Designing Intelligent Agents Coursework

Thomas Neal

Word count: 4352 words (excluding references and figures)

Insights into Market Dynamics: Evaluating the Performance of Diverse Trading Strategies and the Influence of Various Factors Within a Simulated Stock Exchange Environment.

Contents

Introduction

In the modern day's intricate financial landscape, the performance of trading strategies is of paramount importance to investors, financial institutions, and any individual attempting to turn a profit. As various trading strategies compete against one another in these markets, the effectivity and individual performance of the strategy ultimately determines investment returns, risk management, and overall market stability. As we have transitioned from the paper age to the digital/technological age, the field of algorithmic trading was birthed and has witnessed a rapid evolution, enabling traders to automate their strategies, decision processes, and reactions to market dynamics with exponentially higher speeds by utilising computational power. To put this evolution into perspective, multiple sources such as Yahoo Finance [1], QuantifiedStrategies [2], and TheRobustTrader [3] all claim that "algorithmic trading accounts for up to 60-75% of trading in the U.S, Europe, and major Asian Markets". This shines a light on a dominant force in the shaping of market dynamics and liquidity, influencing price movements, volatility, and driving trading volumes.

This research paper aims to investigate the performance of an array of different trading strategies within a simulated stock exchange environment and explore the factors contributing to the effectivity (or downfall) of these strategies. The simulated environment for this study is the Bristol Stock Exchange codebase [4] which is also described within a peer-reviewed paper [5]. By utilising this open-source simulation, we aim to facilitate a setting which is both realistic and controlled where various trading algorithms can be experimented with alongside tinkering with external market conditions to gather insights into the algorithms' performance and robustness within a dynamic market.

Key objectives of this research include:

- Conducting baseline performance comparisons to evaluate individual trading strategies in isolation.
- Performing comparisons to assess the relative strengths and weaknesses of different trading approaches and how they compete against one another.
- Conducting sensitivity analysis to understand how various trading strategies respond to varying market conditions and parameters.
- Investigating the adaptability of trading strategies to changing market dynamics, such as volatility and sudden market shocks.

- Investigating the role of latency dynamics including the delays between a trader's order and market execution, and trader proximity to exchanges.

The experimentation surrounding these objectives aims to provide valuable insights into market dynamics, the efficacy of trading strategies, and the design of more robust and adaptive trading algorithms developed based on the findings from this research. Through a combination of experimental evaluation and data analysis, this paper aims to contribute to the ongoing development regarding algorithmic trading and financial markets.

Related Literature

Bristol Stock Exchange

"The Bristol Stock Exchange (BSE) is a minimal simulation of a financial exchange running a limit order book (LOB) in a single tradable security" [5] created by Dave Cliff at the University of Bristol. By utilising BSE, researchers and students can explore the performance of different trading strategies, assess the impact of market variables and dynamics on the algorithm's results, and investigate the behaviour of autonomous trading agents. The associated paper written regarding BSE [5] provides detailed insights surrounding why the BSE was created, how it works, and what can be done with it.

Impact of Various Factors Influencing Trading Strategy Performance

The performance of trading strategies within financial markets is influenced by an array of factors, such as market dynamics, liquidity, volatility, and various other performance metrics. Research in this area has exposed the complex link between these factors and the effectiveness of different trading strategies.

Chordia, Roll, and Subrahmanyam (2002) [6] provide a comprehensive overview of market liquidity, emphasising its crucial role in shaping trading strategy performance. This paper highlights how liquidity conditions affect trade execution and price impact, ultimately affecting the performance and profitability of trading strategies. In addition to this, Chordia, Huh, and Subrahmanyam (2006) [7] investigate how factors such as liquidity and volatility influence trading volumes and patterns. Their findings demonstrate the importance of understanding market dynamics when designing effective trading strategies. Aldridge (2013) [8] offers practical insights into algorithmic trading strategies, highlighting the need for adaptability and responsiveness to changing market conditions to succeed. By analysing high-frequency trading systems, Aldridge yields valuable insights regarding how to leverage technology to enhance trading strategy performance.

Latency Dynamics

Moaellemi and Saglam (2013) [9] investigate how the presence of latency leads to trading frictions and using a model they created, quantify the cost of latency on transaction costs. This study shows that as the pace and latency of trading improves, the amount of money that can be saved also improves. Furthermore, Stoikov and Waeber (2016) [10] show that if trades are executed very quickly, in just 1 millisecond, they can save around one-third of the spread per share, capturing a larger portion of the price difference, ultimately resulting in cost savings.

Methodology

Environment

To simulate the stock exchange environment for this research, The Bristol Stock Exchange (BSE) made by Dave Cliff will be used. As previously stated, BSE is a "minimal simulation of a financial exchange running a limit order book (LOB) in a single tradable security" [5]. This simulation was written in Python and was specifically designed for research and educational purposes. To experiment with this exchange, we can use existing agents within the codebase (more information in next section) or create our own to experiment with and add to the market. In addition to this, BSE allows us to tinker with the market dynamics to investigate and explore how various factors affect trader performance.

Agents

Each trader that is created within the BSE environment is a subclass of the superclass Trader. Each trader has several attributes and functions that they override to perform their calculations and make their trades. A visualisation of the attributes available to each Trader can be seen in *Figure 1*. The subclass traders carry out their processes via various overridden methods from the superclass such as:

- getorder() – "An individual trader is chosen at random to issue its current response by calling the trader's getorder() method: this will either return the value None, signalling that the trader is not issuing an order at the current time, or it will return an order to be added to the LOB" [5].
- respond() – "Each trader is given the chance to update its internal values that affect its trading behaviour, via a call to the trader's respond() method" [5].
- bookkeep() – "If processing the order resulted in a trade, the traders do necessary book-keeping, updating their blotters, via a call to bookkeep()" [5].

```
self.ttype = ttype  # what type / strategy this trader is
self.tid = tid  # trader unique ID code
self.balance = balance  # money in the bank
self.params = params  # parameters/extras associated with this trader-type or individual trader.
self.blotter = []  # record of trades executed
self.blotter_length = 100  # maximum length of blotter
self.orders = []  # customer orders currently being worked (fixed at len=1 in BSE1.x)
self.n_quotes = 0  # number of quotes live on LOB
self.birthtime = time  # used when calculating age of a trader/strategy
self.profitpertime = 0  # profit per unit time
self.profit_mintime = 60  # minimum duration in seconds for calculating profitpertime
self.n_trades = 0  # how many trades has this trader done?
self.lastquote = None  # record of what its last quote was
```

Figure 1: Trader Superclass Attributes

A few trading agents have been pre-built into the BSE which will be briefly outlined below, but for a more detailed and thorough explanation I suggest reading the BSE paper [5]:

- Giveaway (GVWY) – No hope for making profit, it just gives deals away at the price set within orders.
- Zero Intelligence Constrained (ZIC) – Makes up random prices, but never at a loss.

- Shaver (SHVR) – Shaves a penny off the best price in attempt to always have the best bid or offer on the limit order book (LOB).
- Sniper (SNPR) – This trader lurks until time remaining < threshold of the trading session and gets increasingly more aggressive to try its best to land a trade.
- Zero Intelligence Plus (ZIP) – Uses machine learning techniques to alter the value of its margin in response to events in the market.

Technologies

The BSE is written in Python, which is what is used for all the experimentations and additions to the BSE codebase. After a trading session, BSE dumps the resulting data into csv files where data collection, manipulation, and visualisation is then performed using the Python libraries pandas, NumPy, and matplotlib. Pandas facilitates data manipulation and analysis and allows working with structured data to be fast, easy, and powerful. Numpy provides support for large, multi-dimensional arrays and matrices and provides a collection of mathematical functions to operate on these arrays. Matplotlib is widely used for generating high-quality plots, charts, histograms, and a range of other graphical representations of data.

Challenges

The initial challenge I was faced with during the implementation of this research project is the fact that I had little to no pre-existing knowledge surrounding finance. However, the realm of finance and trading was already an area which I was interested to delve into, which ultimately was the reason for the choosing of this area to research.

The second challenge I overcame was that BSE is quite a large codebase and I had to be able to understand it well to be able to confidently modify it to carry out the research required for this project. Alongside the highly useful lab sheet for this project, the BSE paper [5] was extremely informative and helpful when overcoming this challenge. In addition to this, the fact that the codebase is so well commented and organised really helped to understand what was happening, where, and why.

Experimental Setup

The technical implementation to gather, manipulate, and visualise all the data used in the following experiments can be viewed within the *processResults.py* file. In addition to this, each trading session is 864 seconds long, and each experiment carried out is run num_runs times and the results are averaged out over all the runs to obtain more accurate data. For all the experiments within this paper, every single experiment and data collection process is run with num_runs as 10. This number was found to be a good intersection between accuracy and time saving, as each experiment took considerable time to complete all the simulations, gather the data, average it all out, and plot the results.

