To measure agent efficiencies I have used allocative efficiency, and to measure the market efficiency, I have used Smith's alpha and profit dispersion, as described below.

Firstly, we need to also calculate *Maximum Theoretical Profit* for an agent in order to calculate some of the other metrics. This measurement gives an indication of how much an agent could have made if all profitable assignments n_i provided to them were sold at equilibrium (p_0) prices – profitable meaning theoretical equilibrium below (for buyers) or above (for sellers) an assignment's limit price (l_i).

$$MaxTheoreticalProfit(\pi_{max}) = \sum_{i=1}^{n}(l_i - p_0)$$

(for buyers)

$$MaxTheoreticalProfit(\pi_{max}) = \sum_{i=1}^{n}(p_0 - l_i)$$

(for sellers)

Thus, using this measure of π_{max}, *allocative efficiency* (η) of an agent can be calculated as:

$$AllocativeEfficiency(\eta) = \frac{\pi}{\pi_{max}}$$

This is a measure of how *efficient* the agent was at achieving its maximum theoretical profit. Although a value of $\eta = 1$ is a desired figure, indicating that the agent is achieving perfect efficiency in the market, it is possible to get a result of $\eta > 1$. This happens when an agent is able to exploit weaknesses in their opponents, achieving a trade of value better than that of equilibrium.

It is also important to look at the rate of convergence of prices in the market toward equilibrium. This can be measured using *Smith's alpha*, which is defined as:

$$Smith'sAlpha(\alpha) = \frac{\sqrt{\left(\sum_{i=1}^{n}(p_i - p_0)^2\right)/n}}{p_0}$$

where n is the number of trades in the market, given by $\sum(deals * quantitydealt)$. A small value of α is desired as it indicates a stable market trading close to equilibrium.

One further measurement I will be looking at during this study is *profit dispersion*, which is defined as the cross-sectional root mean squared difference between actual profit and the maximum theoretical profit of individual traders [12].

$$ProfitDispersion(\sigma) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\pi_i - \pi_{max})^2}$$

2.2 Previous Research into Algorithmic Traders

2.2.1 A Brief History of Algorithmic Traders

Anyone with some programming knowledge could technically develop a basic trading agent, albeit maybe not a competitive one. By being programmed to follow certain market-specific trends, better strategies can be and have been developed. But what happens when the dynamics of that market changes? Most successful strategies developed in academia have concentrated on being *adaptive* - meaning that they can be put in any market and learn about their environment quickly to gain maximum efficiency. However, in this section I have looked at a mix of the sophisticated adaptive agents along with a few 'dumb' strategies, to draw comparison.

2.2.1.1 Zero-Intelligence Constrained (ZI-C)

In 1993, Gode & Sunder [9] decided to recreate Smith's landmark experiment, but with an important twist – they wanted to test how intelligent participants in a market would have to be in order to maximise efficiency. One of their objectives was to test 'dumb' agents efficiency against that of human efficiency, to determine whether it was trader intelligence or the organisation of the market that determined efficiency in a CDA.

They tested two algorithms; *ZI-U* (Zero-Intelligence-Unconstrained), which didn't obey to limit price rules specified in Smith's experiment [18] resulting in the ability to make a loss, and *ZI-C* which did conform to the limit price limitation rule. The intelligence of both strategies was, understandably from the strategies' names, non-existent. They used a random pricing rule which would calculate their quotes at any one time based entirely on a random factor.

The study found that the allocative efficiency displayed by dumb ZI-C agents actually ended up being remarkably similar to that of what humans can achieve. They concluded that the reason for price convergence is to do with the emergent behaviour of the market rather than any intelligence of traders. However, Cliff & Bruten [4] later found that the results were actually artifacts of their own experimental design – the study results heavily relied upon the supply and demand chosen in the experiment. By testing the ZI-C agents in 3 alternative markets where the shape of the supply and demand curves differed, ZI-C was shown to consistently fail to reach equilibrium. This confirmed that ZI-C traders do not perform as well as humans in most markets, as they fail to reach the theoretical equilibrium of the market, unlike humans who Smith found consistently converged to theoretical equilibrium in different markets.

2.2.1.2 Zero-Intelligence Plus (ZIP)

After Cliff & Bruten disputed the effectiveness of ZI-C traders, they presented their own agent strategy - *Zero-Intelligence-Plus* (ZIP) traders [4]. These are profit-driven traders that *adapt* to the market based on a simple market learning mechanism, adjusting their profit margins up or down based on the price of other bids and offers in the market, and judging whether they make a transaction or not.

```
Adaptive Rules for ZIP Seller
---------------------------------
 if (deal made)
 {
   if (quotePrice <= shoutPrice) then raiseProfitMargin();
   if ((lastShout == bid) && (quotePrice >= shoutPrice)) then lowerProfitMargin();
 } else {
   if ((lastShout == ask) && (quotePrice >= shoutPrice)) then lowerProfitMargin();
 }

Adaptive Rules for ZIP Buyer
---------------------------------
 if (deal made)
 {
   if (quotePrice >= shoutPrice) then raiseProfitMargin();
   if ((lastShout == ask) && (quotePrice <= shoutPrice)) then lowerProfitMargin();
 } else {
   if ((lastShout == bid) && (quotePrice <= shoutPrice)) then lowerProfitMargin();
 }
```

Listing 2.1: ZIP Trading Strategy Rules

The basic ZIP trading rules are shown in Listing 2.1[1]. When a decision to raise or lower a ZIP trader's profit margin ($\mu_i(t)$) is taken, ZIP modifies the value using market data and an adaptation rule based on the Widrow-Hoff delta ('*learning*') rule:

$$\Delta_i(t) = \beta_i(\tau_i(t) - p_i(t))$$

where β_i is the *learning rate*, and τ_i is the target price, which is set to price of the last shout in the market.

At time t, an update to the profit margin (μ_i) takes the form:

$$\mu_i(t+1) = (p_i(t) + \Gamma_i(t+1))/l_i - 1$$
$$\Gamma_i(t+1) = \gamma_i(t) + (1 = \gamma_i)\Delta_i(t)$$

where $\Gamma_i(t+1)$ is the amount of change on the transition from t to $t+1$, and γ_i is a *momentum coefficient*. Given the limit price (l_i) of the current assignment, ZIP then updates its profit margin ($\mu(t)$) based on these trading rules, where the final quote price (p_i) is given as:

$$p_i = l_i(1 + \mu(t))$$

The ZIP strategy has become a popular benchmark for CDA experiments where algorithmic traders have been studied and evaluated [16] [6] [21] [20] [23]. It was one of the two agents that was used in the IBM experiment 10 years ago [6], which concluded that ZIP was a dominant strategy, beating humans in experimental trials and matching the performance their own GD algorithmic trader (see Section 2.2.1.3).

In recent studies performed by De Luca & Cliff (2011) [13] [12], ZIP has been confirmed again to outperform humans in a CDA environment, although it was found to no longer be the dominating agent strategy, being beaten by an improved version of GD and a relatively new

[1]For a continuous replenishment market, we assume all traders are always active, hence I've omitted the 'active' condition from the trading rules

strategy, AA (see Section 2.2.1.4). It was also recently tested again by De Luca *et al.* as part of the *Foresight* project, and when competing against humans this time, it was seen to be less efficient. This was seen as a surprise finding by the team who investigated this, but after investigation into De Luca's implementation of ZIP [11], the optimality of the implementation of ZIP in these experiments has been called into question. Therefore this is one of the agents that I will look at in this study, attempting to replicate results from this continuous replenishment study to evaluate ZIP's true dominance in real-to-life CDA auctions.

2.2.1.3 Gjersted-Dickhaut (GD)

Developed around the same time as Cliff & Bruten's ZIP trader, yet completely independently, Gjersted & Dickhaut developed a new bidding strategy based on a *belief* function $f(p)$. By taking note of historic trades and shouts that occurred during the last M trades, the algorithm stores this result in H, and the GD agent then builds up a probabilistic model and calculates the chances that a quote at price p will result in a trade.

$$f(p) = \frac{TBL(p) + AL(p)}{TBL(p) + AL(p) + RBG(p)}$$

where:
$TBL(p)$ = number of accepted bids in H where price $\leq p$
$AL(p)$ = number of asks in H with $\leq p$
$RBG(p)$ = number of rejected bids in H where price $\geq p$

GD was one of the two strategies tested by the IBM research team back in 2001 [6], along with ZIP. Since that study, GD now has a couple of new versions studied in literature, including *MGD* which maintains the highest and lowest prices in a trading period to help feed into the belief function, and *GDX* which uses Dynamic Programming (DP) to price orders. As these strategies have been well documented to date, I will not be including these strategies in my research, although it would be valuable to get some comparisons to an optimal version of both ZIP and AA after these experiments.

2.2.1.4 Aggressive-Adaptive (AA)

Developed by Vytelingum in 2006 [23], the *Adaptive-Aggressive* strategy is the first strategy to model 'aggressiveness'. Aggressiveness is defined as a trade-off between profit and the chance of transacting, with an aggressive agent trying to enter competitive bids (or asks) to increase their chance of transacting. On the other hand, a passive agent is actively trying to hold on to achieve more profit even though it has less chance of transacting now.

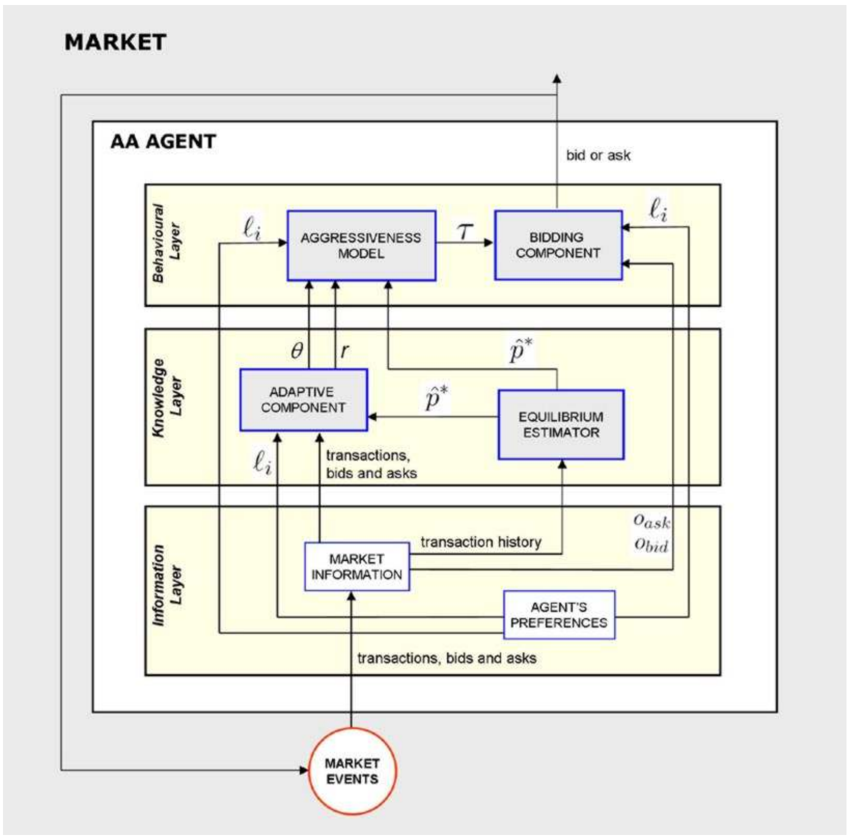


Figure 2.3: The AA bidding strategy, as taken from Vytelingum's paper [23]

To control the level of aggressiveness, a learning rule is used, which actually happens to be the Widrow-Hoff rule again (similar to how ZIP *learns* from the market). Whereas with ZIP we're updating a profit margin value, here we update an aggression value, based on previous market information. Therefore, AA is sometimes referred to as a "more sophisticated version of ZIP".

```
Adaptive Rules for AA Seller
————————————————————————————

 if (deal made)
 {
   if (targetPrice <= shoutPrice) then lowerAggressiveness();
   else then raiseAggressiveness();
 } else {
   if ((lastShout == ask) && (targetPrice >= shoutPrice)) then raiseAggressiveness();
 }

Adaptive Rules for AA Buyer
———————————————————————————

 if (deal made)
 {
   if (targetPrice > shoutPrice) then lowerAggressiveness();
   else then raiseAggressiveness();
 } else {
   if ((lastShout == bid) && (targetPrice >= shoutPrice)) then raiseAggressiveness();
 }
```

Listing 2.2: AA Short-Term Rules

We can observe the similarities of the short term trading rules of AA in Listing 2.2 compared to ZIP (Listing 2.1. However it's hard to get a whole idea of the AA trading strategy from just one component, seeing AA relies on 5 different components (I count long-term and short-term adaptivity as different components).

At any time t, AA can estimate the competitive equilibrium based on a historic window of transaction prices in the market. By introducing a notion of *recency*, the moving average is more sensitive over short periods of time - as to counteract sudden big movements in the market indicating, for example, a market shock. Every time a transaction is made, AA recalculates its estimated equilibrium p^*, and sends it on to a long-term adaptivity component, which deals with updating θ, a property of the aggressiveness model. In this long-term adaptivity component, an internal estimate of Smith's alpha α based on the estimated equilibrium is calculated, and from this an agent can learn, detect and react to foreseen price volatility.

An interesting point to note about AA is that the strategy was developed with dynamic markets in mind. With both short-term and long-term learning, the former is to react to the current state of market information and the latter is to react to any sudden rises and falls – what you the expect the market to be doing right now, or 'one step ahead'.

2.2.1.5 Optimisation Techniques

Optimisation techniques, such as parameter optimisation performed by genetic algorithms, are often used to control parameter choices of agents, normally where a lot of parameters are used. *ZIP60* [3] was developed by Cliff as an extension of his original ZIP algorithm, which saw the parameters of ZIP grow from 8 to 60. By using a genetic algorithm (GA) search to optimise parameters, superior agents can be developed. However, it should be noted that genetic algorithms are used to optimise existing strategies, so I will not be covering them in this extensive investigation.

2.2.2 Experimental Platforms

Previous research in this field has been possible due the building of fairly substantial systems to provide the auction environment. For example, back in 2001, IBM developed a proprietary system called MAGENTA to run their auction experiments to test the dominance of agent strategies and humans [6]. Recently there has been a push towards an open-source solution to allow further research into the field, and it has only been in the last year that we've seen a couple of these platforms come into existence; *OpEx* and *ExPo*.

2.2.2.1 OpEx

OpEx [11] was released last year, after being used previously in several studies by De Luca & Cliff [12] [13] [14]. The system, a mix of hardware and software, consists of 6 netbook computers wired up to a central server, which acts as the *exchange*. Human traders can log into a netbook and take part in a CDA auction, with up to 5 other human participants or a number of algorithmic traders communicating with the exchange.

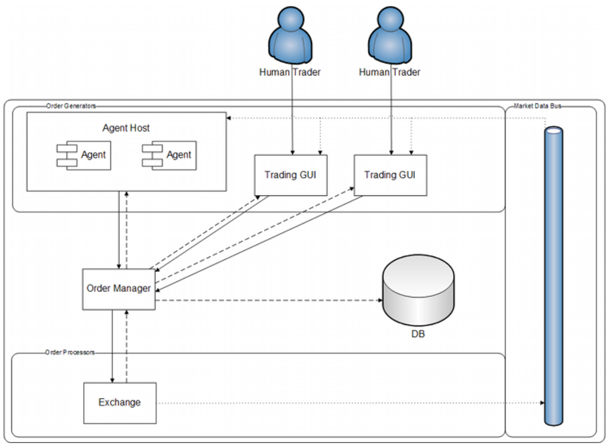
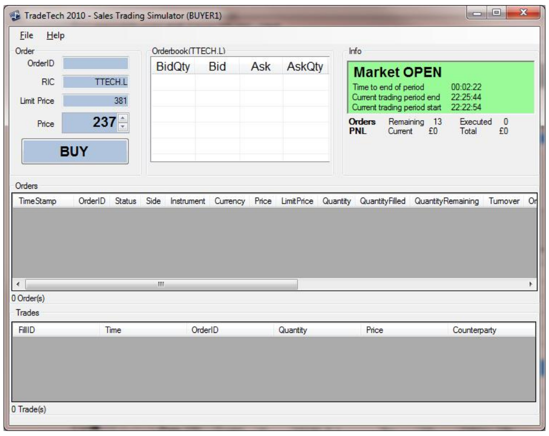


Figure 2.4: An instance of Open Exchange [12]



TradeTech 2010 - Sales Trading Simulator (BUYER1)

File Help

Order

OrderID

RIC TTECH.L

Limit Price 381

Price 237

BUY

Orderbook(TTECH.L)

BidQty	Bid	Ask	AskQty

Info

Market OPEN

Time to end of period 00:02:22
Current trading period end 22:25:44
Current trading period start 22:22:54

Orders Remaining 13 Executed 0
PNL Current £0 Total £0

Orders

TimeStamp	OrderID	Status	Side	Instrument	Currency	Price	LimitPrice	Quantity	QuantityFilled	QuantityRemaining	Turnover	Or

0 Order(s)

Trades

FillID	Time	OrderID	Quantity	Price	Counterparty

0 Trade(s)

Figure 2.5: OpEx GUI screenshot [11]

It was built with flexibility in mind, with the ability to perform more realistic auction experiments that overcome artificial experimental constraints seen in previous research in the field. One of the constraints it was adapted to overcome, the constraint of the *trading day* CDA model, was replaced with a continuous replenishment market in the Foresight study [14]. This is the same market model I use for this research, albeit not on this trading platform.

