Trading Agent Behaviour and Performance

Trading Agent Behaviour and Performance

Bristol Stock Exchange: An Experimental Study

Andris Kokins

Computer Science, University of Nottingham, https://github.com/andriskokins/BristolStockExchange

CCS CONCEPTS • Computing methodologies~Artificial intelligence~Distributed artificial intelligence~Intelligent agents • Social and professional topics~Professional topics~Computing and business~Economic impact • Applied computing~Law, social and behavioral sciences~Economics

Additional Keywords and Phrases: Stock Market, Limit-Order-Book, Algorithmic Trading, Market Simulation, Multi-agent System

1 INTRODUCTION

Trading of goods has been around in the human society for hundreds of years, a common place where people can buy and sell at such marketplace. Haggling is a fundamentally basic concept but over time it has become more complex, the simplicity of the seller wanting to sell for the highest price and the buyer wanting to purchase for the lowest price, is now an intricate system known as the stock market. Whole careers and industries were built up on trading stocks on the market.

As the name suggests, the stock market trades stocks, a stock can be anything from commodities (e.g., gold, oil, corn), tradable debt contracts (e.g., government bonds, known as gilts in the UK and treasury bills in the US), derivative contracts (e.g., value of an asset, group of assets or a benchmark), equities (e.g., ownership or stake of a company, be it private or publicly traded).

The way these stocks are traded in the world's largest exchanges like New York Stock Exchange (NYSE) or London Stock Exchange (LSE), is by having a seller announce an offer for an asset they own, meanwhile a buyer can announce a bid for an asset they want to purchase. At any time, a seller can accept a buyer's bid, and a buyer can accept a seller's offer. This type of bidding auction is known as *Continuous Double Auction (CDA)*.

A seller's desired selling price for an asset (e.g., a share) is called an 'ask'. Conversely, a buyer's stated price to purchase an asset is called a 'bid. Each of these types of trades are tracked in a Limit-Order-Book (LOB), this contains a list of all the asks (sell orders) and bids (purchase orders), these are all the live order that have not yet been fulfilled or cancelled [1]. The ask (sell) book would show the number of shares up for sale and their price, with the cheapest at the top and the rest in ascending order. Similarly, the bid (purchase) book shows the number of shares someone wants to purchase along with a price, note this time it's the highest price at the top and rest are in descending order. The delta between the best ask (lowest selling) price and the best bid (highest buying) price is called the bid-ask spread. The smaller the spread the more

buyers and sellers are willing to trade, in financial terms that means the market has high liquidity, whereas a high bid-ask spread means buyers and seller are less likely to trade so a low liquidity ratio.

2 ENVIRONMENT

This experiment will be using the Bristol Stock Exchange (BSE) developed by Dave Cliff [2] as the simulation environment. BSE is a minimal simulation of a financial market using a limit order book (LOB). It provides what is known in finance as Level 2 market data. There are a total of 3 levels of market data. Level 1 is the most basic level. It typically displays the current best bid price (highest a price a buyer is willing to pay), the best ask price (lowest price a seller is willing to accept), and the volume available at these prices. Level 2 provides more depth, it shows the full LOB, displaying a list of all current bid and ask orders at various prices. Level 3 is the most comprehensive level; this is typically only available to registered market makers (firm or individual that actively buys and sells shares and makes money from the bid-ask spread).

In real exchanges, a trader can choose from a large variety of shares, and they can purchase any amount they want e.g. 0.01 up to hundreds or thousands of a single share. Current implementation of BSE allows only purchasing and selling a single type of share in single dominations. While human investors rather not put all their eggs into one basket, by investing into a variety of stocks to spread the risk of their portfolio, however this can be difficult to model and is out of scope for this experiment [3].

2.1 BSE Graphical User Interface

The BSE provides an overwhelming amount of detail and control over the different parameters all in a single python file. This can be simplified down to an easy-to-use UI with only the essential parameters, as seen in Figure 1. However, the UI can be easily expanded to support the full feature set of the BSE.