Baseline Performance Comparison

By isolating traders, we can begin to gather insights about the behaviour of each trader which may help to inform future decisions and questions. To set up and conduct this experiment, each trader within the BSE (as listed in the *"Agents"* section above) is placed in the environment with an equal number of buyers and sellers of just that trader. This was implemented in the runBSE() function (line 2565:BSE.py) which is what's called to start and run the simulation. Within the function you

specify what traders will be involved in this trading session, and the quantity of each to be added. For example, *Figure 2* shows how you would implement 10 of each trader into the simulation.

```
buyers_spec = [('GVWY', 10), ('ZIC', 10), ('SHVR', 10), ('SNPR', 10), ('ZIP', 10)]
sellers_spec = buyers_spec
```

Figure 2: Implementing 10 of each trader into the trading session.

Pairwise Strategy Comparison

Implementing all the traders to compete in the same environment allows us to gather more accurate insights regarding how these strategies might perform in a real market. To gather this information, 10 buyers and 10 sellers of each trader was placed into the simulation and tasked to compete against each other. For some of the experiments, the number of buyers and sellers differed to explore a more realistic environment as the number of sellers and buyers in a real market is not likely to be equal. Manipulating the number of traders in the market is possible through manipulating the buyers_spec and sellers_spec lists in the runBSE() (line 2561:BSE.py) function. In addition to this, an insider trader was added to the final experimentation. This was accomplished by adding a subclass of the Trader class in BSE.py called Insider_Trader. The insider trader already knows the equilibrium of the market and uses this to his advantage. The code for this trader can be seen in the provided and modified BSE.py file.

Sensitivity Analysis and Adaptation to Changing Markets

These experiments investigate how the traders adapt to two areas, noise, and volatility.

To implement noise, I initially created a parameter in the runBSE function within BSE.py that takes in a noise level and passes it to the market_session function, where the noise is then added to all the trader's trades, this can be seen in *Figure 3*. This code adds the noise to each order price but ensures that it doesn't force the trader to make a loss.

```
# Noise implementation
if order is not None:
    noise_level = random.randint(-noise, noise)
    order.price += noise_level
    if order.otype == 'Ask' and order.price < traders[tid].orders[0].price:
        order.price = traders[tid].orders[0].price
    if order.otype == 'Bid' and order.price > traders[tid].orders[0].price:
        order.price = traders[tid].orders[0].price
```

Figure 3: Code for implementing noise into the market.

An array of noise levels is created and for each level in the array, the simulation is run num_runs times, and the data is collected and visualised to see how the traders are affected by different levels of noise.

To implement volatility, we experiment with continually gradual volatility, market shocks, and normal BSE behaviour. To implement this, a parameter was added to runBSE() which takes in a string value determining the type of volatility for the market. The code to implement volatility can be found at line 2503 within the BSE.py. Gradual volatility includes the use of tying the equilibrium

to a sine wave, and market shocks are investigated using different levels of both market drops, and increases.

Latency and Proximity Dynamics

This experiment was implemented purely using the BSE.py file, not the processResults.py, however, the code for it can be seen at line 2371. This experiment investigates how being closer to the physical exchange, or having a more favourable latency, increases your potential profits as a trader. The assumption here is that by controlling the probability of a trader making a trade, you can investigate how the trader's distance to the exchange, or network latency to the exchange's servers affects the trader's performance. This is due to the assumption that someone who lives closer to an exchange, or has more favourable latency, has a higher probability of making their trade first. To implement this, weighted probabilities are utilised, where a random trader is then chosen using these probabilities and asked to create its order. This means that by modifying trader_weights at line 2377, we can control who is more likely to get their trade in first and carry out our experiments.

Results

Baseline Performance Comparison

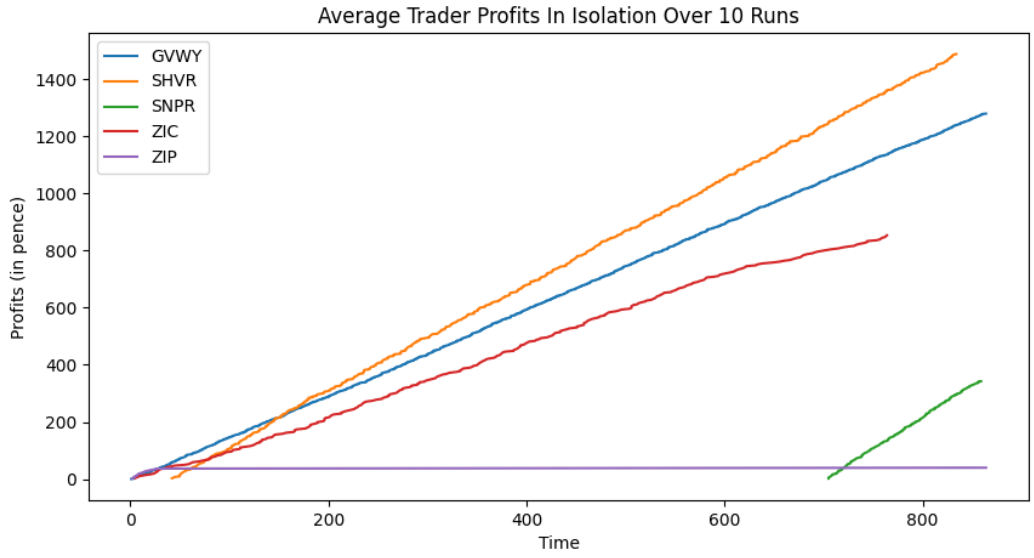

Figure 4: Graph showing trader profits in isolated environments.

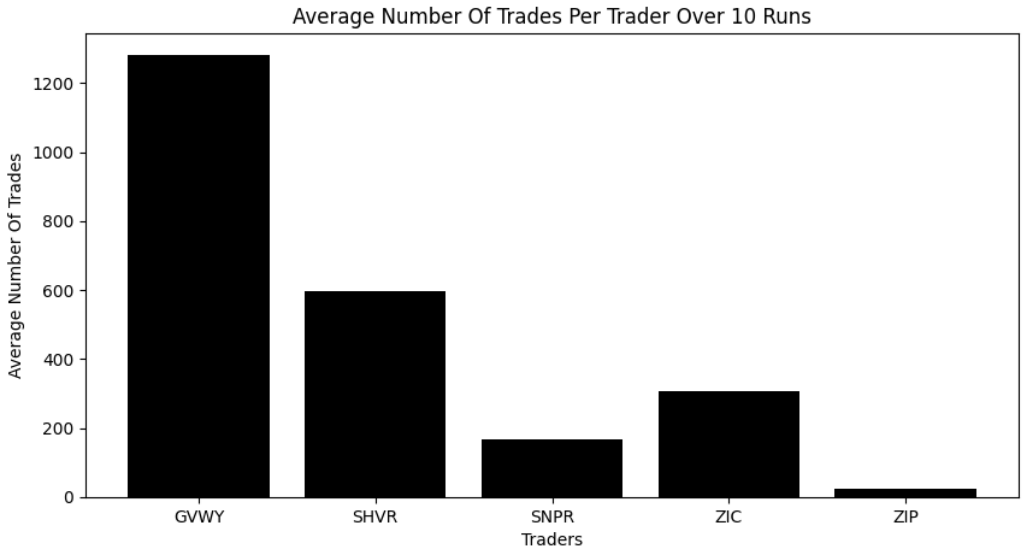

Figure 5: Graph showing average amount of trades per trader in isolated environments.

Pairwise Strategy Comparison

Figure 6: Graph showing trader profits in a pairwise environment over 10 runs.

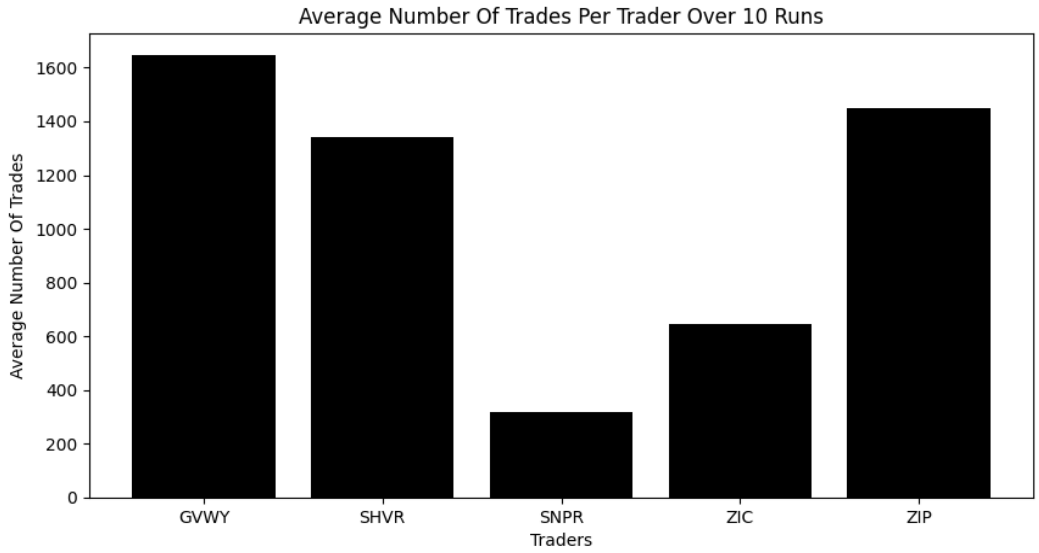

Figure 7: Graph showing number of trades per trader in a pairwise environment over 10 runs.