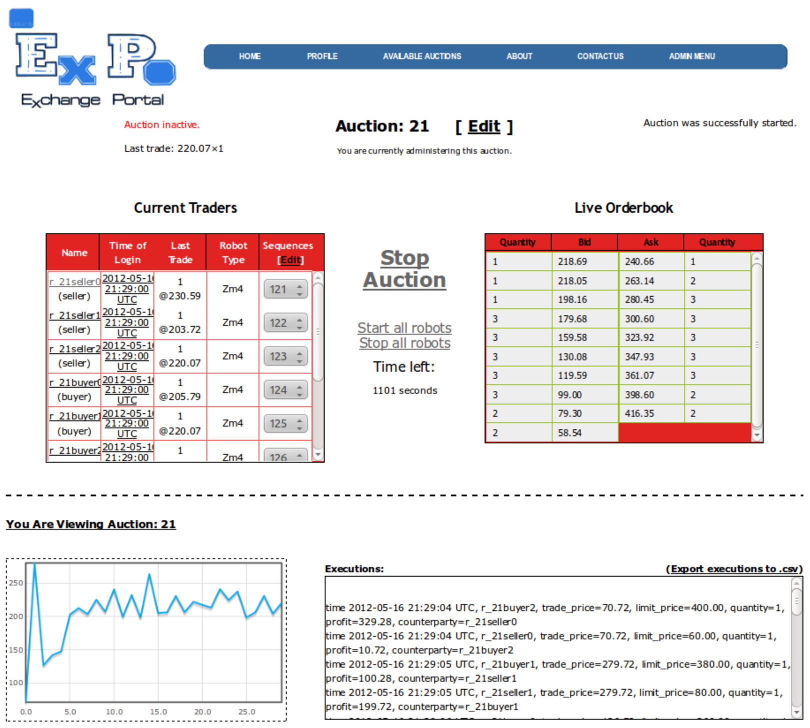
Although OpEx was a step in the right direction for the field of research into the study of experimental and computational economics, the system was still hardware constrained, making adoption rates of the platform understandably low.

2.2.2.2 ExPo

ExPo was built as an alternative to OpEx last year by three interns working for the LSCITS [10] initiative. It was built primarily to replicate an OpEx-like system on the web, with no hardware constraints, and thus little constraint to scale of deployability. It is primarily written in Ruby on Rails, but the platform can compile robots written in C code for running agents in auctions. Like OpEx, a human trader can log on to the system and compete with other humans and against agent traders too. It was built to be as customisable as OpEx, and it utilised a continuous replenishment market too, building on what was previously implemented in OpEx. This made this platform a good choice for running my experiments.

As ExPo was built on a Ruby server, it can be used as a distributed system. That said, it doesn't have to be, with the platform able to work as a local server on a workstation, with everything working centrally. This is a good set up to stress test ExPo, and to run agent vs. agent experiments, but in real-life agents should not sit on the same server as an exchange due to the competitive advantage they get from latency (essentially there is none). Because these agents can work at phenomenal speeds, especially those designed to be operated on real financial exchanges, they would have the advantage of executing profitable trades before anyone else would even get the chance to see them.

I could have run a distributed set-up, using a cluster of machine trading over the ruby web host. However, this would introduce too many possible interferences in results, such as latency and the problem of synchronising automation scripts and robots. I thought these extra interferences had the ability to effect the strength of my conclusions. Also, by performing all experiments centrally, I ensure anyone who would want to further extend my research the ability to do so with minimal constraints to hardware requirements.



Auction inactive.

Last trade: 220.07×1

Auction: 21 [**Edit**]

You are currently administering this auction.

Auction was successfully started.

Current Traders

Name	Time of Login	Last Trade	Robot Type	Sequences [Edit]
r_21seller0 (seller)	2012-05-1 21:29:00 UTC	1 @230.59	Zm4	121
r_21seller1 (seller)	2012-05-1 21:29:00 UTC	1 @203.72	Zm4	122
r_21seller2 (seller)	2012-05-1 21:29:00 UTC	1 @220.07	Zm4	123
r_21buyer0 (buyer)	2012-05-1 21:29:00 UTC	1 @205.79	Zm4	124
r_21buyer1 (buyer)	2012-05-1 21:29:00 UTC	1 @220.07	Zm4	125
r_21buyer2	2012-05-1 21:29:00	1	Zm4	126

Time left:

1101 seconds

Live Orderbook

Quantity	Bid	Ask	Quantity
1	218.69	240.66	1
1	218.05	263.14	2
1	198.16	280.45	3
3	179.68	300.60	3
3	159.58	323.92	3
3	130.08	347.93	3
3	119.59	361.07	3
3	99.00	398.60	2
2	79.30	416.35	2
2	58.54		

You Are Viewing Auction: 21

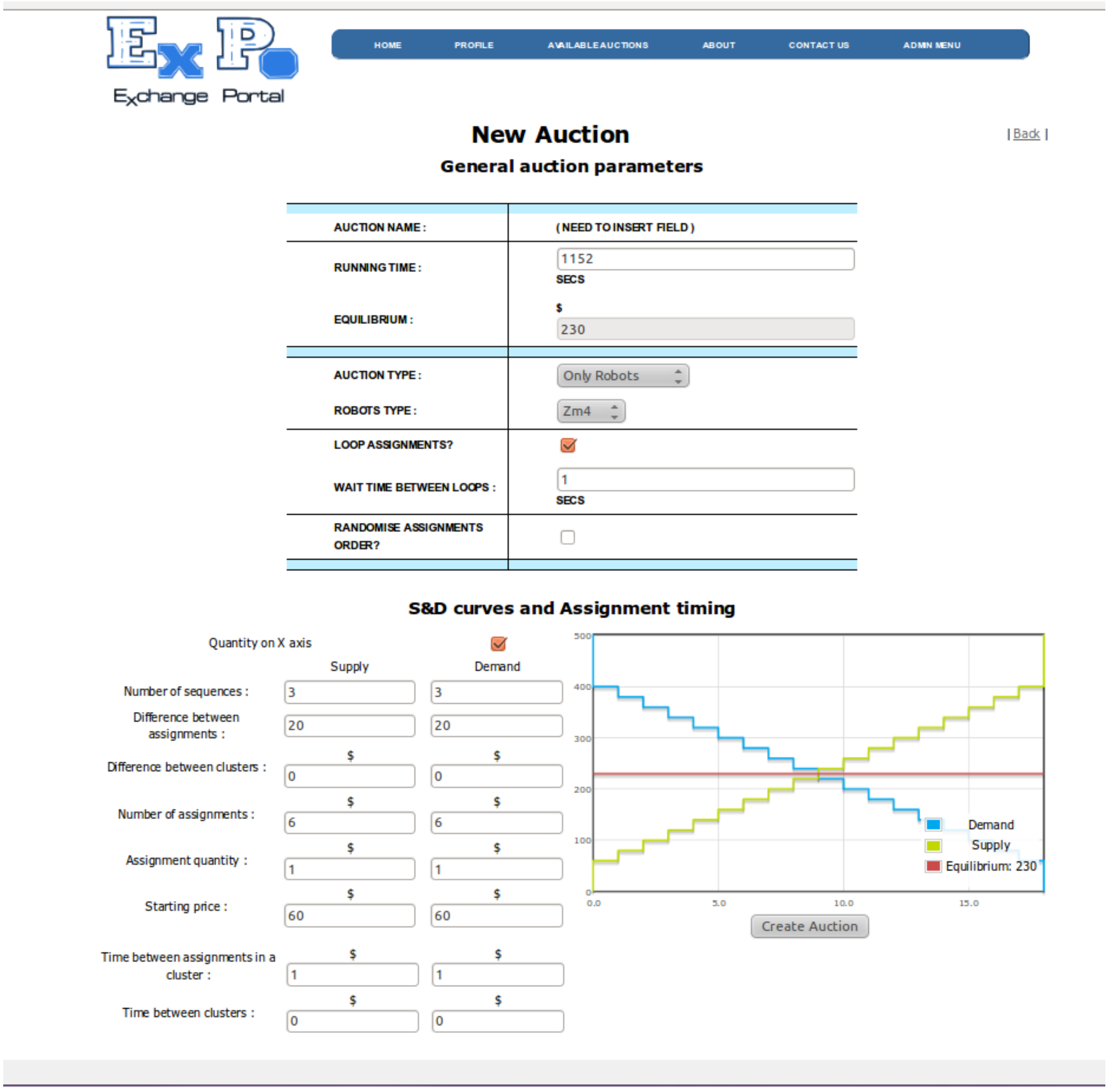
Executions: **(Export executions to .csv)**

```
time 2012-05-16 21:29:04 UTC, r_21buyer2, trade_price=70.72, limit_price=400.00, quantity=1,
profit=329.28, counterparty=r_21seller0
time 2012-05-16 21:29:04 UTC, r_21seller0, trade_price=70.72, limit_price=60.00, quantity=1,
profit=10.72, counterparty=r_21buyer2
time 2012-05-16 21:29:05 UTC, r_21buyer1, trade_price=279.72, limit_price=380.00, quantity=1,
profit=100.28, counterparty=r_21seller1
time 2012-05-16 21:29:05 UTC, r_21seller1, trade_price=279.72, limit_price=80.00, quantity=1,
profit=199.72, counterparty=r_21buyer1
```

Figure 2.6: A screenshot of a working auction in ExPo

It is again an open source platform, like OpEx, but it is still very much in development as a platform. A lot of the technical challenge in this project was to work with this system, tracking down bugs where things went wrong, enhancing stability, creating automating scripts, and writing complex database queries to retrieve important information about the dynamics of completed auctions, agents, trades and shouts. It was also part of the technical challenge was to successfully integrate agent strategies into this new and untried system.

Because of the platform is so new and is not quite finished, this will be the first research paper to introduce ExPo as a platform and as a trading environment. This is why stress testing the system on my hardware was such an important step to take, as it has never been tested in an experimental environment before. I have evaluated the use of this platform and its suitability for previous research. I also plan to release all my work on the ExPo system to the ongoing project, to help in its development and to encourage use of this platform for further work in this area.



New Auction

General auction parameters

AUCTION NAME :	(NEED TO INSERT FIELD)
RUNNING TIME :	1152 SECS
EQUILIBRIUM :	$ 230
AUCTION TYPE :	Only Robots
ROBOTS TYPE :	Zm4
LOOP ASSIGNMENTS?	✓
WAIT TIME BETWEEN LOOPS :	1 SECS
RANDOMISE ASSIGNMENTS ORDER?	☐

S&D curves and Assignment timing

Quantity on X axis ✓

	Supply	Demand
Number of sequences :	3	3
Difference between assignments :	20	20
Difference between clusters :	$ 0	$ 0
Number of assignments :	$ 6	$ 6
Assignment quantity :	$ 1	$ 1
Starting price :	$ 60	$ 60
Time between assignments in a cluster :	$ 1	$ 1
Time between clusters :	$ 0	$ 0

Figure 2.7: A screenshot of the Admin GUI of ExPo

Figure 2.7 above shows the typical set up for an auction through the GUI. The assignment sequences for participants are illustrated by the graph on the right, with the blue line indicating all the demand in the market (i.e. all the demand sequences) and the yellow line indicating all the supply in the market (seller sequences). This is then looped until the end of the auction.

When competitors are added to an auction through the automation scripts I have written, they are put on the same assignment sequences as already exist in the market. This is so that this doesn't cause any artifacts in my experimental design by accidental causing an advantage for a particular group in the auction.

2.3 Research Framework: Problems with Previous Experiments

2.3.1 Implementation of Trading Agents

Due to the fact that a lot of popular agent strategies, such as ZIP, are just that – strategies – a lot of research teams have implemented them differently in their experimental set-ups. This unfortunately causes some discrepancies in how they operate, and thus causes some conflict on how some studies can be evaluated.

2.3.1.1 ZIP Issues

When Cliff [4] originally wrote his ZIP trader, they were programmed to only handle one limit price, they had no explicit notion of time and no persistent orders [6]. So when the IBM team used the algorithm to conduct human vs. agent experimental analysis of CDA markets, they had to adapt the ZIP trader to work with their platform (MAGENTA) in a more realistic auction environment than ZIP was originally designed for. They changed ZIP to be able to handle persistent orders, and implemented an out-bid or under-cut decision (depending on whether the trader is a buyer or seller, respectively) when an order remained open for a certain amount of time without being traded. However, an even more important modification that they made to the ZIP traders was to allow them to have a vector of internal price variables, allowing profit to be made at different values for different assignments. This, they found after implementation, was similar to the implementation that Preist & Tol [16]had independently proposed in a study a couple of years before. Both of these experiments also introduced a 'sleep-time', where if no trade took place within a given time period, they facilitated an automatic competitive price movement - i.e. a price movement towards the best value on the otherside of the orderbook.

But as late as 2006, people were still actively using different versions of ZIP for comparison research. For example, Vytelingum [23] used a strategy for ZIP (and presumably for AA) of updating only the *most profitable* bid (if the agent is a buyer) or ask (if the agent is a seller) at any one time. This implementation seems to have been integrated into De Luca's implementation of ZIP and AA too, as no mention of multiple margins were made anywhere in any of the paper's he's co-authored or any sign of it in his trading agents' code [11] [13] [12].

It is likely that a ZIP trader that holds a profit margin for each limit price is going to be able to maximise their profit more effectively (as they work out best profit for each of their assignments), and this is in fact how I originally wrote the ZIP trader myself. What is interesting is that no one has actually tested out whether a ZIP that can profit maximise on every assignment is better in practice against one which only updates a single profit margin. I therefore make this an aim of my study; to investigate the effects of both strategies.

2.3.1.2 Aggressive-Adaptive (AA) Issues

The AA trading agent is a relatively new algorithmic trading strategy, but has already presented its dominance in CDA auctions against other agent traders [23] [13]. But unlike ZIP, which has been vetted by many different researchers (and as can be seen in Section 2.3.1.1 they may not have colluded yet to one approach for the agent), it seems AA has only been implemented externally by De Luca & Cliff. This means we should treat the strategy as new, and really take time to scrutinise it for being so.

One of the first things that I noticed in the AA strategy was that they can keep orders back from the orderbook if they don't think they'll be able to transact at a profitable price. This is contained in the *bidding component* part of AA's calculations. But dating back to Smith [18], he states it's more desirable to make a limit price trade than no trade at all:

"It is explained that the sellers should be willing to sell at their minimum supply price rather than fail to make a sale"

Preist & Van Tol [16] also state it as a prerequisite for their experiments:

"Each agent is given its own limit price if it is a buyer, it will not buy for over this price, and if it is a seller, it will not sell for less than this. They are free to make any bid/offer subject to this constraint and prefer to make a trade at their limit price than to not trade at all"

One pretty big factor of consideration is that by not putting the order on the orderbook, you could be potentially messing up the perception of demand and supply in the market - as it's not visible to the rest of the market. Therefore I am of the view that an agent should always attempt to trade, even if the best price possible is limit price, as opposed to Vytelingum's paper.

Another attribute to consider is the variable *pMax* and it's implementation in the original Vytelingum paper. *pMax* is defined as the maximum bid or ask price allowed in a market, but this can also be seen as an unfair piece of extra information about the market which could help the strategy avoid trading at prices far from equilibrium. The basis of this assumption is made from looking at De Luca's implementation of *pMax*, who uses it as a determinant for the initial shout price in the market by an AA trader [11] – a case not described in Vytelingum's initial paper. When reading over Vytelingum's paper, it seemed implicit in the AA strategy to force the first move (shout) from another agent to get a better idea of competitive equilibrium, but this is an unfair assumption to make, especially in homogeneous markets of just AA-traders. Therefore, the strategy that De Luca employs of using pMax to determine the initial estimate of competitive equilibrium will need to be used to allow AA traders to make an initial shout, which can be seen in Listing 2.3.

```
public void Reset()
{
        //InitialPriceGuess = 0.7

        double a = _agent.AgentStatus.CurrentInstrument.MinPrice;
        double b = _agent.AgentStatus.CurrentInstrument.MaxPrice;      // pMAX
        double pMin = a + 0.5 * (b - a) * (1.0 - InitialPriceGuess);
        Random r = new Random();
        _estimatedPrice = pMin + r.NextDouble() * (b - a) * InitialPriceGuess;
}
```

Listing 2.3: Equilibrium guess - code extracted from AA agents implemented by De Luca in *OpEx*

This method of calculating the initial guess of equilibrium seems reasonable, giving the initial guess of competitive equilibrium a sensible spread of values to random choose between. But as it uses *pMax* to create the initial guess of equilibrium, it can be seen that different values of pMax may affect trader performance. This is an area which I intend to investigate in Stage 2 of experimentation (Section 4.3.2).