The graphical user interface (GUI) is split into 3 main sections; simulation, traders and output shown as tabs at the top of the main user interface (UI) just below the title. The first tab, *Simulation*, this controls how long the simulation can run for. The first parameter is the number of days, by default this is set to 0.01, this is due to BSE running only on a single CPU thread and increasing this parameter will exponentially increase the duration of the simulation from a few seconds to hours or even days. However, there is a publicly available version of the BSE online that supports multi-threading, which could be also adapted to use this GUI but is out of the scope for this experiment [4]. The *Traders* tab allows the user to configure the number and type of buyer and seller agents. Finally, the *Output* tab allows the user to configure what data or rather the CSV file the BSE simulation should output, by default only the average balances are saved as this is used to process trader's performance.

At the bottom of the GUI there's a *Reset to Defaults* button for the user's convenience to quickly reset the settings after changes. The UI also features an *Apply Settings* button to save the current settings to a python file called *bse_config.py* [1], this file will be used by *BSE.py* as the configuration for the simulation.

1 Note: bse_config.py file cannot be edited to change settings, as the GUI will overwrite it each time the simulation is started via the GUI

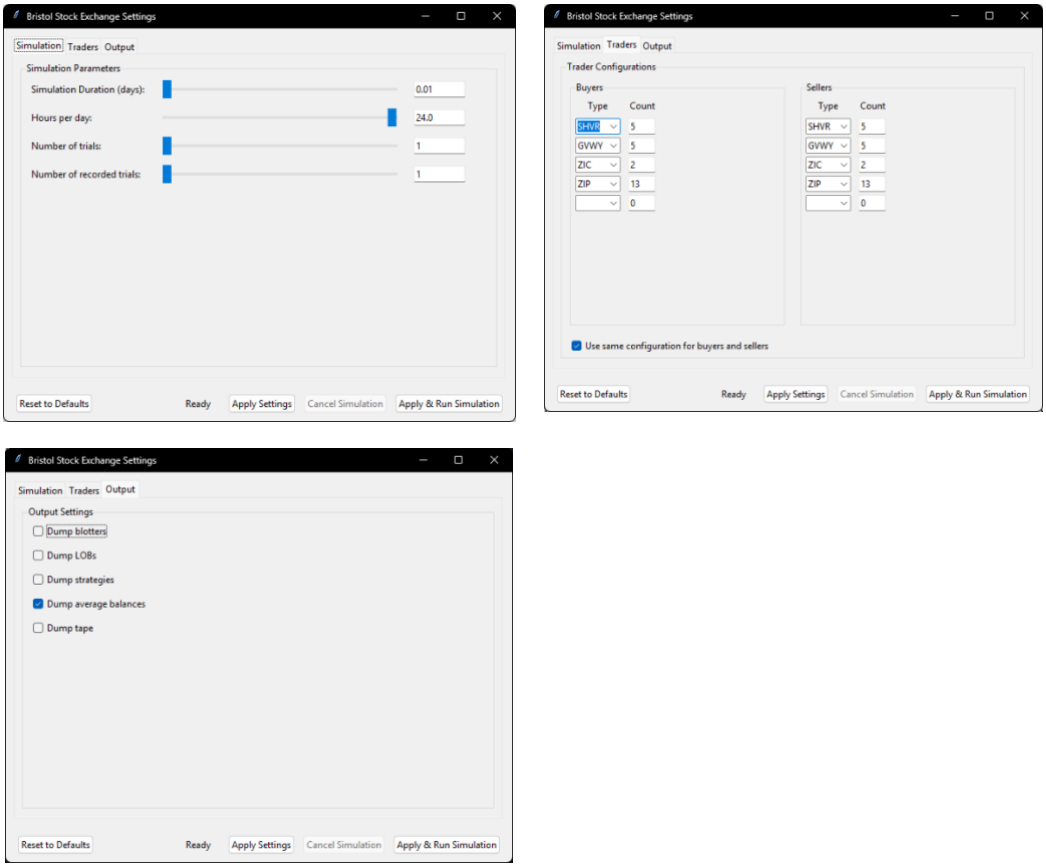


Figure 1: BSE Settings graphical user interface (GUI), used to configure simulation parameters. The three panes show the (top-left) "Simulation" tab for general experiment settings like duration and trials, the (top-right) "Traders" tab for specifying the type and count of buyer and seller agents, and the (bottom-left) "Output" tab for selecting various data dumps.