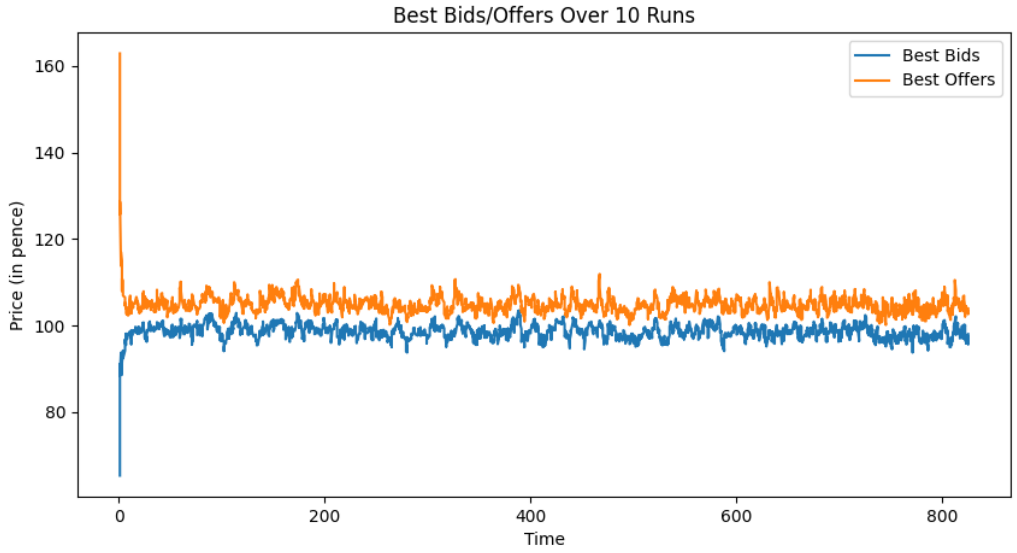

Figure 8: Graph showing the best bids and offers over time in a pairwise environment over 10 runs.

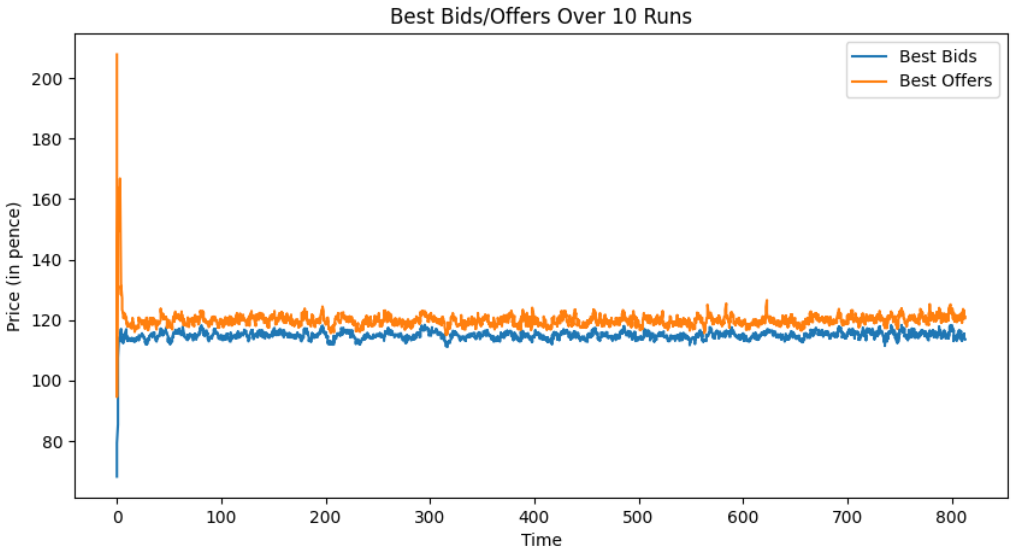

Figure 9: Graph showing the best bids and offers over time in a pairwise environment over 10 runs, with 20 buyers of each trader, and 10 sellers of each trader.

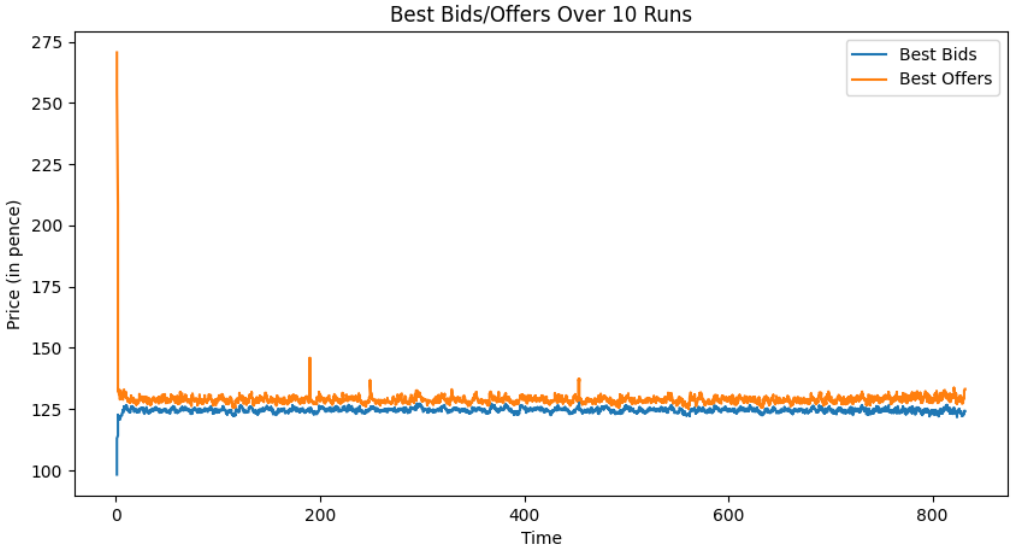

Figure 10: Graph showing the best bids and offers over time in a pairwise environment over 10 runs, with 30 buyers of each trader, and 10 sellers of each trader.

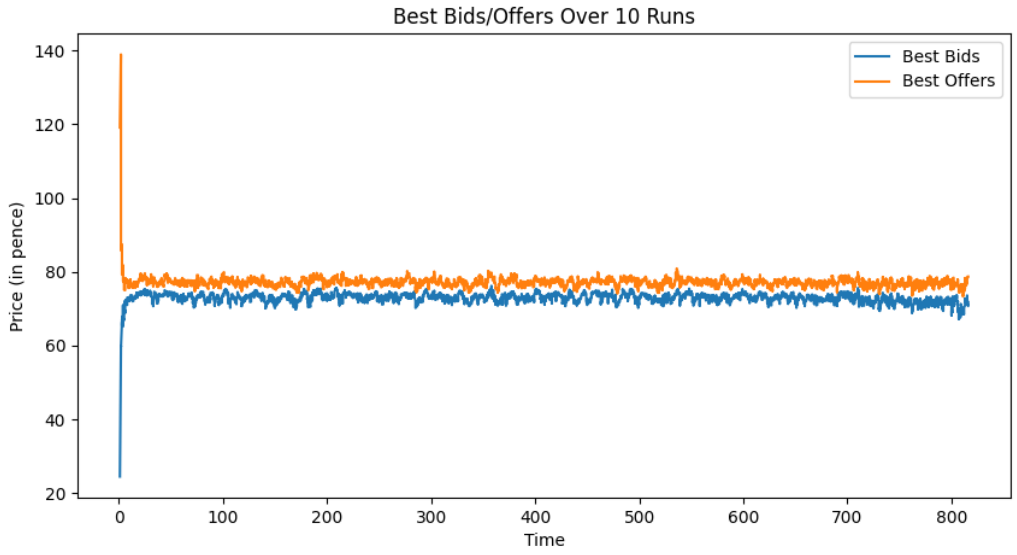

Figure 11: Graph showing the best bids and offers over time in a pairwise environment over 10 runs, with 10 buyers of each trader, and 30 sellers of each trader.