One final issue regarding the AA strategy arises from De Luca's implementation of the agent in OpEx previously [11]. An inclusion of a new *maxSpread* rule can be found in De Luca's code, as seen in Listing 2.4.

```
if (agent.job==JOB_BUY)
{
  if (limitPrice >= bestAsk && (((bestAsk - bestBid) / bestAsk) <= MaxSpread))
  {
    pi = bestAsk;
  } else {
    ...
  }
} else {
  if (limitPrice <= bestBid && (((bestAsk - bestBid) / bestAsk) <= MaxSpread))
  {
    pi = bestBid;
  } else {
    ...
  }
}
return pi;
```

Listing 2.4: maxSpread rule - code extracted from AA agents implemented by De Luca in *OpEx*

This spread condition is slightly worrying, as it's basically stating that a trade should be made when a shout is within a boundary percentage of the equilibrium discovered. The *maxSpread* condition was set to a huge 15% in De Luca's code, meaning trades would be made as far as 15% from equilibrium.

After finding this, I sought to find any previous reference to a spread condition from any source AA implementation by Vytelingum. Although none of his research into AA had any reference to a spread condition, a spread condition did appear in Vytelingum's Risk-Based (RB) agents [24] – an inferior previous version of his AA traders. The value seen here was set to be the absolute value of the minimum indivisible unit of currency (i.e. 0.01); quite far off from the 15% we see in De Luca's code. I believe the only reason it was implemented in the Risk Based strategy agents was to do with price rounding rather than a particular strategy adopted; i.e. the prices throughout the code were being calculated at a greater precision and this was essentially a catch condition for rounding.

Whatever the reason for this implementation of maxSpread, it shall be investigated to explore the effect the condition has on an AA agent.

2.3.2 Real-to-life issues

2.3.2.1 Continuous Replenishment

With realism as a focus of this project, I will be applying the continuous replenishment model; a more realistic experimental model of real-world markets, recently used in the new *Foresight* research [14] to study interactions in human-vs-agent markets. This model allocates orders continuously, not all in one initial block, with the flow of orders trickling into the market gradually. An important stage (Stage 4, Section 4.3.4) of my research will attempt to determine whether continuous replenishment can change dominance hierarchy of traders in an agent-only market, which is something that has never been investigated to date, and something that should have probably initially been included by the 'Foresight' project.

Typical market environments used in previous experiments have followed the ‘trading day’ model of Smith’s original experiments. The problem with this is that it assumes traders only get new assignments at the start of each trading day – typically only one assignment each. Platforms like ExPo help to model markets in a more realistic way compared to previous experiments. By modelling a market as a continuous replenishment auction, we start to model a market in *real time*, allowing assignments to ’drip feed’ into the market, like they would if you were a sales trader on a financial market, receiving assignments from clients.

Cliff & Preist back in 2001 [5] conducted a study using a continuous-market, based on an all-human experiment. It is similar to the design adopted in the recent Foresight paper and that I now adopt in this paper. By performing experiments in this type of market, I seek to explore whether we see any significantly different results from the discreet ‘trading day’ experiments.

It should also be noted that when within such a dynamic and constantly moving market, having an agent property of *active* or *inactive* seems pointless as this inactivity could disadvantage traders in a continuous replenishment market. Therefore, in strategies such as ZIP where it mentions this condition, I had to remove it. There will be a few other important adaptations to make to agents as well to allow them to best operate in these markets, such as what to do when they receive a new or updated assignment.

2.3.2.2 Dynamic Markets

So far, as no-one has previously studied purely algorithmic trading in a continuous replenishment CDA (to the best of my knowledge), no one has looked at what happens to the dynamics of agents and the market when a market shock occurs. I hope to give an insight into this in this paper, by performing a market shock at some point throughout the continuous trading period, and analysing the results.

Dynamic markets are really important to study as all real-life markets are dynamic - i.e. price isn’t assured due to external factors, such as demand or supply side issues.

Chapter 3

Third Party Tools

3.1 ExPo

Throughout this investigation into agent performance in continuous replenishment markets, I used the *ExPo* platform as the agent and auction server. This software was developed by 3 interns under the LSCITS programme [10]. It is currently still in development, and a lot of my work consisted of getting the platform working reliably and automated so that my investigations were possible. The platform is explained in more detail in Section 2.2.2.2.

3.2 MySQL

MySQL was used for all data extraction and for some in-depth statistical analysis too. Several queries were developed in this environment to help with both my research and now any future research to be completed on the ExPo platform by others.

3.3 De Luca's OpEx Trading Agents

A big part of my workload was to decipher how De Luca implemented his versions of algorithmic traders into the OpEx [11] trading platform. I converted parts of his agent code into C and extensively debugged his code to come up with some of the stages of investigation in this paper.

Chapter 4

Project Execution

4.1 Agents

For all agent strategies detailed in this section, they all adhere to the same procedures listed below in this section.

- While agents are in 'sleep', they can continue calculating in the background, they just aren't allowed to place or update an order until they wake up. When they do wake up, the agent has no problem updating or placing a new order competitively, as the quote price(s) is already up to date.
- If an agent receives an assignment with the same limit price as an order they already have on the orderbook, they can group this new assignment to that group of orders, being able to update the group's price in the process. This means an agent can effectively update one group of assignments every time they receive an assignment.
- When findMostProfitable() is called, this finds the most profitable *group* of orders on the orderbook that belong to the agent, such that all limit prices are equal to each other. We assume that in the pseudocode below, an order is always returned.

These adaptations have been made to better suit the continuous replenishment environment that these traders are participating in.

4.1.1 ZIP

In this section I outline the key adaptations that I made to two different versions of ZIP that I used in my experiments.

4.1.1.1 ZIP_S – Cliff's Original ZIP

The implementation of ZIP_S is based on the version of ZIP Vytelingum used in his comparison of AA traders, where only the most profitable order is updated on every wakeup [23].

```
On Wakeup - ZIP_S SELLER
------------------------
 if (bestBid)
 {
   order = findMostProfitable();
   if (bestBid < order->limitPrice) then bestBid = order->limitPrice;
   increaseProfitMargin = (bestBid > lastShout)
   lastShout = bestBid
   changeProfitMargin(increaseProfitMargin, order)
 }

On Wakeup - ZIP_S BUYER
------------------------
 if (bestAsk)
 {
   order = findMostProfitable();
   if (bestAsk > order->limitPrice) then bestAsk = order->limitPrice;
   increaseProfitMargin = (bestAsk < lastShout)
   lastShout = bestAsk
   changeProfitMargin(increaseProfitMargin, order)
 }
```

Listing 4.1: ZIP_S onWakeup()

Essentially, the only thing that is different between ZIP_S and ZIP_M is this wakeup configuration. For all of the parameters, I used the same ones as were used in Cliff & Bruten's original study.

4.1.1.2 ZIP_M – ZIP With Multiple Profit Margins

The implementation of ZIP_M is similar to that of previous experiments [21] [6] [16], although of course there are likely to be discrepancies in individual implementations due to the varying platforms used. Although always updating all of its profit margins for all of its orders behind the scenes, this version of ZIP can update all profit margins it has stored for all orders after every wake.

```
On Wakeup − ZIP_M SELLER
────────────────────────────────
 if (bestBid)
 {
   iterateOrders(order)
   {
     if (bestBid < order->limitPrice) then bestBid = order->limitPrice;
     increaseProfitMargin = (bestBid > lastShout)
     lastShout = bestBid
     changeProfitMargin(increaseProfitMargin, order)
   }
 }

On Wakeup − ZIP_M BUYER
────────────────────────────────
 if (bestAsk)
 {
   iterateOrders(order)
   {
     if (bestAsk > order->limitPrice) then bestAsk = order->limitPrice;
     increaseProfitMargin = (bestAsk < lastShout)
     lastShout = bestAsk
     changeProfitMargin(increaseProfitMargin, order)
   }
 }
```

Listing 4.2: ZIP_M onWakeup()

Every unique limit price received is given a new profit margin set to $\mu_d efault$ and also a new $_g amma$ (i.e. the values of mu and gamma decided at random at the start of running the robot).

4.1.2 AA

Using the same update technique as Vytelingum of only updating the most profitable order [23], every version of AA I test will follow the same method of updating orders on wake:

```
On Wakeup − AA
────────────────────────────────

 order = findMostProfitable();
 updateShortTermAdaptive(_estimatedEquilibrium);
 computeTau(_estimatedEquilibrium);              //using theta, r & price
 price = getBiddingPrice();                      //using tau
 amendOrder(price, order);
```

Listing 4.3: AA onWakeup()

I tested various implementations of AA to get the right implementation for further stages of experiments. These are discussed below, but only have one difference from the original AA algorithm (expressed under the underscore in their name).

4.1.2.1 AA_H, AA_L & AA_D – *pMax* High, Low & Dynamic

Vytelingum never discussed the function for the first shout in the market in his paper [23]. But in De Luca's implementation of AA, he used pMax to randomly guess a price in the market. Unfortunately this gives an even greater and important reliance on pMax, although the concept behind this implementation makes sense - you want to guess somewhere sensible based on the market you're in.

To test the effect of *pMax*, I set one version of AA (*AA_L*) to have a low pMax value of 500, and I set the other version of AA (*AA_H*) to have a high pMax of 2000. I then discuss a dynamic version of pMax, *AA_D*.

4.1.2.2 AA_MS – De Luca's *MaxSpread* Condition

After finding that De Luca had included a *maxSpread* condition in his AA traders used in several papers, including the latest *Foresight* paper, I decided to test this strategy against a standard AA trader without this clause. I have referred to this *maxSpread* trader as AA_MS.

4.2 Experimental Setup

4.2.1 Auction

When a new assignment is provided to an agent, that agent has the ability to put it straight on the orderbook. Although agents can create new orders immediately, each agent can only update their orders once a sleep-time has expired. While the agent is asleep (to be thought of more as '*thinking*'), it is still actively able to calculate new order price using shouts and transactions in the marketplace. Once sleep-time has elapsed, an agent is able to update their order price they've been calculating whilst they were asleep. The ability to put new assignments on the orderbook straight away as they are received is an important difference to previous implementations of sleep-time. An order placed immediately on the book is more advantageous than delaying a trade by waiting.

The sleep-time of each agent was set randomly within a boundary of $\pm(0-25)\%$ of the sleep-time provided. This is the same 'jitter' setting that was implemented by Das *et al.* (2001) [6]. It is used to prevent huge blocks of orders being submitted to the exchange at any one time, which would be an experimental artifact that could alter the system dynamics. Allowing the agents to operate using a stochastic timer will help encourage realism in the experimental setting.

Competitors were added to each auction when and where needed, just before all agents in the auction were started and the auction began. Competitors were added to the same sequences as existing participants so as to not disrupt the market (ExPo doesn't natively support this through the GUI).

Assignment schedules decided upon for the experiments in this research are symmetric between buyers and sellers, and always took the following form:

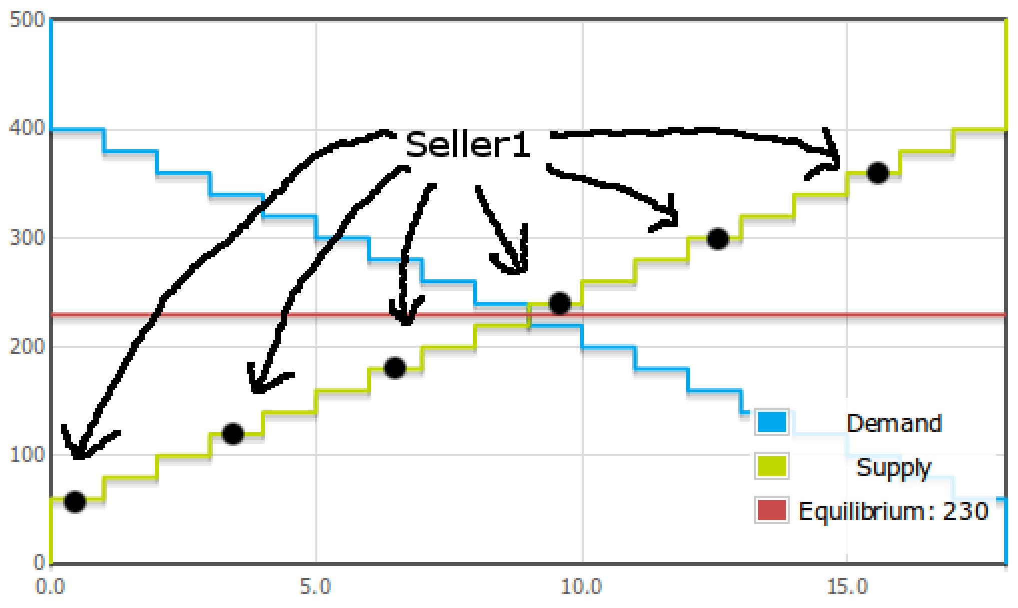


Figure 4.1: Demand and supply assignment schedules in ExPo, highlighting Seller1's assignments

Symmetry was used to ensure that no agents were given an advantage in the market. For example, if we took the demand and supply assignment schedules from the *Foresight* study [14] in Figure 4.2, if a market shock price rise happened after time t, the pricing dynamics between buyers and sellers after the shock would be different to a price fall at time t. This is because different assignments would be left behind in the market at time t, affecting market equilibrium (e.g. more seller assignments than buyer assignments).

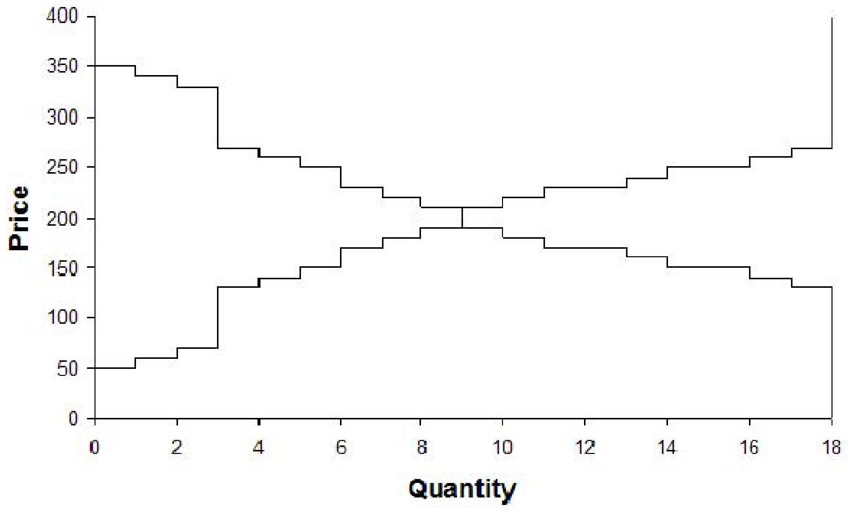


Figure 4.2: Demand and supply assignment schedules used in DR13 - Foresight [14]

For all experiments, the following variables were held constant:

1. The running time for each auction was 1152 seconds. This is because it was similar to the 20 minute length of time that previous human-agent experiments have been conducted in [14], but also because at this value, full trading rounds can be fulfilled at all experimental

settings. For example, when the market is operating at 3 seconds per assignment (as in section 4.3.4), there will be exactly 64 trading rounds, and at 12 seconds per assignment, there will be exactly 16 trading rounds.

2. There are 3 buyers and 3 sellers of each agent strategy in an experiment. This means that in all homogeneous (one strategy only) experiments, 6 agents were competing in the auction in total, and in heterogeneous experiments where two strategies compete, there were 12 agents in the auction.[1]

3. Only one assignment is supplied at a time.

4. The assignment schedules are looped – i.e. *continuously replenished.* As assignments belonging to an agent are grouped by limit price, when an agent receives a new assignment the assignment quantity for that limit price will be incremented.

5. All agents treat current holdings of assignments as a single entity, increasing or decreasing their quote price as a group. However, one or multiple assignments may be traded from a group at any time if only a certain number are able to transact on the orderbook.

6. No retraction of assignments was permitted, and once assignments were distributed, their (limit) prices could not be modified.

7. The NYSE spread-improvement rule limitation does not exist in this environment; buyers and sellers can bid (or ask) what they want in the market, no matter what the current state of the top of the orderbook is.

8. The 'Wait Between Loops' property was set to equal the same as the speed in which assignments went out to all agents across all assignment sequences (labelled 'Time between assignments in a cluster' on the ExPo GUI). I believe this is a bug in ExPo – the time after the last assignment of one loop and the first assignment of the next loop should by default equal the time between assignments within a loop.

9. There were exactly 6 assignments per loop distributed to each agent, with exactly 3 buyer assignment sequences and 3 seller assignment sequences. All assignments arrive sequentially and are exactly 20 apart in price from each other.

10. When a market shock occurs (section 4.3.5), changes to new assignments coming into the market were immediate and their change was directly proportional to the shock. So if a market shock is implemented to change equilibrium price p_0 from 150 to 200, all new assignments coming into the market would be 50 higher in price than they were previously.

4.2.2 Test Environment

I used ExPo to conduct all of the experiments in this research. Each different auction test condition was conducted 5 times, with the averages of metrics and other auction information calculated from these. For analysis of results, I used the non-parametric Robust Rank-Order (RRO) statistical test [8] of these individual 'trials' of each auction, to calculate the significance of these values compared to another set of trials, or between two populations within the same trials.