3 AGENTS

For a simulation to be somewhat useful, all the agents need to heterogenous (e.g., different trading strategies), otherwise the market can just be represented by a single agent and there won't be any interesting dynamics. A trading strategy can be implemented using a variety of different techniques, simple function, probabilistic, neural network and classifier. BSE default trading bots do not implement any artificial intelligence (AI) trading strategies but rather rule based. All the agents in BSE will have same access level of information via the LOB. This type of information is called endogenous, where the agent has access to current and previous trading price. Exogenous information is the second type, these are external events that could affect the price of stock e.g., a trade war, GDP growth, news [3].

4 EXPRIMENTS

For the following experiments five BSE agents will be used - Shaver (SHVR), Giveaway (GVWY), Sniper (SNPR), Zero Intelligence Constrained (ZIC) and Zero Intelligence Plus (ZIP), these will serve as control variables. The simulation will explore four population sizes for these agents: 5, 15, 50, and 100 in total. Testing each possible combination would

equate to n^k where n is the number of factors (agents) and k is the number of levels is $4^5 = 1024$. To reduce the number experiments to a reasonable amount, Latin Hypercube Sampling (LHS) will be used to generate 10 unique runs as shown in Table 1, where each row represents one run.

Each experiment was run on the same machine, running Microsoft Windows 11 version 24H2, with an AMD Ryzen 5700X3D 8 core processor running @3Ghz and using Python version 3.12.

Table 1: Latin Hypercube Sampling of Agent Population Distributions for Market Simulation

Run	SHVR	GVWY	SNPR	ZIC	ZIP
1	50	15	100	15	100
2	50	50	15	100	5
3	15	15	15	5	100
4	100	100	5	50	50
5	50	50	5	5	15
6	100	15	50	100	5
7	5	100	50	5	50
8	5	5	50	15	15
9	15	5	15	50	50
10	15	50	100	50	5

4.1 Impact of Agent Population Dynamics on Trading Performance

The aim of the first experiment is to evaluate the performance of the trading agents, and the effects on the market by varying the trading agent population, as mentioned above. A total of 10 runs was executed as per the LHS configuration, a sample of results can be found on page 9. All the run's statistics were then averaged to eliminate any outliers, and the data was consolidated into a box plot as seen in Figure 2.

4.1.1Results

As shown in Figure 2, each trading agent achieved a minimum of 0 profit but not negative, in hindsight this is somewhat redundant information was each agent starts the market session with 0 profit, therefore no interesting information can be interpolated from this data. However, it is interesting to note, that min is not below negative, now this could be due to the way the BSE fundamentally works, or due to current BSE settings, as the UI does not go extensively into detailed control of each BSE's settings, it is hard to determine the cause of this so this value can be ignored and shift focus on the other statistics.

The second major point is that SHVR agent was the clear winner across the board, the second best was GVWY agent. SHVR's average profit was 12.6% higher than GVWY's with 793 profit and an even better maximum, being 14.3% higher. This lines up with previous research into trading agents that found SHVR to be very dominant agent, in addition to this, the box plot lines up with the dominance hierarchy of SHV>GVWY>ZIP>ZIC [5]. Lastly, SNPR seems to show poor performance with very low profit across the board and the max being only 276 which is 37.4% below SHVR's 25th percentile. However, this doesn't mean the SNPR is a poor trading agent. The SNPR agent works by lurking in the background and waiting until the market is near closing time and if the conditions are right, i.e. bid-ask spread is narrow it will begin trading, so obviously it has less trading time in total but this doesn't mean the performance is as bad as Figure 2 suggests, because looking at Figure 3, SNPR agent starts trading only towards the end but importantly the rate at which its profits rise are quicker than ZIC's. Given this new information, we could assume SNPR was hindered due to the current

BSE configuration, this could be mitigated by applying an LHS to test different BSE configurations, we could assume that making the market sessions shorter and increase how long the simulation runs could greatly increase SNPR's profit.