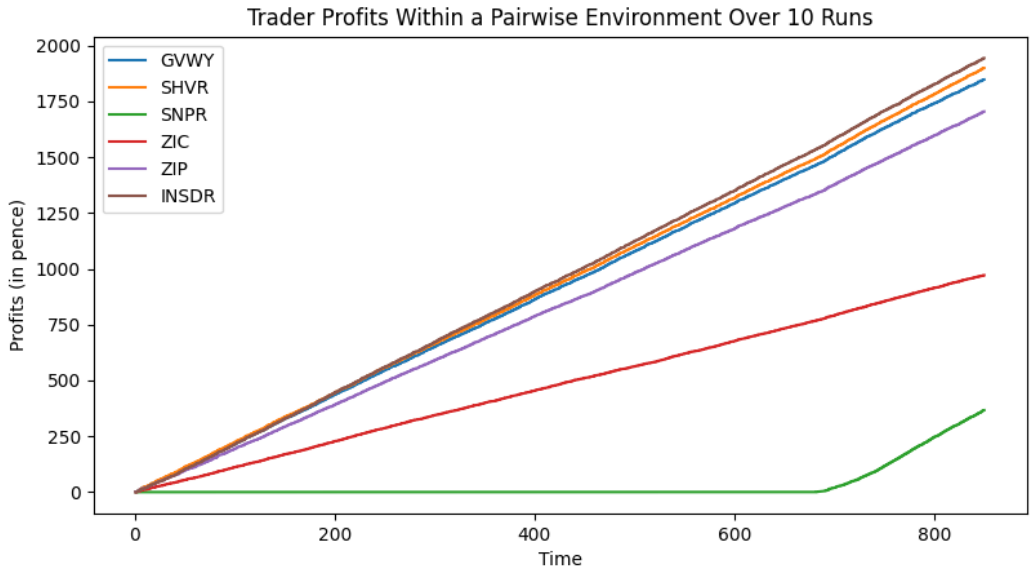

Figure 12: Graph showing trader profits in a pairwise environment over 10 runs, including an insider trader.

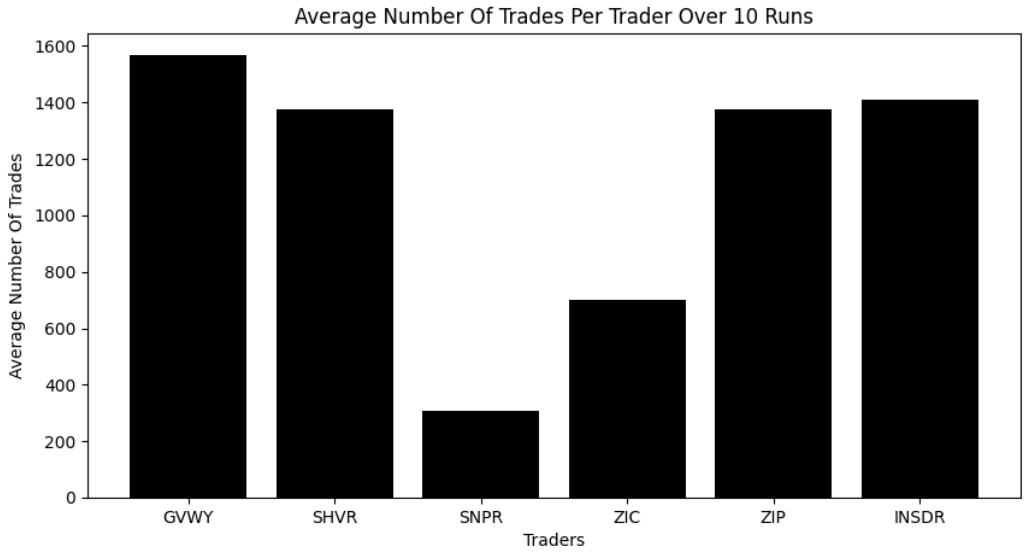

Figure 13: Graph showing number of trades per trader in a pairwise environment over 10 runs, including an insider trader.

Sensitivity Analysis and Adaptation to Changing Markets

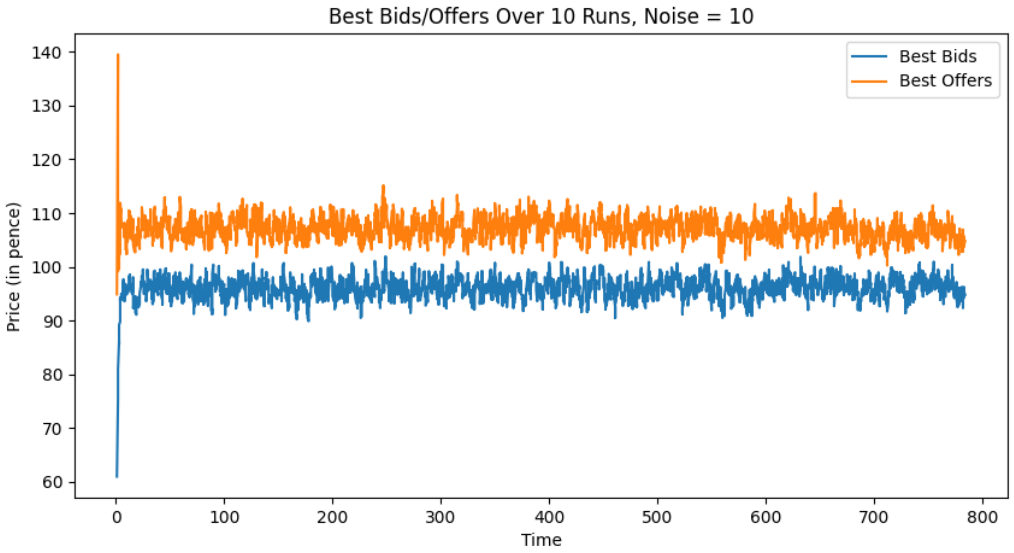

Figure 14: Graph showing the best bids and offers over time in a pairwise environment over 10 runs, with a noise level of +/-10.

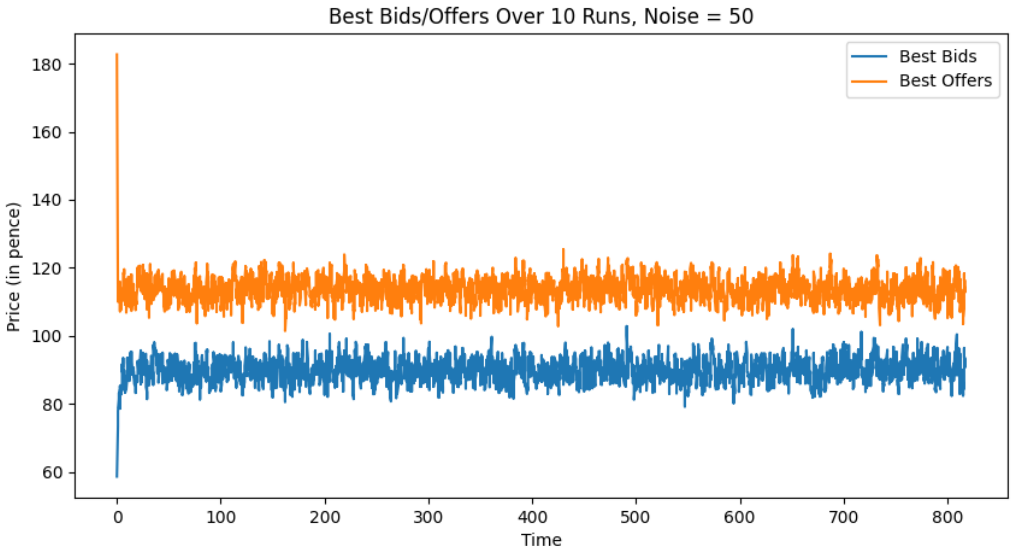

Figure 15: Graph showing the best bids and offers over time in a pairwise environment over 10 runs, with a noise level of +/-50.