[1] Comparisons of hetereogeneous markets and homogeneous markets are still possible as we do not compare like for like, but rather observe strategy dominance and significance

Although ExPo can be set up to perform auctions across a distributed cluster of machines, this presents other issues to consider, including:

- Inconsistent affects of latency across machines.
- Individual hardware concerns, as some of the test computers might be faster than others, they may be set up differently, or some may be more prone to failure or crashing.
- Availability, due to the amount (and duration) of tests I would need to run, and the exclusive access of the network I would need.
- It would be a lot harder to keep track of and supervise experiments across multiple computers.

Therefore the hardware that I used to carry out the experiments in this paper was a dedicated virtual machine running *Ubuntu* – under virtualisation through Oracle's *VirtualBox*. This virtual machine had access to 3 CPU cores clocked at 3.4GHz, 8GB of RAM and 250GB of hard drive space. No other user processes but ExPo were being run on the virtual machine during experimentation. This setup was decided to be able to provide plenty of power to conduct these experiments effectively and efficiently, although stress-testing ExPo on this machine was a vital first step to establish this and the best auction and agent parameters.

4.3 Stages of Investigation

4.3.1 Stage 1: Agent Setup & ExPo Evaluation

It is necessary, with a new untested platform, to test its suitability and stability before running experiments. Therefore, Stage 1 of experiments consisted of experiments to optimise the agents' sleep-time for the market, as well as stress-testing ExPo with increasing numbers of agents on my dedicated machine.

Firstly, as ExPo is still very much in development, investigation into which revision to use was an essential first step for my research. A mandatory condition for this research is that ExPo is at a *stable* revision; a revision where all features implemented to date just work, and has not got any half-implemented features or features which work in an inconsistent or irregular manner. From looking into the logs, and back tracking through code revisions, it seemed revision 145 was the safest to use for my implementation. Although two revisions behind the current status of the project, it contained all the necessary features, agent callbacks and fixes that I needed. The later two revisions focused on creating an orderbook depth feature, but this seemed very much work-in-progress, as some bugs had been noted in the logs and I confirmed these to be true.

The next important task in evaluating the use of ExPo for this research was to use it extensively in pre-tests. During this process, the problems I encountered with the ExPo platform included:

- Memory leaks (agents sometimes would continue to run even if told them to stop)
- Excessive log writing (hundreds of megabytes per trader's log at times)

- Assignment server crashes
- Poll server crashes
- Messy exits from the system, leaving behind open sockets and 'stuck' auctions
- An auction sometimes didn't seem guaranteed to stop exactly on time, which may have been down to a lag on the Ruby web server, although this may have also been a Javascript problem in the web GUI.
- The platform required a lot of power when running all robots and auction on the same machine. To get round this, I ended up giving a dedicated virtual machine 3 out of my 4 processor cores for stability and 8GBs of RAM.

Before attempting Stage 1 of experiments, due to the findings above, I built some stability fixes into the automation scripts I wrote for automatically running my experiments with minimal supervision.

The automation scripts I wrote were adapted to:

- Check for 'stuck' auctions and kill them as necessary
- Kill all robots automatically as ExPo has trouble doing this natively at times (and this causes a massive drain on CPU if agents are left to be active).
- Stop and restart ExPo after each auction
- Added an automatic catch to the auctions, to make sure they stopped on time. This prevented any over-run caused by any lag in the Ruby web server.
- Time gap from starting the agents to starting the auction, to allow robots to enter and initialise variables, and account for any Ruby server lag.
- Add agents of any preloaded strategy on the same assignment delivery sequences as existing agents set up in the market. By keeping competitors on the same playing field as other participants, it ensures that I had the ability to draw conclusive results from experiments as I was able to ensure that no strategy had a particular advantage in the market due to experimental conditions.

The assignment schedules used for this stage are detailed in Table 4.1, with each price going to each corresponding agent at the time in brackets.

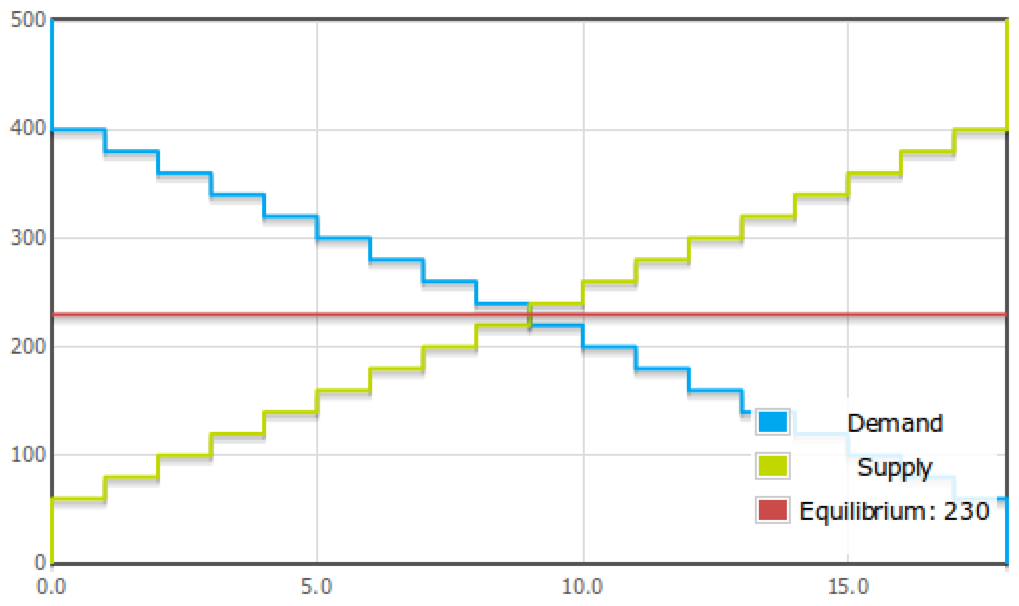


Figure 4.3: Supply and Demand curves, equilibrium = 230

Table 4.1: Assignment Schedules (Stage 1)

	Assignments: Price (Time)					
seller 1	60 (0)	120 (3)	180 (6)	240 (9)	300 (12)	360 (15)
seller 2	80 (1)	140 (4)	200 (7)	260 (10)	320 (13)	380 (16)
seller 3	100 (2)	160 (5)	220 (8)	280 (11)	340 (14)	400 (17)
buyer 3	360 (2)	300 (5)	240 (8)	180 (11)	120 (14)	60 (17)
buyer 2	380 (1)	320 (4)	260 (7)	200 (10)	140 (13)	80 (16)
buyer 1	400 (0)	340 (3)	280 (6)	220 (9)	160 (12)	100 (15)

4.3.1.1 A. Agent Sleep-Time

Adjusting the speed of participating algorithmic trading agents in auctions is an area which has been investigated before in markets where humans were present along with these agents [6] [14]. I performed this stage's experiments to investigate the effect it has on efficiency in robot vs. robot markets.

It is of course, in the nature of HFT, imperative to be the fastest in the market comparatively to other agents agents who can update their orders fastest in tune with the market will succeed in fulfilling more orders at profit. For these agents, it is all about profit and executing profitable trades first in the market rather than how efficient each trade is compared to market equilibrium.

However, this can be an extreme strain on any system, including systems exponentially more powerful than the one I'm using. From a practical point of view, using the described test setup I used to perform these research experiments, if an agent has no inactivity timer (sleep-time) and updates orders on every single shout in the market, there would be an extreme bottleneck for the agent. If the agent can't distinguish what order it has just updated, this actually amounts to an infinite bottleneck of the agent updating its orders every time it updates its orders.

The reason why there has to be a sleep (or rather, *inactivity*) timer is therefore a stability control for ExPo, which was tested in this stage of experiments – I needed to stress-test ExPo in respect to how quick orders could be submitted by agents. However, it is an interesting area to examine, and during these tests, I analyse the difference between agent strategies with differing sleep-times.

In this stage, I focused purely on agent-homogeneous auctions (i.e. buyers and sellers of a single strategy competing against each other). The AA trader used at this stage was AA_L, as this stage is just a test of the stability of the market. All *pMax* versions of the AA trader would have had very similar number of shouts in the market, as the bidding strategy and sleep-times would have all been identical between the different versions.

Table 4.2: Performance Metrics of all Agents and Sleep-Times

Strategy	**Sleep Time**	**Efficiency**	**Smith's Alpha**	**Profit Disp.**	**Total Shouts**	**Total Trades**	**% Failed Assignments**
ZIP_S	1 sec	0.98629463	0.061113769	465.35	8859	606	0.00000%
	2 secs	0.98199703	0.062780388	586.06	5892	579	0.00000%
	4 secs	0.97440323	0.066426249	678.57	4245	580	0.00000%
	8 secs	0.94615610	0.086668991	1962.70	3301	597	0.00000%
ZIP_M	1 sec	0.99376070	0.054667232	361.09	17481	602	0.00868%
	2 secs	0.99564797	0.050893127	362.90	11855	593	0.00000%
	4 secs	0.99471400	0.052859606	308.59	7479	591	0.00000%
	8 secs	0.99663150	0.049169642	472.76	5258	588	0.00000%
AA_L	1 sec	1.00008230	0.021308169	105.24	8845	577	0.00000%
	2 secs	0.99993417	0.017484836	89.64	5753	577	0.00000%
	4 secs	0.99957793	0.016177165	96.97	4023	576	0.00000%
	8 secs	0.99803647	0.018207416	153.86	3199	580	0.00000%

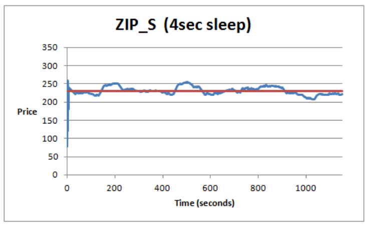


Figure 4.4: ZIP_S

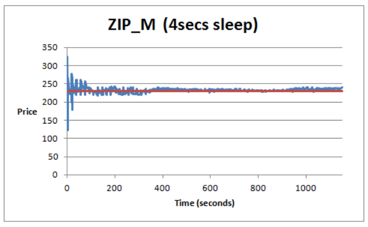


Figure 4.5: ZIP_M

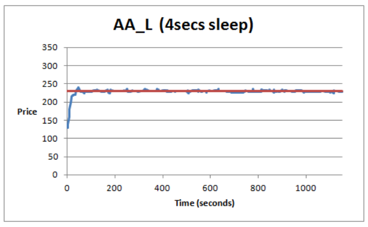


Figure 4.6: AA_L

As can be seen from the results in Table 4.2, ExPo handled all sleep-times very well in this homogeneous 3 vs. 3 environment, with only 1 assignment (0.00868%) failing to be put on to the orderbook throughout all of the experiments. This singular failed assignment was incurred by the ZIP_M trader, and occurred when their sleep-time was set to 1 second. ZIP_M is the strategy that updates all of it's current orders on the orderbook every sleep-cycle, and this can be seen by the amount of shouts made by these agents (which were around double that of the comparison strategies). By far this agent is the most demanding, so it is no surprise that given a failure, it is incurred by this agent.

The graphs of Figures 4.4-4.6 show the dynamics of each agent in a homogeneous auction. Visually, one can see that AA_L converges very quickly and ZIP_S doesn't converge as tightly as the other two. This is also expressed in Table 4.2. Another observation is that ZIP_M converges very well from around 350 seconds, but before then is a little erratic. We would expect Smith's alpha to be higher at the beginning of the auction compared to the other two strategies as it looks like it has poorer converge to equilibrium.

Using the data gathered from this stage of experiments, I used Robust-Rank Order [8] to calculate the significance level of the results. This was to determine whether an agent's speed of updating assignments is a significant factor in a continuous replenishment market.

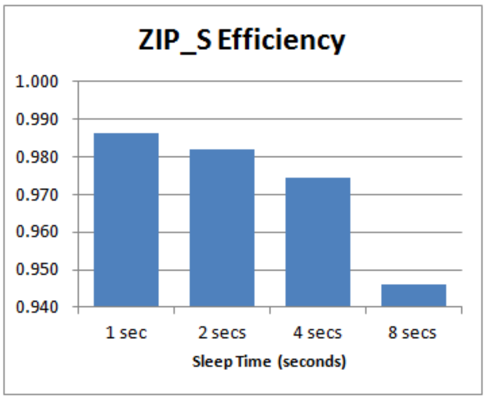


Figure 4.7: ZIP_S Efficiency by Sleep-Time

ZIP_S		
Sleep-Time Efficiency Comparison	**Critical Value**	**Significance**
1 sec vs. 2 secs	1.44766	$(0.048 < p < 0.103)$
2 secs vs. 4 secs	4.20336	$(0.008 < p < 0.028)$
4 secs vs. 8 secs	2.85916	$(0.008 < p < 0.028)$
1 sec vs. 8 secs	Infinity	$(p <= 0.004)$

Figure 4.8: ZIP_S RRO Sleep-Efficiency Significance

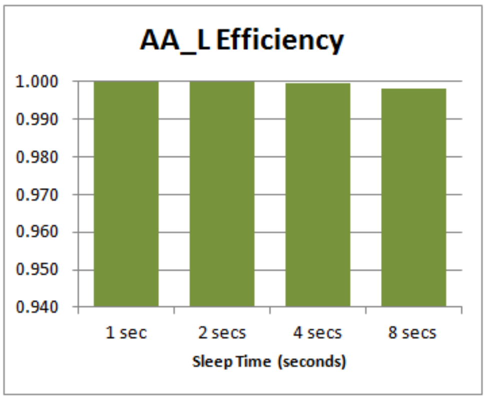


Figure 4.9: AA_L Efficiency by Sleep-Time

AA_L		
Sleep-Time Efficiency Comparison	**Critical Value**	**Significance**
1 sec vs. 2 secs	1.32241	not significant
2 secs vs. 4 secs	2.85916	$(0.008 < p < 0.028)$
4 secs vs. 8 secs	2.85916	$(0.008 < p < 0.028)$
1 sec vs. 8 secs	Infinity	$(p <= 0.004)$

Figure 4.10: AA_L RRO Sleep-Efficiency Significance

From analysing the efficiencies of both ZIP_S and AA_D in more detail, we can see that slowing these two strategies down decreased their efficiency in the market. We can also see from Table 4.2 that Smith's Alpha (α) increased as agents were slowed down too, indicating that the market became less stable (less convergence to equilibrium) when these agents acted slower. This is actually in contrast to the suggested findings in the recent *Foresight* paper [14], which used similar implementations of ZIP and AA. That particular paper found that slowing the agents down decreased the value of α, resulting in more stable markets. However, they measured α in heterogeneous markets, where agents traded against humans. A possible explanation for this finding is that a market with humans acts slower, so agents benefited from acting at a more similar pace.

However, the same finding was not true with ZIP_M, as can be seen below in Figure 4.11.

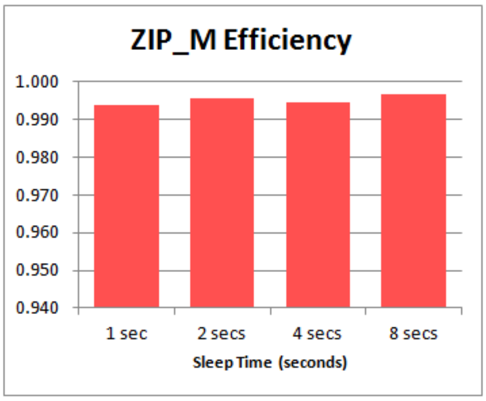


Figure 4.11: AA_L Efficiency by Sleep-Time

ZIP_M		
Sleep-Time Efficiency Comparison	**Critical Value**	**Significance**
1 sec vs. 2 secs	-1.76777	$(0.048 < p < 0.103)$
2 secs vs. 4 secs	0.92164	not significant
4 secs vs. 8 secs	-4.20336	$(0.008 < p < 0.028)$
1 sec vs. 8 secs	-2.85916	$(0.008 < p < 0.028)$

Figure 4.12: ZIP_M RRO Sleep-Efficiency Significance

We can observe from Table 4.12 that ZIP_M has a negative significance value for all significant comparisons. This actually suggests that the second group is significantly higher than the first group - e.g. in the *1 sec vs. 8 secs* experiments, agents updating every 8 seconds have a significantly higher average efficiency than that of those operating at 1 second). This is a similar finding to that of the original IBM experiment [6], as they made similar modifications to ZIP and found that slower agents acted more efficiently, although they didn't make as many trades as when acting faster. It should be noted, however, that their implementation of sleep was slightly different to the implementation used here (see Section 2.3).

Given that nothing else is notably different between ZIP_S and ZIP_M, it is suggested that it is the internal strategy (or *protocol*) of updating orders that seems to be affecting agent efficiency when operating at different update speeds. At a sleep time of one second, these ZIP_M traders were updating about 4-6 prices a second (depending on what they have on the orderbook). That's a lot of price movement in a very small amount of time, in a market full of these traders. It may be the case that these frequent price alterations cause less efficient deals to be made at times, as true value is lost in *noise*.

4.3.1.2 B. Number of Agents (per assignment sequence)

In the previous stage of testing, I concentrated on purely homogeneous auctions, where 3 buyers would trade against 3 sellers. Although I only needed ExPo to operate robustly for a maximum of 12 agents in a market at any one time (two strategies competing against each other in heteroegeous tests), stress-testing the system is important to evaluate ExPo's stability as a development platform. It's also important for the development of robust automation scripts; aimed to enhance stability and make sure that even irregular problems do not occur.