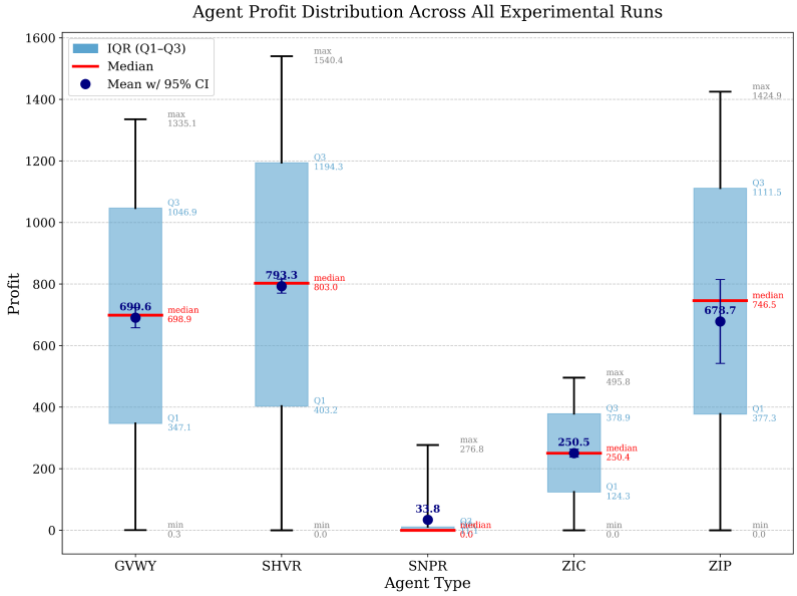


Figure 2: Box blot summarising results from 5 runs for experiment 1

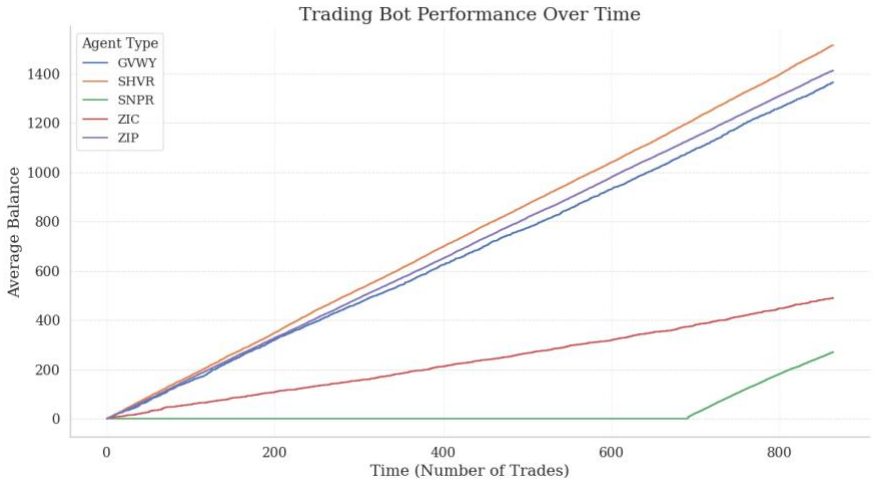


Figure 3: Line chart showing performance of 5 agents over time

4.2 Effects of Supply and Demand Disparities

For this experiment, the focus will be on the ratio of sellers to buyers. **Error! Reference source not found.** used a b alanced ratio of 1:1 and it focused purely on the performance of the agents themselves. Here, the aim is to see how varying the seller-to-buyer ratio impacts both market behaviour (such as equilibrium price) and the performance of the agents.