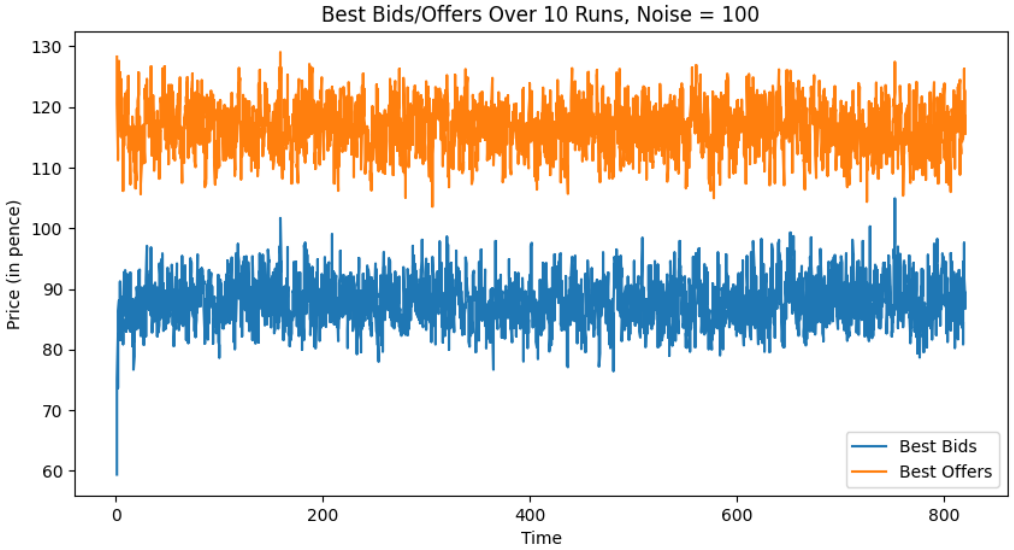

Figure 16: Graph showing the best bids and offers over time in a pairwise environment over 10 runs, with a noise level of +/-100.

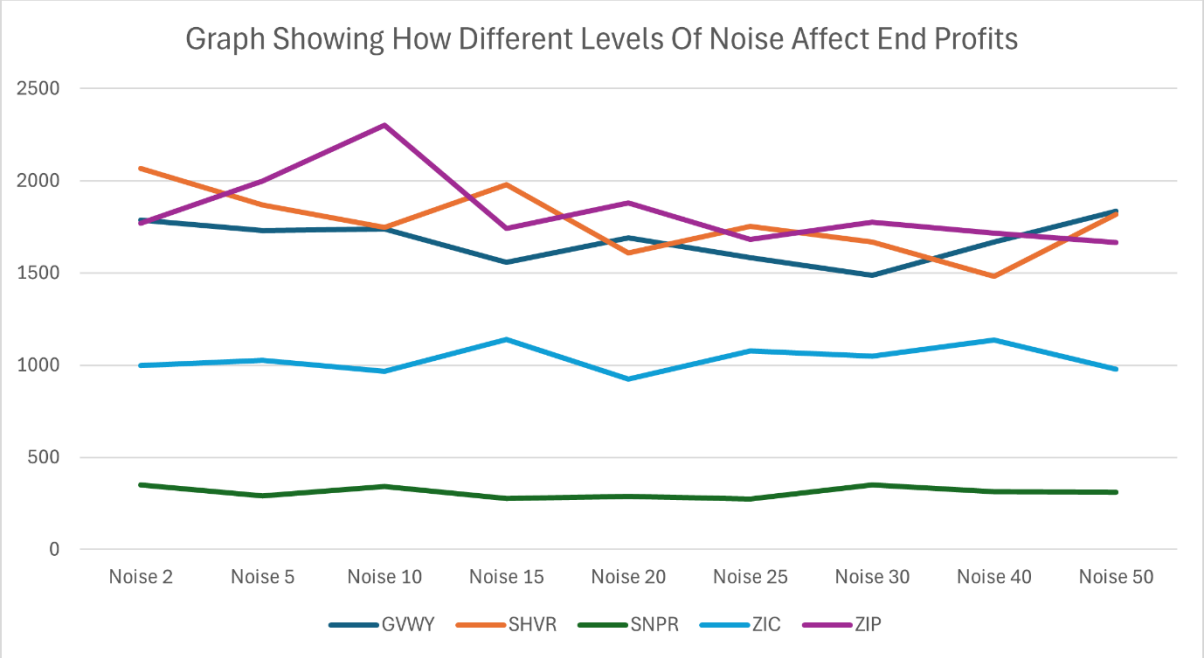

Figure 17: Graph showing how different levels of noise affect the end profits of each trader, with noise levels ranging from 2 to 50.

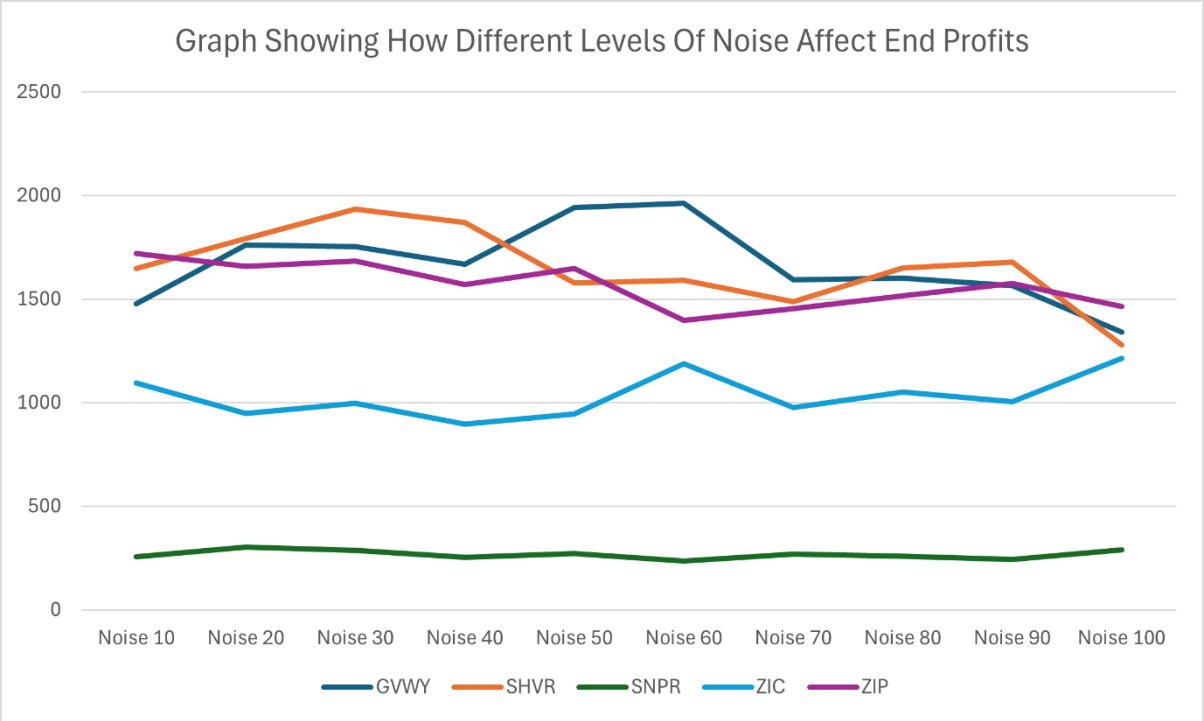

Figure 18: Graph showing how different levels of noise affect the end profits of each trader, with noise levels ranging from 10 to 100.

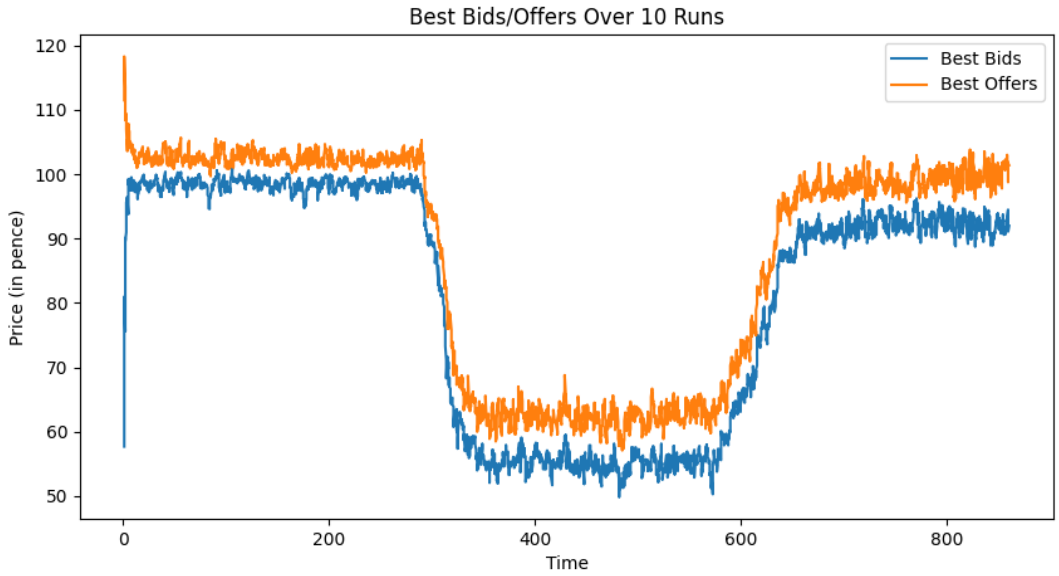

Figure 19: Graph showing the best bids and offers over time in a pairwise environment over 10 runs, with a market drop of 50%.