I used the agent strategy ZIP_M for this stage, in a homogeneous environment, as we could see from the previous sleep-time experiment that this agent is clearly the most demanding in the market. I added agents to the auctions as competitors, as they would be added in hetereogenous tests, with up to 3 lots of competitors being added (bringing the total amount of ZIP_M agents in the market to 24).

Table 4.3: ExPo Stability

Strategy	**Number of Agents**	**Sleep Time**	**Efficiency**	**Total Shouts**	**% Failed Assignments**
ZIP_M	12	1 sec	0.99064613	19923	0.73351%
		2 secs	0.99684928	17428	0.07813%
		4 secs	0.99714770	14351	0.00434%
		8 secs	0.99713698	10217	0.00000%
	18	1 sec	0.73431413	20522	9.86400%
		2 secs	0.60982801	18765	15.50926%
		4 secs	0.61601374	15796	18.35648%
		8 secs	0.85197622	11214	4.37211%
	24	1 sec	0.19709643	20515	38.97786%
		2 secs	0.22224838	18564	31.85113%
		4 secs	0.18138427	14762	34.98915%
		8 secs	0.18290611	12248	28.37457%

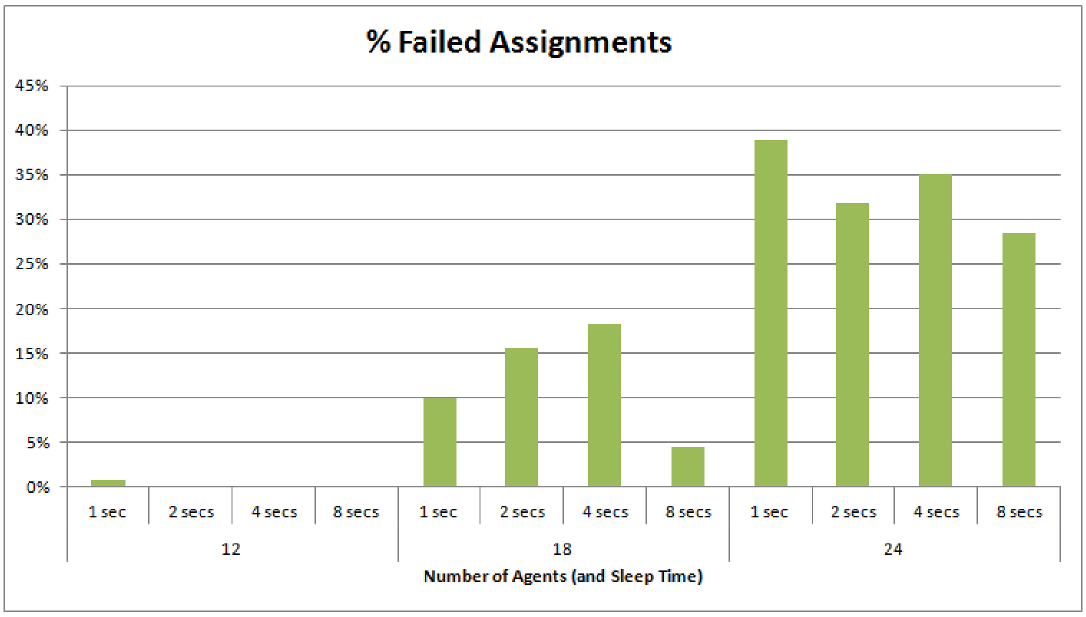


Figure 4.13: Percentage of assignments that were never put on the orderbook

It can be seen from these experiments that ExPo becomes quite significantly less stable as more agents compete in the auctions, especially for 18 and 24 trading agents in the market. Although there are many reasons to why this instability occurs, there are a couple of particular observations to note at this stage.

ExPo could be trying to amend an order, by adding another assignment to the group on the orderbook, which doesn't exist any more (it's been traded). This condition would have to occur in an incredibly small time-frame, as the agents work out whether to amend or make a new order immediately before they send the command to ExPo. There is no error checking in ExPo for whether an amendment succeeds or not, so the robot is unable to tell whether the new assignment has been put on the orderbook successfully or not. In a crowded market with several agents, this is a possible scenario, made more likely by the drain on the CPU caused by all the activity of the auction and participants, causing each agent to go slower.

The demand on the CPU is too much with too many agents. No matter how much sleep-time these agents have, they are continually operating in the background, ready for new assignments to come through, or calculating their bid/offer prices as shouts and transactions occur. This causes agents to crash out of the market (presumed to be inactive as they get stuck), only to resume and enter back in to the market. This is best illustrated by querying the last login time of the agent. This attribute should be identical between all agents as they only login once - before the start of the auction. However, it can be noted that as the number of agents increases beyond 12 in the market (6 vs. 6), there were varying last login times between some agents, indicating they had crashed somewhere and had to restart. When they get stuck, they miss incoming assignments. The percentages of agents having crashed in the auctions are noted in Table 4.4.

Table 4.4: ExPo Crashes

Strategy	**Number of Agents**	**Sleep Time**	**% Agents Crashed**
ZIP_M	12	1 sec	-
		2 secs	-
		4 secs	-
		8 secs	-
	18	1 sec	46.67%
		2 secs	70.00%
		4 secs	57.78%
		8 secs	33.33%
	24	1 sec	89.17%
		2 secs	99.17%
		4 secs	98.33%
		8 secs	97.50%

Essentially, both of these things could be a problem for the scalability of ExPo, albeit scaled greater than what I require for my experimentation. More than anything, I'm interested in the

results of the 6 buyers vs. 6 sellers experiments, as I will be using these quite frequently to test strategies against one another.
The key findings and conclusions were:

- 0.73% of all assignments (169 of 23040) don't go on the orderbook when ZIP_ has a sleep-time of 1 second in a 6 vs. 6 market. This effectively rules out using a sleep-time of 1 second, even though on the surface the agents were trading well and efficiently.
- This decreases to 0.07% when operating at a 2 sec sleep-time (18 assignments).
- This decreases again to 0.008% when operating at a 4 sec sleep-time (just 1 assignment in 23040).
- No assignments failed to be put on the orderbook when operating at 8 sec sleep-time.
- No agents crashed in a 6 vs. 6 market.
- Nearly all robots crashed at some point in a 12 vs. 12 market.
- About 30-40% of assignments failed to be put on the orderbook when there were 24 agents in the market (12 vs. 12), irrespective of how the agent's sleep-time was configured.

As no agent crashed in any of the 6 vs. 6 agent markets, we can conclude that ExPo is suitable for the further experiments I will carry out. At this level, the amount of assignments failing to be put on the orderbook in the market was also fairly insignificant for all variations of sleep-time. However, to maximise efficiency of my experiments, where speed is not of utmost importance, I use a 4 second sleep time for all agents for the remainder of my experiments.

4.3.2 Stage 2: The Influence of pMax on AA Traders

Stage 2 of experiments was to test the effect of the parameter *pMax* on the performance of AA traders. Using the same market setup as Stage 1, and using the sleep-time result from the previous experiment too, 2 different versions of a typical AA trader (AA_L and AA_H) were tested both homogeneously and heterogeneously against each other.

The biggest indication of unfairness comes from looking at the homogeneous auctions of these two variations of AA. The measure of Smith's Alpha.

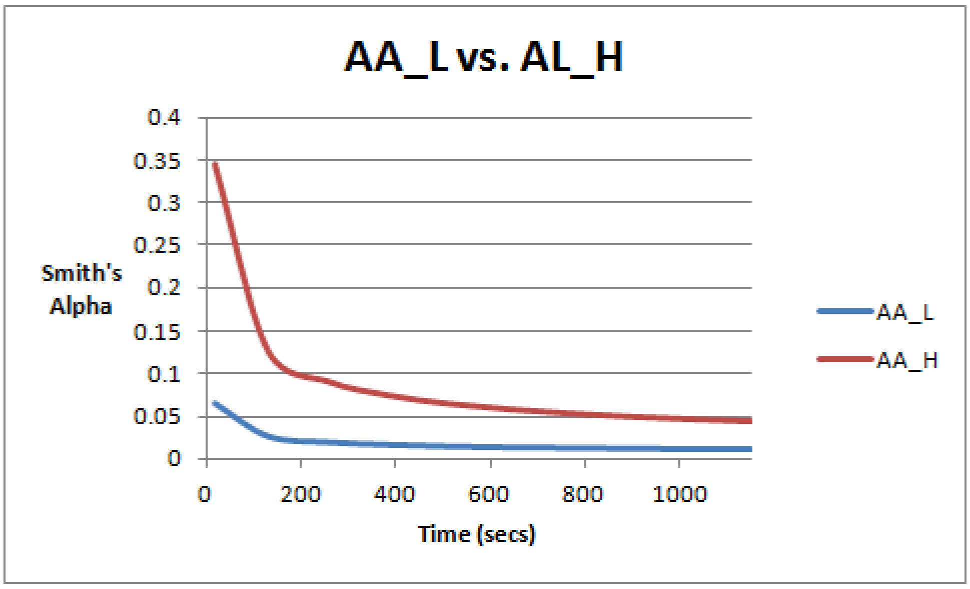


Figure 4.14: A comparison of Smith's Alpha over time between AA_L and AA_H

4 out of the 5 heterogeneous experiments conducted with AA_L vs. AA_H were also won by AA_L, but it was not deemed to be significant to a value of 10.3% by applying Robust Rank Order to the experiments (0.625 critical value). This is probably mostly due to the small amount of tests performed here, and the prevalence of one outlier, and one could expect the difference to be significant for a larger dataset of completed tests.

However, from observing the difference in Smith's Alpha, we can safely conclude that the value of pMax effects the performance of the AA strategy. By manually setting this value to be close the equilibrium, the agent is getting an unfair advantage.

To fix this, and essentially make the AA strategy more *adaptive* to any market it is exposed to, I then implemented a small fix that would take the reliance of a predetermined market price out of the picture. I figured that the best way of knowing what the maximum price in the market can be related to the assignments each agent receives – readily available knowledge for all agents. Therefore, pMax was set to be equal to two times the maximum assignment limit price received:

```
void *addAssignment(int *id, double *limit_price, int *quantity, const char* type)
{
   int order_id;
   assignment_t *a = std_addAssignment(id, limit_price, quantity, type);

    if(isFirstAssignment)
 {
  limitPrice = a->limit_price;
 pMAX = 2 * limitPrice;
  resetEquilibriumEstimator();
 }
 else
 {
  if (limitPrice != a->limit_price) limitPrice = a->limit_price;
  if (pMAX <  (2 * limitPrice))pMAX = 2 * limitPrice;
 }
}
```

I then tested out this strategy, hereforth known as *AA_D*, and compared the homogeneous trials to the two native versions of AA:

Table 4.5: AA - Comparison of pMax Values

Strategy	**Efficiency**	**Smith's Alpha**	**Profit Disp.**
AA_L	0.99937280	0.011404418	97.26
AA_H	0.99936500	0.043605286	204.35
AA_D	0.99932267	0.046874235	253.39

It's interesting to see that *pMax* doesn't affect the efficiency of AA traders, but has a fairly dramatic affect on Smith's Alpha and profit dispersion. The newly modified AA_D can be seen to be easily beaten by AA_L, with a much higher rate of convergence to equilibrium, which we would expect given that the same conclusion was drawn from AA_H and AA_L comparisons. We have already established that AA_L is given unfair information about the market.

There was no statistical significance between the efficiencies or market (Smith's) alpha of the homogeneous AA_D and AA_H. I believe the main reason why AA_D didn't significantly beat AA_H in these test is due to the assignment distribution pattern. Assignments in all experiments are dealt out in order, meaning that the lowest and highest price either side of the market are dealt out at the beginning, affecting the assignment of *pMax* for at least the first few assignments.

From the results, we can conclude that *pMax* has an effect on the rate of equilibrium convergence in the market. Realising that the variable pMax could be a good value to have available to the robot (i.e. some idea of a market *maximum*), I decided to link assignment prices to maximum market price, representing a dynamic alternative to a set value. This means that if an AA_D trader was put in a market where the equilibrium was just $p_0 = 5$, it would be able to be instantly efficient in this market too, whereas AA_L and definitely AA_H would have significantly lower success due to the fact that pMax was artificially high to begin with and therefore their convergence to equilibrium would take longer. A setting of twice the maximum limit price is of course just an example used for these experiments; this could be optimised using an alternative technique.

Therefore, due to the fact that AA_D is now a fully *adaptive* version of the AA trading strategy, which doesn't have a strict value of *pMax* and instead adapts this value from their provided assignments, I use this strategy in the remainder of experiments.

4.3.3 Stage 3: The Influence of *MaxSpread* on AA Traders

The premise of Stage 3 of experiments was to test the effect of the parameter *MaxSpread* on the performance of AA traders, first introduced in De Luca's implementation of an AA trader in OpEx [13] [14]. Using the same market setup as the experiments in Stage 2, and using the dynamic-pMax version of an AA trader (AA_D), a version of AA with *MaxSpread* set to 15% (as it was in the Foresight experiments) and a version without were tested both homogeneously and heterogeneously with each other. The version with the *MaxSpread* condition

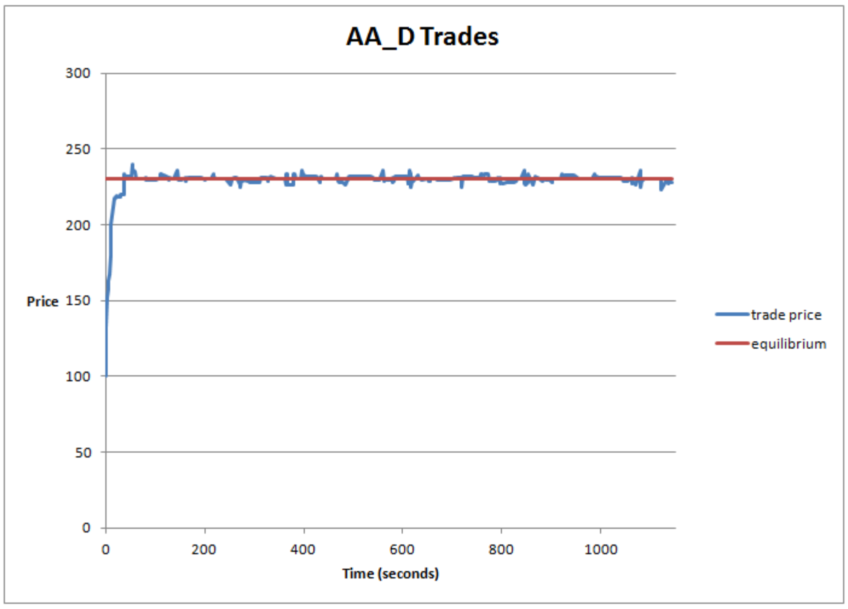


Figure 4.15: A typical auction illustrating trades made by homogeneous AA_D agents

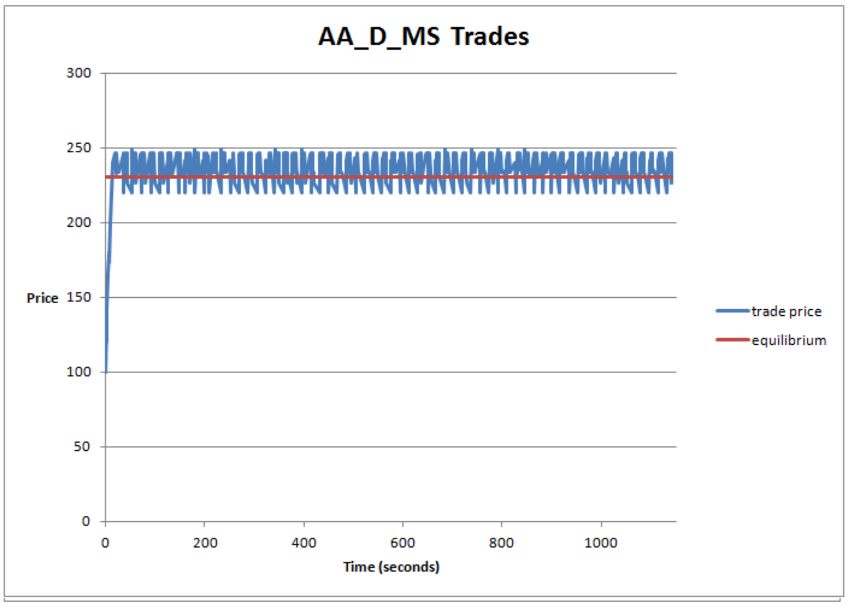


Figure 4.16: A typical auction illustrating trades made by homogeneous AA_D_MS agents

As can be seen in the trading dynamics expressed in Figures 4.16 and 4.15, and looking at the average metrics in Table 4.6, the AA trader with the *MaxSpread* condition performs quite considerably worse in homogeneous markets compared to a trader without this implementation. It can be seen that trades are less efficient, bringing in less of their maximum theoretical profit, and also have a significantly higher value of Smith's Alpha, indicating less convergence to equilibrium. These two findings are understandable, as essentially, the AA_D_MS is accepting prices in the market up to 15% different from what they've calculated is a competitive deal (their understanding of what market equilibrium is).