Basic economics suggests that prices rise when demand exceeds supply and fall when supply exceeds demand, so the expectation is that the equilibrium price will increase when there are more buyers than sellers, and decrease when there are more sellers than buyers.

To keep the experiment to a reasonable size, a subset of runs will be used from LHS configuration above. Since SHVR was the best performing agent in the last experiment, the base LHS runs will be focused on SHVR, run 6 which is SHVR dominant and run 7 as the SHVR weak. The following ratios will be used: 20:80 (1:4) and 80:20 (4:1).

Table 2: Population and ratio distribution for runs 6 and 7

Agent	Run 6 (Ratio 20:80)		Run 6 (Ratio 80:20)		Run 7 (Ratio 20:80)		Run 7 (Ratio 80:20)	
	Buyers	Sellers	Buyer	Seller	Buyer	Seller	Buyer	Seller
SHVR	20	80	80	20	1	4	4	1
GVWY	3	12	12	3	20	80	80	20
SNPR	10	40	40	10	10	40	40	10
ZIC	20	80	80	20	1	4	4	1
ZIP	1	4	4	1	10	40	40	10

4.2.1Results

Figure 4 shows the average results from the 4 runs visualised as a box plot. Comparing to Figure 2 from the previous experiment the change is that the ZIP agent has taken over as the leading best trading agent, across the board. However, all agents had their profits decrease across every metric, with the biggest decrease being 49% for the SHVR agent and the least decrease was ZIC at only -15%, followed by SNPR agent with -21%. The worst performers from the previous experiment have had the least disruption to their profits unlike the best performers who've seen almost a 50% reduction.

Figure 5 shows the impact of heavy buyer and seller ratios for respective run, and as expected, when the demand is greater than the supply, the price increase. In the 80:20 ratio, the equilibrium was around 200 meanwhile the seller dominant market's was 120 which is almost double. However, the stability of the markets was not equal, run 7 with the ratio 4:1 buyer to sellers was a lot more stable with very little fluctuations except the occasional extremely high and low ask prices. On the other hand, run 6 with the 1:4 ratio of buyers to sellers, had quite an erratic equilibrium, being as high as 160 and as low as 80.

Overall, the markets have not behaved as expected, with either a constant high demand or supply the expectation would be that the price would steadily either trend downward or upwards, now once again this could be due to the precise configuration of the BSE that was not fully explored by adding more control parameters in the GUI. As the equilibrium price finishes not too far off when it has initially started. Alternatively, this could also be due to the fact that each trader can only sell or buy one share at a time, so even though the agent knows it is cheap to buy due to high supply, it takes too long for it to acquire many shares within the current time constraints. In the real markets, any single trader could move thousand of a single share at once which could drop the price dramatically or increase it, once again this is something that could be explored further by expanding on the current BSE implementation to simulate the markets more realistically.