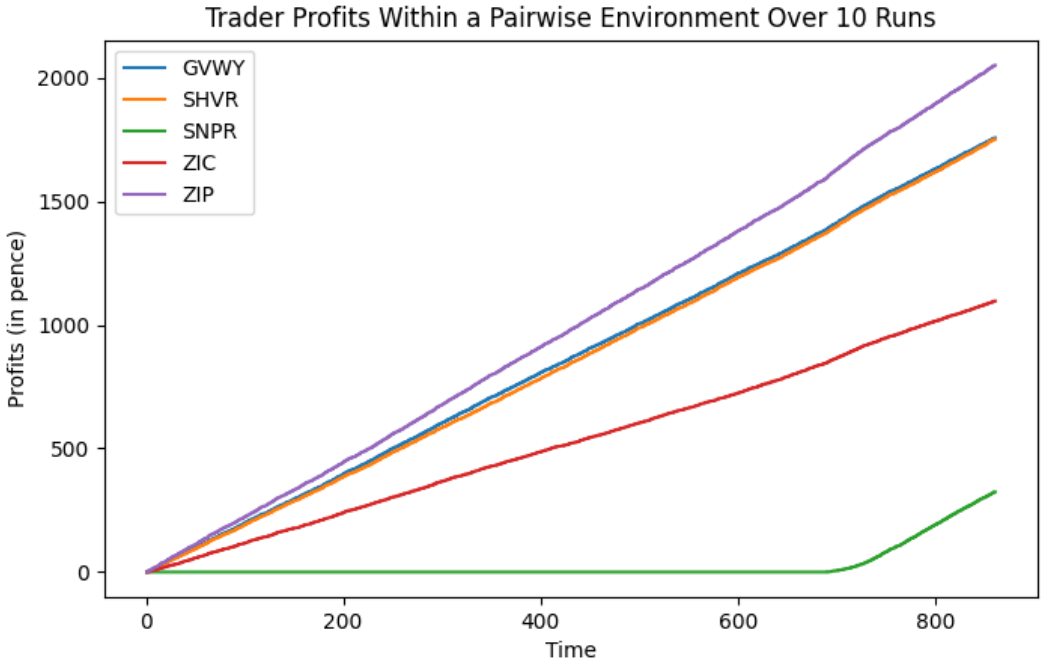

Figure 20: Graph showing trader profits in a pairwise environment over 10 runs, with a market drop of 50%.

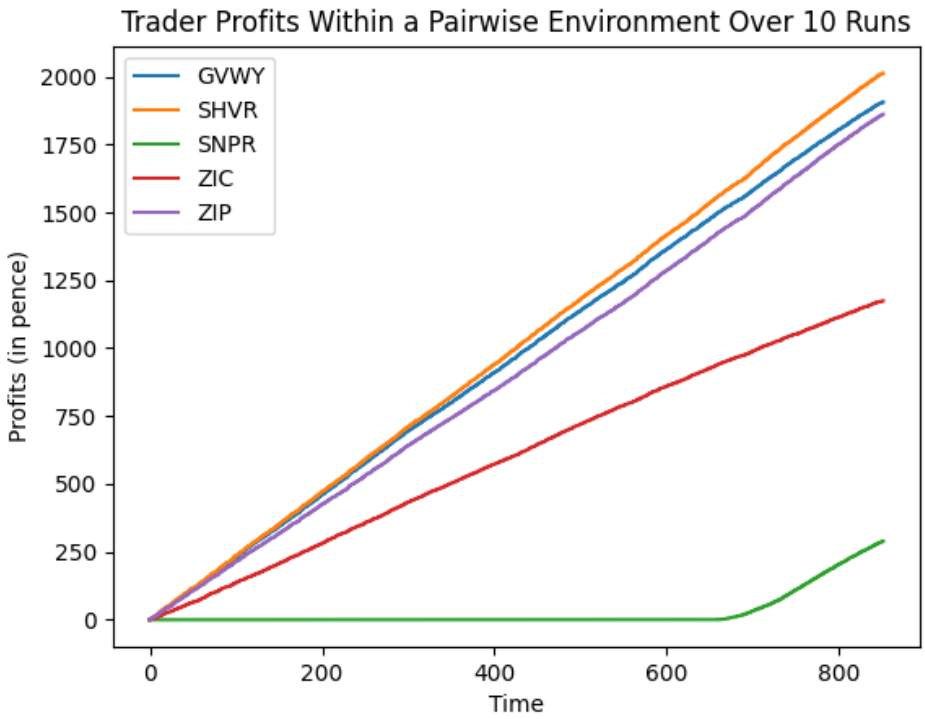

Figure 21: Graph showing trader profits in a pairwise environment over 10 runs, with a market increase of 50%.

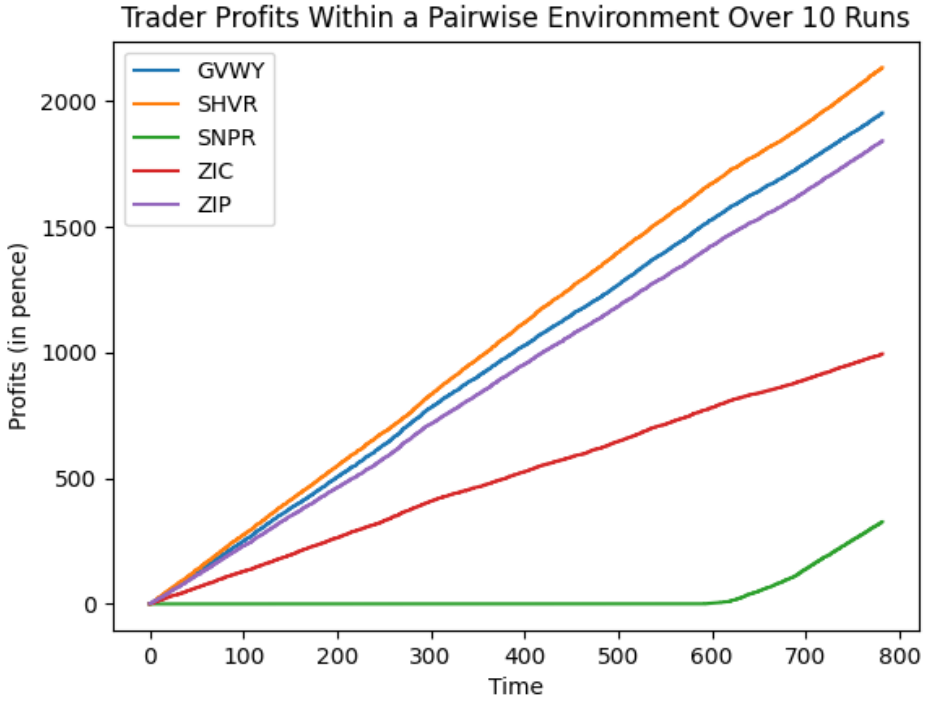

Figure 22: Graph showing trader profits in a pairwise environment over 10 runs, with a market increase of 100%.

Figure 23: Graph showing the best bids and offers over time in a pairwise environment over 10 runs, with a market volatility controlled by a sine wave function.

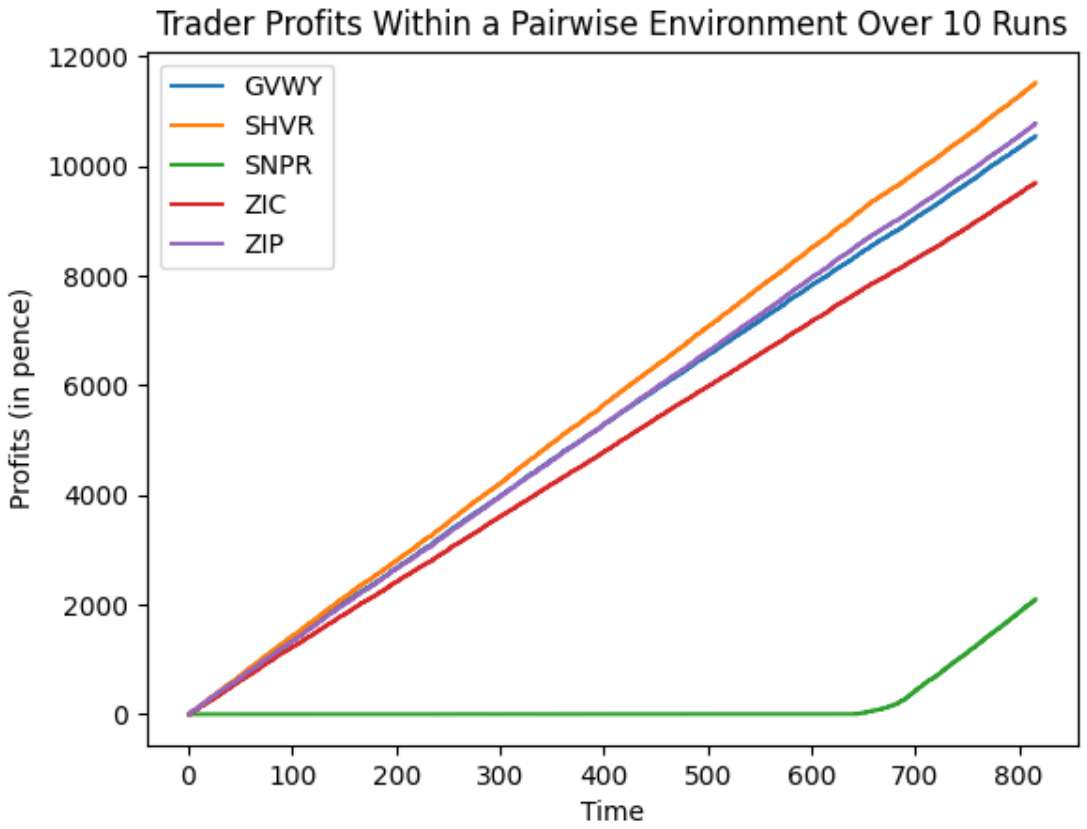

Figure 24: Graph showing trader profits in a pairwise environment over 10 runs, with a market volatility controlled using a sine wave function.