Another interesting thing to note here is looking at how many how many trades were made by each strategy. AA_D_MS made about 10% more trades than a normal AA trader (AA_D) and provided the most liquid market out of all of the strategies. It should be noted that although AA_D_MS made more trades, they were not in total more profitable.

Table 4.6: Strategy Leaderboard

Strategy	**Efficiency**	**Smith's Alpha**	**Profit Disp.**	**Total Shouts**	**Total Trades**
ZIP_S	0.97440323	0.06642625	678.57	4245	582
ZIP_M	0.99471400	0.05285961	308.59	7479	594
AA_D_MS	0.98780180	0.0657801	530.46	4036	639
AA_D	0.99932267	0.04687423	253.39	4104	577

The difference was significant in heterogeneous experiments too, with AA_D achieving significantly higher efficiency compared to its *MaxSpread* counterpart (significance level $p<=0.004$). From these findings, we can conclude that the *MaxSpread* condition is not optimal, and reduces the efficiency and performance of the AA strategy. Although previous experiments using De Luca's implementation of AA traders (and hence using *MaxSpread*) confirmed Vytelingum's findings [23] that AA traders were the dominant agent strategy [23] [14], my findings suggest that their dominance has been understated. This is especially true in the latest research conducted part of the *Foresight* programme [14], where the study found that humans appeared to be more dominant than even the agent-dominant AA strategy in continuous replenishment markets. This therefore calls this previous experiment into consideration, possibly invalidating the results found.

One point of discussion in the *Foresight* paper is that they hoped that the review would encourage further research and replication of key results; something they believed was not practised enough. This finding stresses this point even further; that it is vitally important to replicate key research in the field of experimental study.

4.3.4 Stage 4: Market-Speed Experiments

Stage 4 of the experiments sees ZIP_S (based on Cliff's original implementation), ZIP_M, and AA_D (dynamic-emphpMax with no *MaxSpread* condition) compete with each other both homogeneously and heterogeneously at different speeds of market.

The reason why these speeds are chosen is that they provide a good spread of speeds, with individual agents receiving new assignments as fast as 3 seconds apart from each other in the fastest market, and as slow as 24 seconds apart from each other in the slowest markets. We choose this speed for the slowest market for a particular reason - it is similar to speed of the market in the tests conducted in the *Foresight* paper which also uses continuous markets. We are therefore able to draw comparisons from results.

First, by looking at homogeneous environments, we can analyse the market performance.

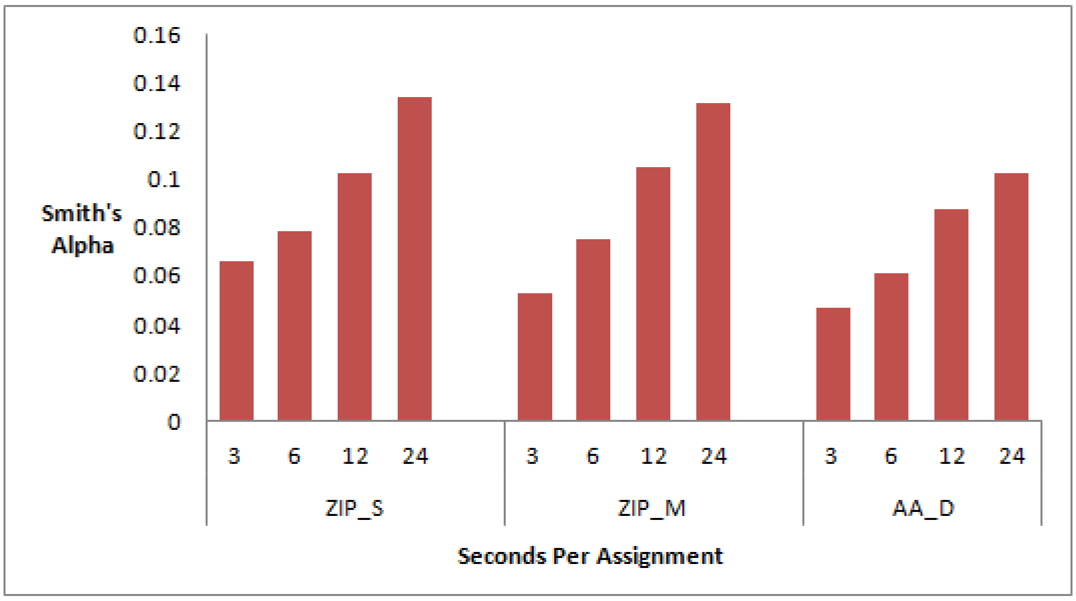


Figure 4.17: Smiths Alpha Comparison at Different Market Speeds

By looking at the comparison of Smith's Alpha in Figure 4.17, we can see that the rate of convergence to equilibrium is higher in faster markets across all homogeneous tests of agent strategies. This is an understandable result, as there are more assignments dished out in faster markets. With more assignments being traded at nearer-equilibrium prices as the auction progresses, the rate of convergence over the time period is expected to be lower.

However, although Smith's Alpha gives us an idea of market performance, we now look at the average efficiency achieved by each agent in homogeneous environments.

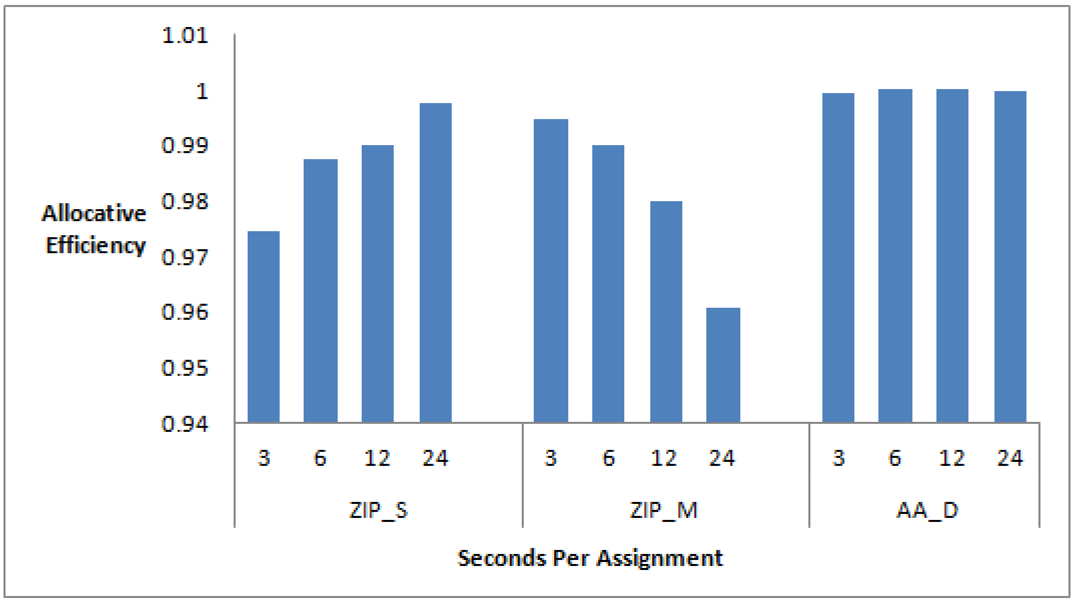


Figure 4.18: Allocative Efficiency Comparison at Different Market Speeds

Figure 4.18 shows a very interesting pattern in the efficiency of agents, especially for ZIP_M, which updates all current orders on every wake from their sleep-cycle, as opposed to their most profitable order. We can observe that the efficiency of ZIP_M is significantly less in slower auctions, whilst the other two strategies benefit from a market speed decrease.

By pitting each strategy against each other, I've constructed a winners table:

Table 4.7: Robust Rank Order Comparison of Strategies at Different Market Speeds

	3 Seconds per Assignment		**6 Seconds per Assignment**	
Competitors	**W/L**	**Significance**	**W/L**	**Significance**
AA_D vs. ZIP_S	5/0	$(p <= 0.004)$	5/0	$(p <= 0.004)$
AA_D vs. ZIP_M	3/2	$(0.028 < p < 0.048)$	5/0	$(p <= 0.004)$
ZIP_M vs. ZIP_S	5/0	$(p <= 0.004)$	4/1	$(0.004 < p < 0.008)$

	12 Seconds per Assignment		**24 Seconds per Assignment**	
Competitors	**W/L**	**Significance**	**W/L**	**Significance**
AA_D vs. ZIP_S	4/1	$(0.048 < p < 0.103)$	5/0	$(p <= 0.004)$
AA_D vs. ZIP_M	5/0	$(p <= 0.004)$	5/0	$(p <= 0.004)$
ZIP_M vs. ZIP_S	1/4	$(0.048 < p < 0.103)$	2/3	not significant

The findings from heterogeneous competition suggests support for the findings of efficiency within homogeneous markets. In fast markets, ZIP_M performs strongly, even challenging AA performance in fast markets. However in slower markets, ZIP_M struggles to perform against even it's ZIP counterpart, resulting in either non-significant difference from the efficiency of ZIP_S, or even inferiority at times (indicated in red at a market speed of 12 seconds). The AA strategy doesn't seem to be effected much at all by market speed when challenging the other two (ZIP) strategies, which is supported in the results of the homogeneous tests too.

From analysing all of these results, we can conclude that AA seems to be the dominant agent strategy across all market speed experiments, performing significantly better that the two versions of ZIP in terms of allocative efficiency and Smith's Alpha. We can also conclude that ZIP_M is a superior version of ZIP to that of ZIP_S in fast continuous replenishment markets; significantly and conclusively beating its counterpart in markets where assignments are handed out more regularly. Although once the market is slowed down, this superiority is somewhat diminished. It was seen early on in Stage 1 (Figure 4.5) that the dynamics of the ZIP_M trader were somewhat erratic for a while until convergence after about 350 seconds in a fast paced market. There is only one thing that is different in the code of a ZIP_M agent and a ZIP_S agent; the policy of updating orders. When ZIP_)M is still quoting around 4-6 assignments every 4 seconds, but the market isn't moving as quick, it's possible that it suffers from overfitting the orderbook, only for orders to be bought (or sold) for not as allocatively efficient profit.
Therefore, one possible reason for ZIP_M's decline in efficiency and dominance against counterpart ZIP_S' strategy, is therefore that it struggles to reach equilibrium as fast when quoting and updating so many orders.

Further investigation is therefore needed to determine the effect of order updating strategy on different types of agent. Its been suggested that updating all current orders is a superior order-strategy in fast markets for the ZIP trading strategy, but it would be good to test this on AA too in future research.

4.3.5 Stage 5: Dynamic Markets Experiments (Market Shocks)

Stage 5 of the experiments looks at how ZIP_S (based on Cliff's original implementation), ZIP_M, and AA (dynamic-pMax with no maxSpread) deal with market shocks naively, i.e. without any modifications.

In previous experiments, market (price) shocks have only been tested in the 'trading days' format of experiments, akin to Smith's original experiments [18]. A market shock occurs in continuous double auctions (CDAs) when assignment prices rise; as these sequences form the supply and demand in the market. To the best of my knowledge, market shocks have never been explored in a continuous replenishment environment, which is what I investigate in this stage of experiments.

It is important to analyse dynamic markets, where market shocks occur, as well as simpler static markets which I have investigated up to this point. This is because all real-life markets are dynamic - a market shock could occur at any time due to supply-side or demand-side factors. By exploring dynamic markets in these experiments, we can get a clearer idea of the real economics behind supply and demand in action, and some examples of realistic situations that could occur in real-life markets.

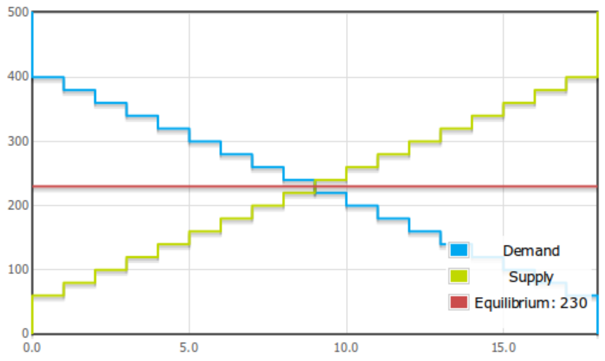


Figure 4.19: M1 - Assignment schedules with equilibrium = 230

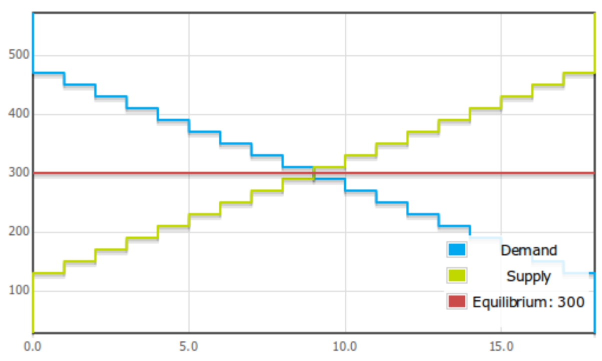


Figure 4.20: M2 - Assignment schedules with equilibrium = 300

The two different markets used in these experiments can be seen in the above figures. A price rise market shock occurs where assignments of M1 (the supply and demand in the market) *transition* to the new supply and demand of M2. A price fall market shock occurs when assignments of M2 transition to those of M1. It is a transition because although all new assignments (sellers' supply and buyers' demand) are now valued higher or lower, existing assignments that exist on the orderbook actually create a different picture of supply and demand in the market.

Typical graphs of how the strategies performed when exposed to both M1M2 & M2M1 market shocks are shown below.

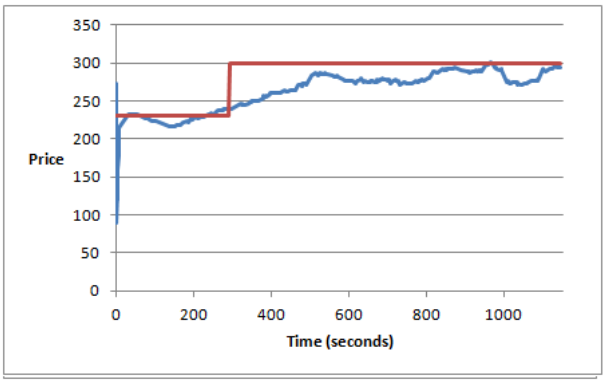


Figure 4.21: ZIP_S – market shock (price rise)

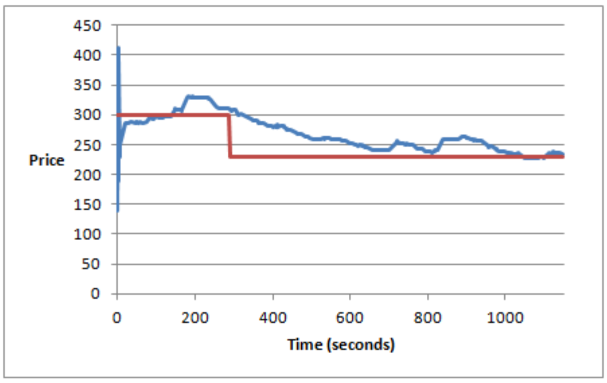


Figure 4.22: ZIP_S – market shock (price fall)

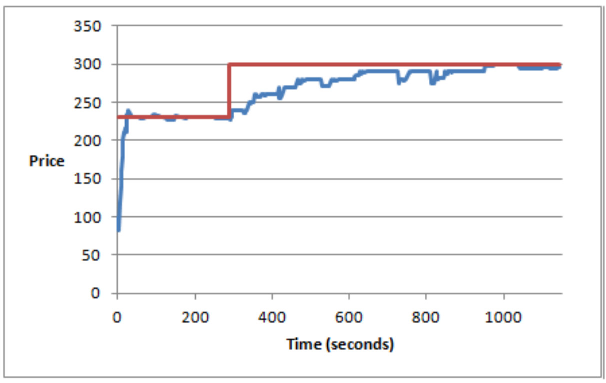


Figure 4.23: AA_D – market shock (price rise)

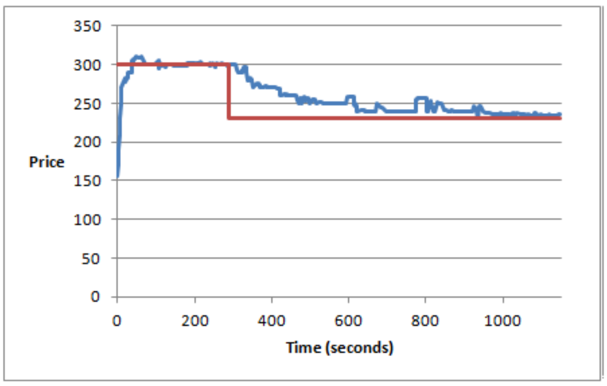


Figure 4.24: AA_D – market shock (price fall)

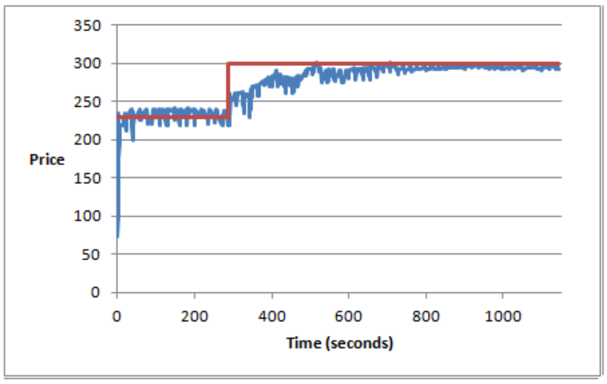


Figure 4.25: ZIP_M – market shock (price rise)

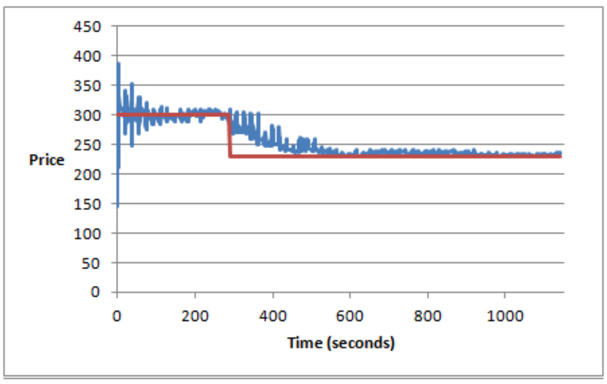


Figure 4.26: ZIP_M – market shock (price fall)

The difference between the two formats of market (*trading day* and *continuous replenishment*) and the corresponding dynamics of the market after a shock has been initiated is key to understanding the results presented in the diagrams above. Given that supply and demand in a market are determined by the number of available units to buy and sell, market equilibrium is given by the crossing of the demand assignment schedule (total buyers' assignments) and supply assignment schedule (total sellers' assignments). When a market shock occurs in a continuous replenishment market, although all new assignments coming into the market are immediately priced higher (or lower) than previous assignments, previous un-traded assignments still exist in the market. This therefore creates a gradual shift in actual market equilibrium towards a new equilibrium. This new equilibrium will get ever closer to the equilibrium of the new supply and demand assignment schedules, as the proportion of new assignments (with their new value) becomes far greater than the proportion of previous assignments left behind on the orderbook.