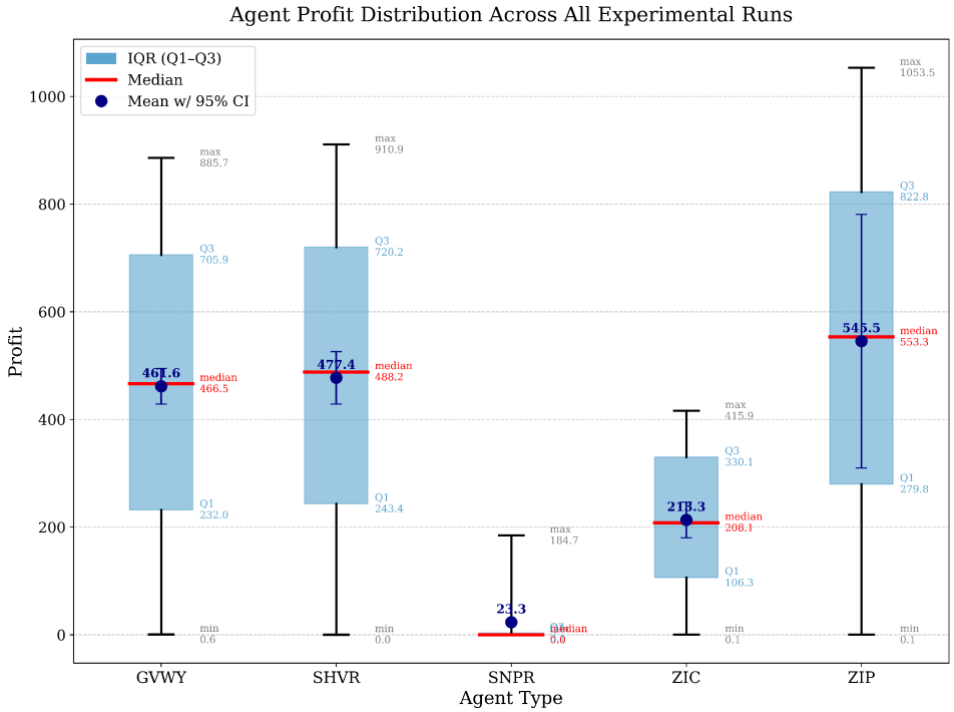
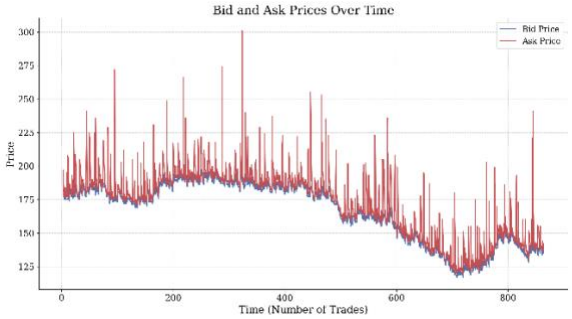
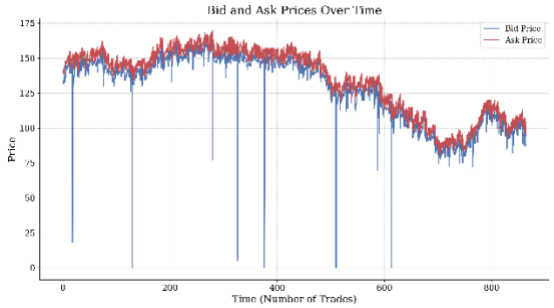


Figure 4: Box blot summarising results from 4 runs for experiment 2

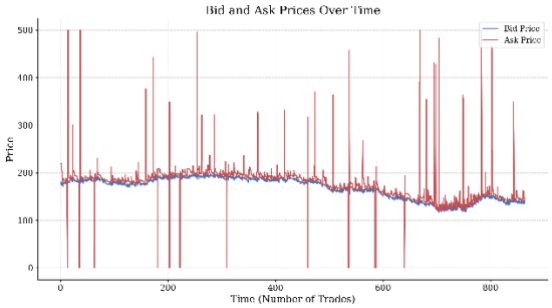
Experiment 2, run 6, ratio 20:80



Experiment 2, run 6, ratio 80:20



Experiment 2, run 7, ratio 20:80



Experiment 2, run 7, ratio 80:20

Figure 5: Bid-Ask Equilibrium for each run in experiment 2[2]

5 CONCLUSION

This study has investigated trading agent behaviour and market dynamics within the Bristol Stock Exchange simulation environment, yielding interesting insights into algorithmic trading strategies and market dynamics.

The first experiment compared heterogeneous trading agents, and found that the shaver (SHVR) agent statistically outperformed all the agents within the experiment across a variety of measurements, achieving 12.6% higher average profits than the second-best Giveaway (GVWY) agents. The performance hierarchy of trading agents (SHVR > GVWY > ZIP > ZIC), aligned with previous research. The most interesting agent was the Sniper (SNPR) agent, with its unique strategy it didn't seem to perform well just by looking at the statistics that this paper used, but showed promising results once it's requirements were met to trade and weren't captured by the boxplots, suggesting a wider variety of statistics and graphs could be used to better capture each agents abilities.