Latency and Proximity Dynamics

Trader	Equal Distances	Profit when closest to exchange	Percentage Increase (%)
GVWY	1697	1855	9.31
SHVR	2054	1987	-3.26
SNPR	296	340	14.86
ZIC	1078	1551	43.88
ZIP	1805	2002	10.91

Figure 25: Table showing profits (in pence) when the traders have equal probability of getting a trade vs when they have double the probability than the rest of the traders (closest to the exchange).

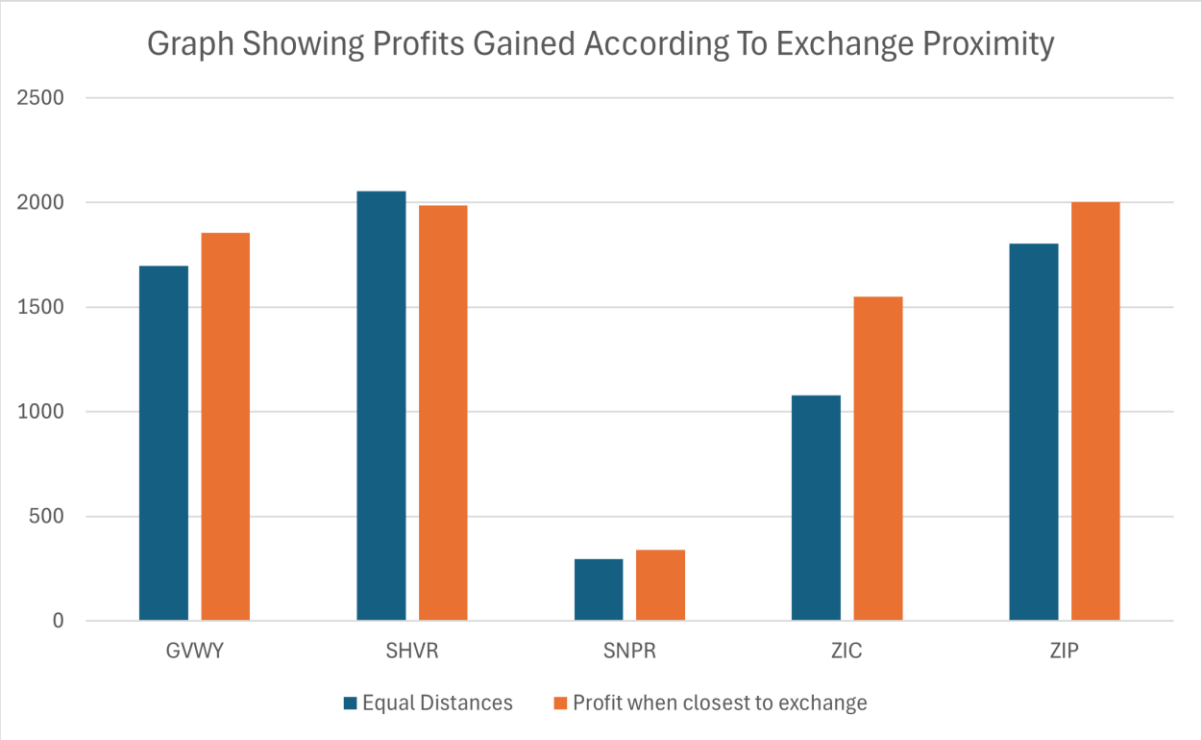

Figure 26: Graph showing profits (in pence) when the traders have equal probability of getting a trade vs when they have double the probability than the rest of the traders (closest to exchange).

Discussion

Baseline Performance Comparison

In *Figure 4* we begin to see some initial behaviours such as SNPR lurking until the end to profit as expected, SHVR being the most profitable in an isolated environment, and ZIP performing very poorly. ZIP relies on other trades to calculate its strategy and perform trades, but when ZIP traders are put against ZIP traders, not much appears to happen as there are no trades to base their order against, and little profit is made. A possible explanation for the profitability of each trader may lay in *Figure 5*, where the order of least trades executed to most trades executed perfectly matches the order of least profits to highest profits in *Figure 4*. This suggests that the traders that can execute more trades, will perform better. We can also observe the fact that while the giveaway trader was designed to give deals away and not profit, it accidentally ends up profiting anyway, and proves itself to be relatively successful against expectations.

Pairwise Strategy Comparison

In *Figure 6,* where all the different traders are implemented into the same trading sessions, we begin to see how their performance compares. Sniper proves to be least profitable but remains its previously observed behaviour of waiting until the end of the market to begin "sniping" profits. However, the sniper is still profitable, which can be seen as a win for sniper. ZIC makes some profit, but only around 50% of the rest of the traders' profits, except sniper. Giveaway, Shaver, and ZIP traders all position very close within each other and are earning good profits, suggesting that these three are the most effective and profitable traders out of the group. Again, in *Figure 7* we see a direct correlation between profitability and frequency of traders per trader, suggesting that a higher number of trades implies more profit. When investigating the equilibrium price of each trading session in a pairwise environment with equal buyers and sellers, we can observe that the market settles on the average price between the price demanded by sellers and buyers at around 100 pence (£1), the estimated value of the security, as seen in *Figure 8*. However, in *Figures 9, 10, and 11*, we can observe that by changing the quantity of the buyers and sellers within the market, making them different, we can directly affect the equilibrium price. As seen in *Figures 9 and 10*, when there are more buyers than sellers in the market, the equilibrium price rises. On the contrary, as seen in *Figure 11*, when there are more sellers than buyers in the market, the equilibrium price drops. However, no changes in profitability between traders was observed when modifying the quantity of buyers and sellers, suggesting that all the traders are adaptable to changes in trader populations and further solidifying each trader's spot in the rankings of most profitable trading strategy amongst the group as previously seen in *Figure 6*. Jaffe (1974) discovered that "the returns to the insiders was on average 9.5% greater than the return to the stock market as a whole" [11]. This is also evident in our investigation when we introduce the insider trader. *Figure 12* demonstrates that insider traders have the potential to be the most profitable, as expected, and hence why insider trading is illegal. The insider trader also breaks the correlation between profitability and trade frequency, acquiring the most profit with the second most trades as seen in *Figure 13,* also suggesting that the insider trader has a higher profit per trade than the Giveaway and Shaver traders due to its obtained information.

Sensitivity Analysis and Adaptation to Changing Markets

As the level of noise introduced into the market increases, the bid-ask spread appears to grow larger as seen in the difference between a noise level of +/-10 in *Figure 14*, +/-50 in *Figure 15*, and +/- 100 in *Figure 16*. In addition to this, as the noise level increases, we also observe an increase in bid and offer variability, as expected. *Figure 17* demonstrates that the level of noise introduced into the market doesn't appear to have any correlation with end profits, which was unexpected. Due to this surprise, it was again tested with a different range of noise levels as seen in *Figure 18*, where no correlation could be drawn again. This investigation suggests that each of these traders is resistant, adaptable, and performs well within a noisy environment, giving them extra points for performance and adaptability.