The reason why we use this model of persistent assignments, rather than retractable assignments, is because we assume that the traders are *sales traders*; assigned to try and sell (or buy) all assignments provided to them. This assumption is almost intrinsic of the experimental set up; both the one I conduct here and all the previous CDA agent or agent-human experiments that I've discussed in this thesis. It also provides us with a good model of noise, as old assignments are left behind un-traded but still on the book. The agents I study here haven't been designed to take into account any extra information other than shouts and transactions in the market, and so they have some limitations in their behaviour when market shocks are present in continuous markets. Here I aim to improve on this weakness.

It can be seen that the effects of a price rise and price fall are symmetric in the diagrams of naive (unmodified traders) market shock performance above. Therefore I investigated a market shock of a price rise for the remainder of my experiments, with a view to conclude results for a market shock in either direction. In the homogeneous experiments, without any modifications to the agents, the results can be seen in Table 4.8.

Table 4.8: Profit Spread in M1M2 Market Shock (Homogeneous)

	Average Profit Per Trade		
Strategy	**Buyers**	**Sellers**	**% difference**
ZIP_S	97.08	71.65	35.50%
ZIP_M	90.62	72.50	24.99%
AA_D	98.28	69.46	41.49%

Every buyer in each market outperformed every seller. A reason for this can be drawn from the fact that any transaction that a buyer can make after the market shock is potentially more profitable for the time the market is below the new assignment schedule equilibrium p^s. Given higher limit prices l^s around this new price p^s, every trade is profitable where $p_i < l_i^s$ (where p_i is the price of the trade). But with the actual market equilibrium trailing below p^s for a considerable amount of time, buyers are able to pick up deals cheaper than expected in this 'lag' period. For example, in Figure 4.23, the area between the new assignment schedule equilibrium line (in red) and the trading performance is profit that sellers are losing out on.

Additional key findings from this homogeneous study were:

- The spread in profit per trade between ZIP_M buyers and sellers was significantly smaller than the spread between ZIP_S buyers and sellers ($0.071 < p < 0.089$).
- The spread in profit per trade between ZIP_M buyers and sellers was significantly smaller than the spread between AA_D buyers and sellers ($0.071 < p < 0.089$).
- There was no significant difference in spread in profit per trade between ZIP_S buyers and sellers and AA_D buyers and sellers.

As ZIP_M traders seem to adjust to a market shock quicker than the other strategies in a homogeneous environment, it could be due to their ability to update their orders quicker. For example, other strategies can only update their most profitable order per sleep cycle, which means 1 price per 4 seconds is raised in value. ZIP_M however updates all of their present assignments per sleep cycle, which would cause more of an impact on the orderbook – other strategies would be reacting to 4 price rise signals rather than just 1. Therefore it can be suggested that because of the homogeneous market, ZIP_M agents are able to facilitate a quicker rise to the new assignment equilibrium.
However, when pitting the agents against each other in heterogeneous trials, using the same market shock, we see quite a different result.

Table 4.9: Market Shock (Price Rise) - Spread of Profit

	Average Profit Per Trade		
Heterogeneous Experiments	**Buyers**	**Sellers**	**% difference**
ZIP_S vs. ZIP_M			
ZIP_S	98.81	71.81	37.58%
ZIP_M	95.41	68.62	39.05%
ZIP_S vs. AA_D			
ZIP_S	96.99	71.66	35.35%
AA_D	95.91	68.50	40.01%
AA_D vs. ZIP_M			
AA_D	95.83	70.79	35.38%
ZIP_M	98.29	70.23	39.96%

From these results we can conclude that ZIP_M, when in a market by itself, adapts and reacts to market shocks faster than the other agent strategies, mostly in part because ZIP_M traders can update all of their current orders at once – leaving less 'easy-pickings' for buyers (if the price has gone up) or sellers (if the price has fallen). However, in a heterogeneous environment, as soon as they are competing against agent strategies that take longer to adjust to the new assignment equilibrium point, it seems the ZIP_M strategy is outperformed by the competing strategies. This could be because they are updating their prices consistently too high each time missing out on trades, or that they are getting distracted by shouts in the market due to their competitors not reacting as quickly.

4.3.6 Stage 6: Exploratory - Assignment-Adaptive Agents

If human traders were being provided with, say, new assignments of greater limit price to sell (or buy) in the market, it can be expected that they will pick up on the fact that the good they are dealing with is rising in value. They stand to make a lot more (or less) profit in the market due to this price shock and so would be expected to act on this foreseen shock in their best interests. For example sellers, who would see their potential profit margins on previous assignments rise, would be able to ask for more money in exchange for their holdings.

However, agent strategies thus far have had no mechanism of analysing the assignment prices given to them over time – something which human traders have the innate ability to do. One can see from the market shock experiments above that when a price rise market shock occurs in the market, agent sellers don't know that their existing holdings have become more valuable. In terms of profit, they actually lose out in the market, letting buyers pick off their cheap and slow-rising prices. To address this issue, I explored the possibility of modifying agents so that they could react and adapt to market shocks in the way we would expect humans to.

Here, I focus on just one agent strategy, ZIP_M, because this strategy seems more adaptable than the other strategies to market shocks natively – I thought it'd be interesting to see if I could improve it. The first step was to give each trader a memory of the last limit prices it had received, seeing as this is the prime indication of a price rise (new limit prices rise and fall based on the shock). The implementation of this assignment price window was very similar to that of an AA trader's historic trade price window; different only in the fact that we want to access it from oldest to newest. Every time a new assignment was received by a trader, it was immediately stored in this assignment price window, limited by a history of 20 prices. This value for the size of the window was picked as it provides a big enough subsection of recent limit prices to analyse, but also given that assignments are allocated in groups of 6 assignments, it's a number which ensures no bias, as it is non divisible (a divisible window would provide the same result for every window move). Analysis on these prices would only happen once this window was at full capacity.

Using the history of trade prices, it was necessary to incorporate a method to analyse change of prices over time. The key priorities for this were:

1. Robust to outliers.
2. It had to give a good valuation of change rather than absolute differences. This would not be achieved by such methods as averaging the prices in the window.
3. The values representing change had to be consistent with and similar to each other.

After eliminating many calculation methods, such as mean, median and cumulative difference from average, I decided on taking a different approach; one that would take the *gradient* of change. By treating the price window like a time series on a graph, I figured that taking the line of best fit from the data might be the best way to establish change. Given this, I decided on calculating the simple regression model of Ordinary Least Squares (OLS) [19], given by

$$Gradient(\nabla) = \frac{\sum x_i y_i - \overline{y}\sum x_i}{\sum x_i^2 - \overline{x}\sum x_i}$$

where y would be the position of the limit price l_i in my assignment price window, in chronological order. I then plugged the result of this into a simple logarithm function

$$shockIndicator(\phi) = \begin{cases} -\ln(-\nabla + 1) & \text{if } \nabla < 0 \\ \ln(\nabla + 1) & \text{otherwise} \end{cases}$$

to help accentuate positive change as a value above 1, and a negative change as a value below -1.

Now that I had obtained a consistent number representing limit price change at any point in time (after the window becomes full), I needed to decide how to use it in my ZIP_M agent's strategy. The primary goal of using this value was to increase the profit margins of the sellers when $\phi > 1$ (indicating higher possible prices in the market) and to increase the profit margins of the buyers when $\phi < -1$ (indicating lower possible prices in the market). The reason why I did not aim to alter the strategies of buyers when a price rise is predicted is that they would not benefit from it, and vice versa for sellers and a price fall. They could try and trade their holdings faster to gain maximum advantage, but this strategy might cause them to become less profitable in the long run (as they're accepting non-optimal deals at the time) and they don't know that traders on the other side of the orderbook are definitely going to change their prices dramatically too.

The first implementation I did was, on any assignment received, to update all existing order prices based on an arbitrary percentage increase of the top of the opposing side of the orderbook. It would update prices only in it's memory at this point, until the trader wakes from their sleep-cycle, where it would update the order prices themselves. The percentage I chose for this implementation was 20%; a value that seemed to bring about significant change to trader quotes after a market shock in my particular experimental set up.

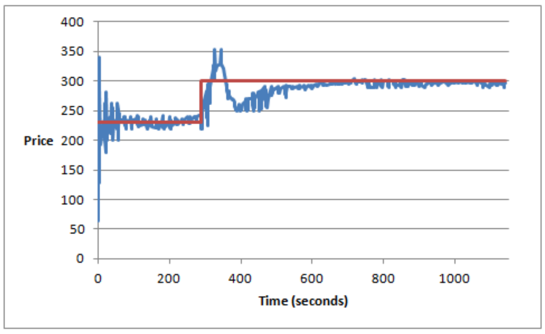


Figure 4.27: ZIP_M – Just using assignment price window to react to market shock

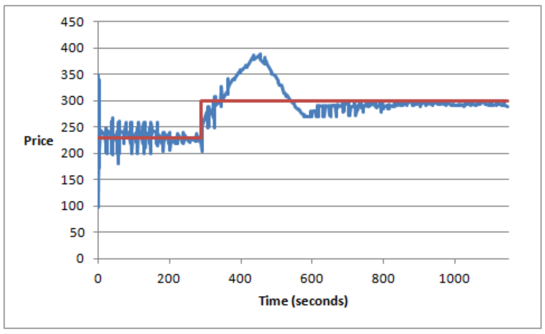
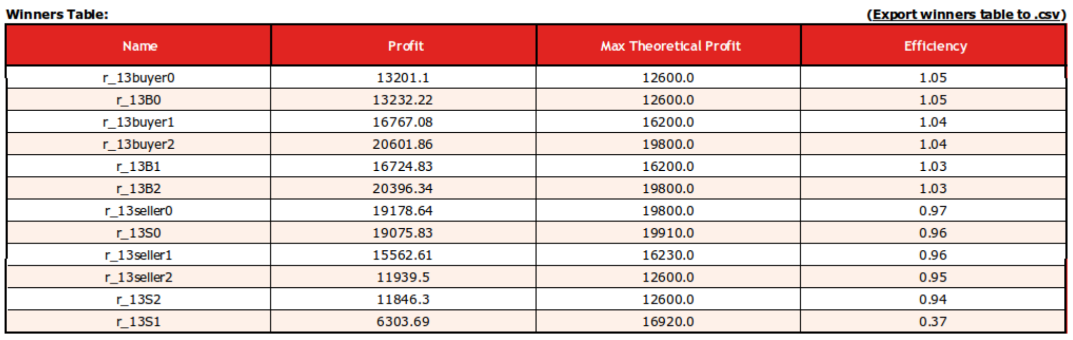


Figure 4.28: ZIP_M – cumulative steadying of shock reaction to react to market shock

The initial results can be seen in Figure 4.29, and although sellers react by rapidly increasing prices in the way we would expect by increasing their prices to a level which all participants could now afford, it soon dropped off steeply. This happened because the algorithm as it stands only counts significance or no significance, with no intermediate, or 'winding-down', phase in between, and resorts straight back to current true market convergence to new prices. Therefore, I added a strategy that would essentially 'wind-down' increases in price but still demand higher than market settling price for their assignments. This would be based on the total cumulative value of ϕ, and starting at the initial percentage increase (20%), the increase in prices would be directly related to a climb-down strategy. The results can be seen in Figure 4.28.

When all sellers employed this extra shock-adaptive strategy in a market, I found that every seller benefits, being able to on average either match or beat buyers average profit (compared to previously when they would lose consistently and by a margin of $\sim$ 25%). What's even better is that next to no profit was lost in the market itself, suggesting this strategy enhances profit equality between buyers and sellers during a market shock. However, when tested in environments where other 'dumb' sellers existed (i.e. sellers without this shock-adaptive strategy) the results were very interesting:

- When half of a 6 seller (12 agents total) market used this shock-adaptive strategy, they lost out significantly to the other dumb sellers in the market – who ended up in turn beating everyone in the market, including buyers.
- When only one shock-adaptive seller was included in a market of 12 traders (5 other sellers), again it helped to boost the profits of every seller in the market apart from itself, who again lost out significantly.


Winners Table: (Export winners table to .csv)

Name	Profit	Max Theoretical Profit	Efficiency
r_13buyer0	13201.1	12600.0	1.05
r_13B0	13232.22	12600.0	1.05
r_13buyer1	16767.08	16200.0	1.04
r_13buyer2	20601.86	19800.0	1.04
r_13B1	16724.83	16200.0	1.03
r_13B2	20396.34	19800.0	1.03
r_13seller0	19178.64	19800.0	0.97
r_13S0	19075.83	19910.0	0.96
r_13seller1	15562.61	16230.0	0.96
r_13seller2	11939.5	12600.0	0.95
r_13S2	11846.3	12600.0	0.94
r_13S1	6303.69	16920.0	0.37

Figure 4.29: Winner table of Auction 13 - r_13S1 is modified to boost values

- The profit spread between buyers and sellers of 12 agent markets using all dumb strategies was much higher than in markets where at least one shock-adaptive strategy was used, although the shock-adaptive strategies suffered heavily.

These findings are interesting. It suggests that modified shock-adaptive strategies need to work together as a group, otherwise others profit from their work and missed opportunity to sell. It only took one of these adapted agents to be in the market to help facilitate more competitive selling given the market shock. This suggests that these adapted strategies act as *price pushers* in the market – if they're working alone, others will benefit from the pushed up prices, but if they're working together, every strategy can benefit from small buyer/seller spreads.

Put another way, the asks quoted by the modified ZIP_M are much higher than the rest of the market. The unmodified agents react to these higher shouts by increasing their own profit margins. Hence, the behaviour of the modified agent has a secondary impact on the behaviour of the unmodified ZIP_Ms, which turns out to benefit these agents, but not the modified agent. If the entire market is modified then everyone benefits, but if any unmodified agent enters the market, it parasitically benefits from the behaviour of the modified and will flourish, eventually exterminating the modified agents from the marketplace.

In part, these findings could be due to the simple example strategy that I've implemented here to get sellers (and buyers) trading within a tighter average spread after detecting a shock. The implementation chosen here has little reliance on absolute or any meaningful values (otherwise I would have just shocked the shock-adaptive agents 70 in price), but it is possible that they are environmentally-specific to the testing environment I was operating in. This method also hasn't been configured to deal with the different paces of possible rises or falls in price; it is just mentioned as a framework for future research. What would be more suitable here instead of this simple implementation is something more like an adaptive learning rule – Widrow Hoff rule for example [25] that was used as the basis of the ZIP algorithm market learning. This would ensure that the agents are fully adaptive and ready to participate into any auction.

One important piece of research that should now be conducted in the field is to see how humans react to market shocks in a continuous replenishment environment, like these tests in ExPo. This would be able to provide insight into how quickly humans converge to the equilibrium of the new supply and demand entering the market, and whether we can learn anything from human decisions that we can transfer to algorithmic trading strategies.

Chapter 5

Conclusion

5.1 Evaluation of Results

Throughout the course of this dissertation, I have implemented, conducted and analysed a range of experiments to investigate the performance of algorithmic trading agent strategies in the more realistic continuous replenishment auction environment, akin to real financial exchanges. By carefully scrutinising recent research, and looking over past implementations of algorithmic agents, I have managed to evaluate conflicting versions of strategies, and implement my own adaptive fixes where possible. I have also investigated dynamic markets for the first time in this environment, and adapt agents accordingly to be able to detect market shocks.