The second experiment looked at how market dynamics shift under skewed buyer-seller ratios. When moving into a more imbalanced market, there was a drastic reduction in profits across each agent, the highest being almost 50% for the previous best agent SHVR. This is where ZIP emerged as a new best trading agent albeit with small margin of confidence.

[2] Note: Full resolution graphs can be found in the */graphs* directory

As economics theory predicted, markets with low supply and high demand will lead to increase prices, however, this also unexpectedly lead to more stable market conditions and less price fluctuations, unlike a market with lots of supply.

5.1 Future developments

This project made a good start into exploring trading agents with BSE; however, this was just a small part of what even BSE has to offer as part of its rich feature set, offering fine control over each parameter. Expanding the GUI to allow greater control over the BSE simulation would greatly increase the flexibility and nuances of experiments that can be run. In addition to more controls via the GUI, seen as there is a functional multi-threaded version of BSE, it would be more practical to use that version instead as the BSE version hasn't received any updates from the other in a while.

In terms of expanding the experiments, the first thing would be to introduce new types of trading agents. BSE already comes with an Adaptive-Aggressive agent that was not used in the experiments. Another important part of real trading is the latency between receiving a stock's price and then sending your order to buy or sell. This is known as High-Frequency-Trading (HFT), where a vast amount of small trades are made to make small profit on each transaction, and the faster an agent can send out a request e.g. by physically being closer to an exchange the more money they can make.

REFERENCES

[1] D. Cliff, "BSE: A Minimal Simulation of a Limit-Order-Book Stock Exchange," *arXiv (Cornell University)*, Jan. 2018, doi: https://doi.org/10.48550/arxiv.1809.06027.

[2] D. Cliff, "davecliff/BristolStockExchange," GitHub, Jan. 08, 2022. https://github.com/davecliff/BristolStockExchange

[3] H. A. Wan, A. Hunter, and P. Dunne, "Autonomous Agent Models of Stock Markets," *Artificial Intelligence Review*, vol. 17, no. 2, pp. 87–128, 2002, doi: https://doi.org/10.1023/a:1014500409896.

[4] M. Rollins and D. Cliff, "Which Trading Agent is Best? Using a Threaded Parallel Simulation of a Financial Market Changes the Pecking-Order," arXiv (Cornell University), Jan. 2020, doi: https://doi.org/10.48550/arxiv.2009.06905.

[5] D. Cliff and M. Rollins, "Methods Matter: A Trading Agent with No Intelligence Routinely Outperforms AI-Based Traders," IEEE Xplore, Dec. 01, 2020. https://ieeexplore.ieee.org/abstract/document/9308172.

A APPENDIX

A.1 Experiment 1 Results

Table 3: Profit summary from experiment 1, run 1

	GVWY	SHVR	SNPR	ZIC	ZIP
mean	757.15	831.66	43.93	259.26	782.88
std	415.27	452.53	79.13	144.84	426.94
min	0.00	0.00	0.00	0.00	0.07
25%	389.80	437.96	0.00	132.63	408.92
50%	784.17	866.75	0.00	263.37	815.09
75%	1155.20	1250.29	55.54	400.73	1179.74
max	1363.07	1513.49	272.84	474.57	1417.46

Table 4: Profit summary from experiment 1, run 2

	GVWY	SHVR	SNPR	ZIC	ZIP
mean	674.80	807.20	33.33	241.26	759.32
std	387.09	458.14	72.73	141.68	426.42
min	0.37	0.00	0.00	0.00	0.00
25%	338.99	409.59	0.00	118.16	399.34

	GVWY	SHVR	SNPR	ZIC	ZIP
50%	676.83	814.59	0.00	239.92	762.20
75%	1023.32	1216.23	0.00	365.11	1144.00
max	1316.55	1566.76	280.90	484.09	1457.70