When introducing a market drop of 50% during the trading session as seen in *Figure 19 and Figure 20*, we can observe different results than we would see in a normal trading session such as the one in *Figure 6*. It appears that ZIP is the most adaptable to market drops, as it overtakes both Shaver and Giveaway trader during this shock, who both seem to have taken a hit in terms of end profits due to the market shock, where ZIP has increased its profits. However, during market shocks where the price of the security increases by 50% as seen in *Figure 21*, Shaver appears to be the most profitable. This is also the case during a market increase of 100%, where Shaver takes the lead even more as shown in *Figure 22*. However, both *Figure 21* and *Figure 22* hardly differ

from the baseline pairwise strategy comparison, suggesting that the traders neither thrive nor suffer from upwards market shocks, but their performance in fact remains similar. When tying the equilibrium to a sine wave, as seen in *Figure 23*, we can investigate how the traders adapt and perform in a market which is often seen in the real world, where markets are consistently up and down, like a wave. As seen in *Figure 24*, the graph again hardly differs from the baseline pairwise strategy comparison, except for one trader, ZIC. ZIC appears to very adaptable to this continual rise and fall off pattern, and despite originally being at nearly 50% of the top 3 traders' profits as seen in *Figure 6*, it is now grouped together with the top 3. Another observation lays within the y-axis, where the profits appear to be around 6 times higher than the original comparison in *Figure 6*. This suggests that while ZIC is the most adaptable to this continual rise and fall off pattern, all the traders seem to adapt to it effectively and massively boost their profits, giving all the traders, particularly ZIC, extra points for adaptability to changing markets.

Latency and Proximity Dynamics

The effect of being the closest trader to the exchange has on each traders' profits can be seen in *Figure 25*. Here we see that Shaver doesn't appear to benefit from being closer to the exchange, and in fact suffered a small percentage decrease in end profits. Giveaway, Sniper, and ZIP reacted positively to being closer to the exchange and all three received decent profit increases. ZIC showed a massive profit increase from being the closest to the exchange with an increase of 43%. These results are also visualised in *Figure 26*, where we can see how exchange proximity affects our traders' profits. In 4 of the 5 traders, we observe that a higher probability of getting a trade before the other traders (better latency) leads to reduced costs, just as was shown in the study by Moaellemi and Ciamac (2013) [9].

Conclusion

In conclusion, this study has yielded valuable insights into the performance and adaptability of the various trading strategies provided within the Bristol Stock Exchange (BSE) environment. The project's objectives were effectively addressed through an analysis of baseline performance, pairwise strategy comparisons, sensitivity analysis, market adaptability, and examination of latency and proximity dynamics.

Initially, by conducting baseline performance comparisons, we were able to evaluate the individual trading strategies in isolation. By doing so, we observed some behaviour of each trading agent such as sniper lurking until the end. The results highlighted a correlation between trade frequency and profitability, suggesting within BSE more trades generally yield more profit, and there's less effect of profit per trade.

Secondly, carrying out pairwise comparisons allows us to assess the relative strengths and weaknesses of each of the trading agents and gather more realistic information about how these agents may perform in a real environment. We observed a variation of end profits among the traders such as sniper being the least profitable and shaver the most. When introducing changing market conditions such as a changing population of sellers and buyers, we observe solid adaptability from all the traders due to no produced effect on profitability. Furthermore, the introduction of the insider trader introduces the highest earner out of all the traders, emphasising the ethical and legal implications of insider trading.

Thirdly, sensitivity analysis provided insights into how the trading strategies respond to market conditions and parameters. For example, by introducing noise into the market, we observe

changes in bid and offer variability, however, the traders all presented excellent adaptability to the noise as no observed effect of introducing a range of noise levels was observed.

Furthermore, the investigation into trader adaptability demonstrated each trader's ability to respond to changing market dynamics, including various types of volatility and sudden market shocks. This experiment discovered that the ZIP trader is the most adaptable to market drops, and Shaver is the most adaptable to upward market spikes. When introducing realistic volatility, we observe that the ZIC trader is the most adaptable. Although, in this experiment, while ZIC was the most adaptable, all the trader displayed excellent levels of adaptability, and all increased their profits roughly six-fold.

Finally, our examination of latency and proximity dynamics sheds light on the impact of proximity and latency on trader profitability. Proximity benefited four out of five of the traders, with ZIC benefitting massively. However, shaver suffered a loss. This could perhaps be because Shaver likes to see the other trades first before shaving off the price. These findings align with existing literature regarding the relationship between latency and proximity dynamics and trading performance.

With more time and space within this report, I would like to further investigate or propose future research into developing effective traders for the BSE by learning from the strengths and weaknesses of each trader discovered within this study. Each of the trader's investigated yielded solid performance and adaptability to changing markets. However, in the modern day, we have much more powerful technology that could be utilised to create a more powerful agent such as machine learning. For example, deep learning models have shown promise in predicting future prices with 72% accuracy [12]. In addition to this, while the BSE can be used to simulate realism, there is nothing more real than deploying these strategies in live trading environments and analysing their performance to gather further insights for future innovation.

Overall, by addressing this project's objectives, this study contributes to a higher understanding of various trading strategies and yields insights to fuel future innovation within algorithmic trading.

Reflection

Successes

The first success is that through the experiments, results, and discussions using a range of performance analytics and manipulated market factors that influence the traders' performance, the question was successfully answered. For the first section of the question, we evaluated the performance of diverse trading strategies effectively, analysing profitability, trade frequency, some behaviour patterns, and how they compete against other strategies. By answering the second part of the question we evaluated how various market factors affect the traders' performance and how well they adapt by introducing volatility, noise, and latency.

The second success is that a lot of testing was carried out. The testing section was very time heavy, however, it has ensured solid accuracy of results with a wide variety of results for each investigation to help answer the question at hand.

The final success was making the connections between the literature regarding insider trading and latency/proximity. The literature regarding insider trading helped to support the findings regarding the implemented insider trader within this paper. In addition to this, our

latency/proximity findings were also supported by the literature found. This allowed us to further build on other research and contribute to higher understanding.

Limitations

One limitation of this research is that investigations within a simulated environment cannot be 100% assumed to apply to the real world and real markets. This could potentially render the findings within this research incomparable to real world scenarios and may only be applicable to the BSE.

The second limitation is that only six types of traders were investigated. There are many trading strategies that exist and would exist in a real market, potentially rendering this research quite limited in terms of findings for algorithmic trading in general.

The final limitation is that this simulation only investigates the trading of a single security with the quantity of one for each trade. This is very unrealistic when compared with real world markets. In future, it would be more accurate and realistic to allow the traders to buy and sell multiple quantities of the security for more realism and allow for a more accurate comparison with real markets and trading.

References

1. *Algorithmic trading market - growth, trends, COVID-19 impact, and forecasts (2022 - 2027)* (2022) *Yahoo! Finance*. Available at: https://finance.yahoo.com/news/algorithmic-trading-market-growth-trends-112400870.html (Accessed: 22 April 2024).
2. Groette, O. (2024) *What percentage of trading is algorithmic? (algo trading market statistics)*, *Quantified Strategies*. Available at: https://www.quantifiedstrategies.com/what-percentage-of-trading-is-algorithmic/ (Accessed: 22 April 2024).
3. *What percentage of trading is algorithmic? (algo trading volume analysis)* (2024) *THE ROBUST TRADER*. Available at: https://therobusttrader.com/what-percentage-of-trading-is-algorithmic/ (Accessed: 22 April 2024).
4. Cliff, D. (no date) *Davecliff/BristolStockExchange: BSE is a simple minimal simulation of a limit order book Financial Exchange*, *GitHub*. Available at: https://github.com/davecliff/BristolStockExchange (Accessed: 22 April 2024).
5. Cliff, D. (2018). BSE: A Minimal Simulation of a Limit-Order-Book Stock Exchange. In M. Affenzeller, et al. (Eds.), Proceedings 30th European Modelling and Simulation Symposium (EMSS 2018), pp. 194-203. DIME University of Genoa.
6. Chordia, T., Roll, R. and Subrahmanyam, A. (2001), Market Liquidity and Trading Activity. The Journal of Finance, 56: 501-530. https://doi.org/10.1111/0022-1082.00335
7. Tarun Chordia, Sahn-Wook Huh, Avanidhar Subrahmanyam, The Cross-Section of Expected Trading Activity, *The Review of Financial Studies*, Volume 20, Issue 3, May 2007, Pages 709–740
8. Aldridge, I. (2013) *High-frequency trading: A practical guide to algorithmic strategies and trading systems*. Hoboken: Wiley.
9. Moallemi, C., Saglam, M., 2013. OR Forum—The Cost of Latency in High-Frequency Trading, *Operations Research 2013* 61:55, 1070-1086 https://doi.org/10.1287/opre.2013.1165

10. Stoikov, S. and Waeber, R. (2016) ‘Reducing transaction costs with low-latency trading algorithms’, *Quantitative Finance*, 16(9), pp. 1445–1451. doi: 10.1080/14697688.2016.1151926.
11. Jaffe, J. F. (1974). Special Information and Insider Trading. *The Journal of Business*, *47*(3), 410–428. http://www.jstor.org/stable/2352458
12. M. U. Gudelek, S. A. Boluk and A. M. Ozbayoglu, "A deep learning based stock trading model with 2-D CNN trend detection," 2017 IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, 2017, pp. 1-8, doi: 10.1109/SSCI.2017.8285188.