I can conclude that:

1. AA has been proven to perform significantly better than ZIP (in any form) in a continuous replenishment market, thus reinforcing the findings of the recent *Foresight* Driver Review paper [14]. It also supports the findings of Vytelingum (2006) [23] and De Luca & Cliff (2011b) [13], who found AA dominated in traditional CDA set ups. The AA strategy was found to be dominant in all static market tests, which included being tested in markets of varying speed and being altered to operate at a variety of speeds.

2. The *maxSpread* property that De Luca [11] incorporated into his research has been found to perform significantly worse than a version of AA without this condition. This casts doubts on the findings of the recent *Foresight* paper, in which he was a collaborator, which found that humans could beat even the agent-dominant strategy of AA in continuous replenishment environments. This highlights the importance of replication of studies in this field. With regards to this maxSpread condition, De Luca has recently recognised that the implementation of this was indeed a bug. [1]

3. I've exposed pMax, information provided to AA regarding the maximum allowed bid or ask in the market, to be an unfair variable in the strategy of AA. The value of this variable effects the level of convergence in a market (Smith's alpha), with small (close to equilibrium) values of this making an agent extremely efficient, but unfairly as they have been given more knowledge about their market. By taking a simple dynamic approach, I've presented how it can be made adaptive without significant loss in efficiency or market convergence.

[1] http://sourceforge.net/p/open-exchange/wiki/Home/

4. A ZIP strategy able to update multiple profit margins (ZIP_M) performs significantly better than a version of the same strategy with just a single profit margin to update (ZIP_S) in fast markets. As the market slows, this dominance is lost. This suggests that a non-optimal version of the ZIP strategy has been used in recent studies, such as Vytelingham (2006) [23] and De Luca & Cliff (2011a, 2011b) [13] [12].
5. Agents are able to be adapted to accurately determine when a market shock happens. By using simple regression, a shock in the market can be known about instantly.

The new all-agent tests I've conducted here do not therefore change the previously discovered dominance hierarchy of known algorithmic traders. This suggests that performance in static continuous replenishment markets is similar to that of traditional trading day market formats that more closely follow Smith's experimental design [18], as similar performance can be achieved in both.

What is most interesting though is that when testing these agent strategies in dynamic markets, the multiple profit margin ZIP agent seemed to beat the other strategies in narrowing the spread between the profit of buyers and sellers. This suggests that it was quicker to adapt to the new emerging equilibrium than the other strategies were, when tested in homogeneous markets. Therefore, this could be an upset for the dominance hierarchy of agents, especially seeing as AA was designed to operate efficiently when encountering a market shock. Of course, more research will need to be carried out before concluding anything, but it is likely that an agent who can best perform in dynamic markets would be the most important strategy in the field - because nearly all real life markets are inherently dynamic.

Further to these findings, I have observed the following:

- Enhancing one trader to react more profitably to market shocks results in them losing out in the market unless everyone in the same position as them (i.e. fellow buyers or sellers) exercise the same strategy. Also, small minorities of modified strategies can minimise the profit spread for other buyers or sellers at the expense of their own performance.
- Convergence to equilibrium without the "NYSE" spread improvement rule was seen to be exhibited. Das *et al.* (2001) [6] originally thought the inclusion of this rule facilitated equilibrium convergence, but we see all agents in the experiments presented here converge to competitive equilibrium. This was probably due to the type of market I studied and the length of the continuous auction.

5.2 Evaluation of ExPo

Being the first person to use the ExPo platform in research, and with the platform still in development, a lot of preparation was needed to make it a stable and suitable platform to conduct research. Throughout the course of my dissertation, I have:

- Written over 20 SQL queries to extract and pre-analyse data, working with about half a gigabyte worth of database files.

- Created an automated queue system for experiments, with option to add competitors, and with stability controls to stop messy system exits affecting results or environment performance.
- Conducted stress tests of the ExPo system to help understand when and where the system can break and how best to run my experiments in a reliable controlled atmosphere.
- Run hundreds of hours of tests, performing six different stages of investigation, with all the tests running reliably and automatically after developing the necessary scripts to facilitate this.

The amount of time I spent patching up and improving the functionality of ExPo, along with all writing of detailed queries and scripts to provide everything I needed for this project, is not really expressed proportionally in this thesis. That's because I concentrated on providing as much interesting information and detail regarding experiments and results, as this is what I set out to investigate, and this is essentially the crux of the importance of this paper. However, by developing ExPo further and presenting it now as a system where these kind of experiments can be easily run from a standard Linux machine (or Windows machine if you're running a dedicated virtual machine like I was), I hope it will encourage others to use and contribute to the platform.

Of course, the platform still has some limitations, such as the inability to run more than 12 agents reliably in auctions. However, ExPo has been designed with scale in mind, and by setting it up as a web server, I'm sure the system will be able cope with big agent workloads much more efficiently, as the workload is spread across several PCs.

5.3 Research Framework: Recommended Extensions

The field of algorithmic trading research is huge, and in a commercial sense, incredibly profitable. So much is happening in commercial deployments, and so much money is spent on proprietary research, but relatively little is known in academia by comparison.

Here I present a few exciting extensions to the work I've completed thus far, in the hope that it will encourage others to pick up my work and carry on in one of these directions.

5.3.1 Human-Agent Iteraction in Dynamic Continuous Replenishment Auctions

This is an area in which I previously concluded in Stage 6 that would be invaluable to understand. How do humans react to market shocks in the middle of a trading period? Do we think that some people will react signifcantly quicker than others? If they do, will it be of benefit, or will they price themselves out of the market like the modified ZIP_M traders did in the experiments I ran?

A lot of questions arise from this research idea, as it is research that I believe will help us understand the real-life dynamics of market shocks in a competitive atmosphere.

5.3.2 Random Assignments

One of the limitations of my research, and the research of many others which have conducted similar experiments before, is that assignments are allocated in an order in the market. This is somewhat of an experimental artifact in the design of some auctions, and it would be a good first step in new research to look at ways to randomise the assignment order without affecting equilibrium.

5.3.3 Orderbook Depth

Financial exchanges often limit the display of their orderbook to show only the top few bids and offers in a market. Although other orders can be made, unless they are priced at a certain level, they may not be displayed to the public.

Looking into increasing realism once again, it would be an interesting step to investigate how the limitations on the display of data can effect the dynamics of a market of agents – trading strategies that rely on detailed shout and transaction information from the exchange orderbook. Does this limitation rule enhance the market as it reduces the 'noise' of untradable assignments?

5.3.4 AA_Multi and Other Algorithmic Traders

Now that I've introduced and studied ZIP_M, the same modifications could be introduced to AA, and performance between this implementation and the original AA strategy could be investigated. Being a different strategy from ZIP, different variables would need to be stored for each 'margin', such as aggressiveness and theta (a property of long term aggressiveness). Because it would act a little differently, it might be good to use as a benchmark to test against ZIP_M which employs the same order update rules.

I chose, given the time frame of this project and the state of previous research, to just investigate a couple of different agent trading strategies, and scrutinise their techniques and mechanisms. But this research could be expanded to cover other such agent strategies as GD, Kaplan Sniper, or even Genetic Algorithm inspired traders, such as ZIP60 [3].

5.3.5 Market Makers

Most research has been based on the simple assumption of separate buyers and sellers in the market with their separate roles. However, Market Makers (MMs), a class of trader that holds inventory of stock, quote both buy and sell positions to benefit from the bid-ask spread and provide liquidity in the market. There are officially appointed MMs in most financial markets to provide liquidity, although they are regulated and controlled participants. However, since the deregulation of the London Stock Exchange in 1986, there has been an influx of unofficial MMs enter the exchange, completely legally, with many of them adopting HFT strategies.

It would be interesting to study the practice of market making rather than just sales trading, which is implicit of our market design of separate buyers and sellers in a marketplace. To be able to do both, we might see some highly intelligent traders being developed that act on both buy and sell signals.

5.3.6 Sentiment Analysis

High-Frequency Trading (HFT) agents are now using financial information from the vast stores of information on the internet to help enhance their trading strategies regarding particular investments in a matter of milliseconds [15]. Real-time analysis of machine-readable news-feeds to compute market sentiment is becoming commonplace to help guide agent trading strategies, however the methods of analysis of financial information, let alone how they affect agent trading, are often commercial secrets and hence relatively little information is available in the public domain.

Previous research into HFT has strived to give agents 'equal' internal information (state of current orderbook, past orders, etc) about the market, trying to equal the playing field with human traders. But what happens when we equal the real-life playing field even more by allowing agents access to external information (news) about the market? This is a very exciting, and current, study that would investigate the unexplored field of adapting agent strategies automatically based on a predicted forecast about the market.

Appendix A

Data Examples

A.1 Robust Rank Order Worksheet

Table A.1: Example RRO calculation

auction_id	id	name	name	efficiency					
16	181	r_16S0	Zm4	0.997008833	1	U(YXi)	4	3	U(XYi)
17	187	r_17S0	Zm4	0.982029833	0		1	3	
18	193	r_18S0	Zm4	0.977905667	0		1	4	
19	199	r_19S0	Zm4	1.013766	1		5	0	
20	205	r_20S0	Zm4	0.979751333	0		1	3	
16	91	r_16seller0	Zs4	0.987578667	0	**U(YX)**	**2.4**	**2.6**	**U(XY)**
17	97	r_17seller0	Zs4	0.994201667	1				
18	103	r_18seller0	Zs4	1.001553833	1	**Vx**	**15.2**	**9.2**	**Vy**
19	109	r_19seller0	Zs4	0.960349833	0				
20	115	r_20seller0	Zs4	0.989586833	1				
						RRO	-0.09033	not significant	

I created an Excel spreadsheet that would calculate the Robust Rank Order of results between two populations. Please refer to Feltovich's study to find the critical value list [8]. All tests were done with 5 vs. 5 populations.

A.2 Example SQL - for Table 4.2

```
SELECT agents.auction_id, agents.id, agents.name, robots.name,
 AVG(agents.profit / agents.max_theoretical_profit) AS accurate_eff_average,
 AVG(smiths.smiths_alpha) as smiths_alpha,
 AVG(pd.profit_dispersion) as profit_distribution,
 AVG(shouts.num_shouts) AS total_shouts_average, smiths.dealcount,
 SUM(assigns.bottlenecks)/(COUNT(agents.id)/5) AS bottlenecks,
 COUNT(agents.id)/5 as num_agents,
 FLOOR((agents.auction_id-1)/5) AS id_group
FROM thesisstage1a.agents, thesisstage1a.robots,
     (    SELECT deals.auction_id,
      SUM(POW((deals.price - auctions.equilibrium), 2)*deals.quantity)
        AS sum_trades,
      sum(deals.quantity) as dealcount,
       (SQRT((SUM(POW((deals.price - auctions.equilibrium), 2)*deals.quantity)) /
        (sum(deals.quantity))) / auctions.equilibrium) AS smiths_alpha
          FROM thesisstage1a.deals, thesisstage1a.auctions
          WHERE deals.auction_id = auctions.id
          GROUP BY deals.auction_id
     ) AS smiths,
     (    SELECT logs.auction_id, COUNT(logs.id) AS num_shouts
          FROM thesisstage1a.logs
          GROUP BY logs.auction_id ORDER BY logs.auction_id
     ) AS shouts,
     ( SELECT agents.auction_id,
     SQRT(SUM(POW((agents.profit - agents.max_theoretical_profit),2))
       / COUNT(agents.id)) AS profit_dispersion
          FROM thesisstage1a.agents
          GROUP BY agents.auction_id
     ) AS pd,
     ( SELECT assignments.auction_id,
     SUM(assignments.quantity) AS bottlenecks, assignments.id
          FROM thesisstage1a.assignments
          GROUP BY assignments.auction_id
     ) AS assigns
WHERE agents.robot_id = robots.id
AND smiths.auction_id = agents.auction_id
AND shouts.auction_id = agents.auction_id
AND pd.auction_id = agents.auction_id
AND assigns.auction_id = agents.auction_id
AND agents.auction_id > 0
AND agents.auction_id <= 60
GROUP BY id_group
ORDER BY agents.auction_id, robots.name, efficiency DESC;
```

A.3 Example Ruby Script - AddCompetitorsStage4B.rb

```
#To be placed in the script/ folder
#Usage: script/rails runner -e production script/run_auction.rb <auction_id>
include RobotControl
include SocketMethods

#Arguments start with idx=2, since first two are taken by '-e' and 'production'
auction_id = ARGV[2].to_i #first argument
number_of_sequences = ARGV[3].to_i
robot_id = 0
repeats = 0

#if sequences
#------------------------------
if((auction_id >= 16) && (auction_id <= 20))
    robot_id = 2
    repeats = 1
end

if((auction_id >= 21) && (auction_id <= 25))
    robot_id = 3
    repeats = 1
end

if((auction_id >= 26) && (auction_id <= 30))
    robot_id = 2
    repeats = 1
end
# ------------------------------
if(robot_id != 0)

    buystring = 'B'
    sellstring = 'S'

    repeats.times do |j|

        group_num = (auction_id * (2 * number_of_sequences))
           - ((2 * number_of_sequences) - 1)
        puts group_num
        #number_of_sequences = 3

        # --SELLERS--
        number_of_sequences.times do |i|
          Agent.create(:auction_id => auction_id, :group_id => group_num,
                         :name => ["r_",auction_id,sellstring,i].join,
                         :agent_type => 1, :robot_id => robot_id)
          group_num = group_num + 1
        end

        # --BUYERS--
        number_of_sequences.times do |i|
          Agent.create(:auction_id => auction_id, :group_id => group_num,
                         :name => ["r_",auction_id,buystring,i].join,
                         :agent_type => 2, :robot_id => robot_id)
          group_num = group_num + 1
        end

        buystring = buystring + 'B'
        sellstring = sellstring + 'S'
    end
end
```

Bibliography

[1] SEC & BCG. Organizational study and reform. Report, U.S. Securities and Exchange Commission, March 10th 2011. Published by SEC.

[2] SEC & CFTC. Findings regarding the market events of may 6, 2010. Report, U.S. Commodity Futures Trading Commission and the U.S. Securities and Exchange Commission, September 30th 2010.

[3] D. Cliff. Zip60: Further explorations in the evolutionary design of online auction market mechanisms. Technical report, HP, 2005.

[4] D. Cliff and J. Bruten. Minimal-intelligence agents for bargaining behaviours in market-based environments. Technical Report HPL-97-91, HP, 1997.

[5] D. Cliff and C. Preist. Days without end: On the stability of experimental single-period continuous double auction markets. Technical report, HP, 2001.

[6] R. Das, J. E. Hanson, J. O. Kephart, and G. Tesauro. Agent-human interactions in the continuous double auction. *The Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI), Seattle, USA (August, 2001)*, 2001.

[7] D. D. Davis and C. A. Holt. *Experimental Economics*. Princeton University Press, 1992. p167.

[8] N. Feltovich. *Communications in Statistics - Simulation and Computation*, chapter Critical Values for the Robust Rank-Order Test, pages 525–547. Department of Economics, University of Houston, 2005.

[9] D. K. Gode and S. Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 1993.

[10] LSCITS. Large scale complex it systems, 2007.

[11] M. De Luca. Open exchange (opex), 2012.

[12] M. De Luca and D. Cliff. Agent-human interactions in the continuous double auction, redux. In *ICAART*, 2011.

[13] M. De Luca and D. Cliff. Human-agent auction interactions: Adaptive-aggressive agents dominate. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[14] M. De Luca, C. Szostek, J. Cartlidge, and D. Cliff. Studies of interactions between human traders and algorithmic trading systems. In *The Future of Computer Trading in Financial Markets*, volume DR13 of *Foresight Project*. Government Office for Science, UK, 2011.

[15] G. Mitra, D. diBartolomeo, A. Banerjee, and X. Yu. Automated analysis of news to compute market sentiment: Its impact on liquidity an trading. In *The Future of Computer Trading in Financial Markets*, volume DR8 of *Foresight Project*. Government Office for Science, UK, 2011.

[16] C. Preist and M. van Tol. Adaptive agents in a persistent shout double auction. Technical report, HP, 1998.

[17] S. Shang, J. Jiang, Y. Wu, G. Yang, and W. Zheng. A knowledge-based continuous double auction model for cloud market. In *2010 Sixth International Conference on Semantics, Knowledge and Grids*, 2010.

[18] V. L. Smith. An experimental study of competitive market behaviour. *The Journal of Polical Economy*, 1962.

[19] J. H. Stock and M. M. Watson. *Introduction to Econometrics*, chapter Estimating the Coefficients of the Linear Regression Model. Pearson Education Limited, 3rd edition, 2012.

[20] G. Tesauro and J.L.Bredin. Strategic sequential bidding in auctions using dynamic programming. In *AAMAS*, 2002.

[21] Gerald Tesauro and Rajarshi Das. High-performance bidding agents for the continuous double auction. Technical report, IBM, 2001.

[22] H. R. Varian. *Intermediate Micro Economics*, chapter 1. W. W. Norton and Company, 7th edition, 2006.

[23] P. Vytelingum. *The Structure and Behaviour of the Continuous Double Auction.* PhD thesis, University of Southampton, 2006.

[24] P. Vytelingum, R. K. Dash, E. David, and N. Jennings. A risk-based bidding strategy for continuous double auctions, 2004.

[25] B. Widrow and M. E. Hoff. Adaptive switching circuits, 1960.