Table 5: Profit summary from experiment 1, run 3

	GVWY	SHVR	SNPR	ZIC	ZIP
mean	672.28	780.75	33.72	270.01	723.56
std	387.70	448.31	72.98	162.80	412.47
min	0.73	0.00	0.00	0.00	0.11
25%	332.03	391.53	0.00	124.70	365.36
50%	682.13	778.83	0.00	267.20	726.59
75%	1024.70	1179.87	0.00	413.00	1089.96
max	1319.00	1545.03	279.70	534.90	1411.99

Table 6: Profit summary from experiment 1, run 4

	GVWY	SHVR	SNPR	ZIC	ZIP
mean	672.09	781.58	29.70	249.76	723.70
std	389.15	448.83	68.14	145.04	421.23
min	0.09	0.00	0.00	0.00	0.00
25%	335.96	392.02	0.00	122.62	357.46
50%	670.84	787.19	0.00	251.67	722.57
75%	1012.45	1170.92	0.00	375.46	1089.17
max	1341.18	1552.57	283.10	505.07	1446.62

Table 7: Profit summary from experiment 1, run 5

	GVWY	SHVR	SNPR	ZIC	ZIP
mean	676.85	765.36	28.41	232.04	702.49
std	388.11	440.50	65.57	131.54	403.90
min	0.35	0.00	0.00	0.00	0.00
25%	338.69	384.98	0.00	123.20	355.40
50%	680.73	767.41	0.00	230.00	705.23
75%	1017.84	1153.11	0.00	340.00	1054.83
max	1335.65	1524.22	267.60	480.20	1390.67

Note: In the interest of saving space, the rest of the results can be found in supplementary material (*graphs/runs[1...10]*).

A.2 Experiment 2 Results

Table 8: Profit summary from experiment 2, run 6, ratio 20:80

	GVWY	SHVR	SNPR	ZIC	ZIP
mean	467.55	460.16	23.43	255.29	402.35
std	262.63	259.77	48.01	146.54	228.22
min	0.00	0.00	0.00	0.38	0.00
25%	236.47	229.55	0.00	128.97	206.20
50%	477.53	471.21	0.00	253.45	410.00
75%	719.07	697.26	6.02	389.36	610.00
max	879.00	876.75	182.44	497.26	771.20

Table 9: Profit summary from experiment 2, run 6, ratio 80:20

	GVWY	SHVR	SNPR	ZIC	ZIP
mean	485.58	448.24	23.52	167.37	414.54
std	279.07	253.25	48.89	96.13	229.53
min	1.93	0.00	0.00	0.00	0.00
25%	244.53	228.10	0.00	79.22	224.00
50%	486.13	451.47	0.00	169.89	423.00
75%	746.27	681.96	0.00	260.32	618.60
max	939.60	859.50	188.62	317.82	787.00

Table 10: Profit summary, from experiment 2, run 7, ratio 20:80

	GVWY	SHVR	SNPR	ZIC	ZIP
mean	406.26	559.88	22.00	229.47	948.40
std	234.21	312.41	46.41	147.78	539.52
min	0.04	0.00	0.00	0.00	0.53
25%	199.16	288.40	0.00	102.60	478.58
50%	405.76	579.40	0.00	206.60	958.88
75%	616.21	844.00	0.00	361.60	1428.11
max	796.22	1063.80	176.46	492.60	1856.81

Table 11: Profit summary, from experiment 2, run 7, ratio 80:20

	GVWY	SHVR	SNPR	ZIC	ZIP
mean	487.04	441.15	24.24	201.24	416.72
std	274.98	243.01	49.67	107.26	235.52
min	0.47	0.00	0.00	0.00	0.00
25%	247.70	227.40	0.00	114.60	210.36
50%	496.49	450.60	0.00	202.60	421.22
75%	741.93	657.40	6.50	309.20	634.52
max	928.16	843.40	191.08	356.00	798.88

A.3 Install Instructions

Unzip *comp3004-master.zip*

Open comp3004-master using PyCharm or similar IDE

PyCharm should prompt to create a new virtual environment, select at least Python version 3.12 and install dependencies from *requirements.txt*

Run *BSESettings.py*

To run the simulation click *Apply & Run Simulation*