Wall Street Bets: The Game

A Competitive Multi-Agent Approach

Carolina Carreira
carolina.carreira@tecnico.ulisboa.pt
87641

João Olival
joaoljolival@gmail.com
94111

Sebastião Almeida
sebastiao.lna@outlook.com
97115

Abstract

The stock market is a system where multiple agents interact. This has inspired us to implement this fun game to educate users about the economy. Each agent is an investor and agents can have different strategies (based on real life examples) and their goal is to maximize their value. By testing the developed system we learned more about the agent's interactions and how different strategies perform against each other. We evaluate the agents to see which strategy is best for each scenario.

1 Introduction

The economy is an ever-changing environment. This year the stock market has been on the news due to the WallStreetBets [4] subreddit. The behaviour of a relatively small group of people that gathered in an online forum and rallied behind the stock of a failing yet nostalgia-evoking game retailer company managed to send shock waves through the world of finance and make it to mainstream news, upsetting the usual balance of the stock market. This event led many people to become interested in the stock market, us included. As a result, we created a game to **simulate the stock market** in a fun and educating way, and to try to evaluate how different market strategies might fare in a setting that is meant to simulate the real-world stock market. Hopefully, the users may learn strategies that they could use in their own stock investments.

They buy and sell shares to the central bank , an entity that knows everything.

The players (agents) are **competitive** towards each other and can have one of several different strategies. We aim to study how these different strategies perform against each other in a variety of environment settings, with different degrees of entropy.

Our **goals** with this project are to implement intelligent agents that will provide insight about strategy and scenario characteristics. It is important to keep in mind that these rules are based on real economic models but are not realistic. This is just a tabletop game.

Our findings suggest that reactive agents are very effective in an high entropy environment and that the implemented Reinforcement Learning Agent performs better in low entropy environments, as it learns quicker (due to the consistent reward in a more stable environment).

2 Approach

2.1 Functional Requirements

To design our system, we based ourselves on a set of **Requirements** that combine characteristics of the real world and the stock market with the limitations of this kind of simulations and with our academic goals for this project.

1. Our system should have a **customizable set of agents** whose goal is to maximize their total value (i.e. cash in the bank + total value of stock owned, see formulae 3) over time.
2. Our system should work in **discrete time** and all agents should have the opportunity to buy and/or sell stocks at each time step.
3. The number of steps of each run should also be customizable.
4. The price of the stocks should take into account the normal rules of supply and demand.
5. There should also be some **realistic events** that can happen throughout the simulation and influence the price of the stocks, though the option to disable them will also be present.
6. Each agent should start with a fixed amount of cash to invest in stocks.
7. There should be a series of companies with relations between them that influence their stock price (competitors and complementary industries).

2.2 Environment

The environment has the following properties:

1. **Inaccessible**: the agents do not know what share other agents have. They only know which shares are for sale and how much each share is worth.
2. **Discrete**: as there are a set number of possible actions (see section 2.4), and they always have the same output (e.g. if an agent has money and a share is for sale he is always going to be able to buy that share).
3. **Static**: the game works in round. In one round each agent makes a decision after the order, so while one agent is deliberating the game stops.
4. **Non-deterministic**: as some shares can have random value changes between rounds (see section 2.2.1) and some events can also impact stock value (see section 2.2.1).
5. **Non-episodic**: all the rounds are dependent on each other. Current share value is influenced by past share

values and the agent's actions have impact in the evolution of the shares' value.

This is environment is similar to a tabletop game environment (see Section 1).

2.2.1 Types and Values of Shares. The value of shares depends on several factors and is calculated at the end of each round:

- **Supply and demand**: if at the end of the round there are more shares of company A for sale than in the previous round, the value of A's shares decreases. Likewise, if there are less for sale than in the previous round its' share value increases.
- **Complementary industries**: if two companies are complementary, their value increases proportionally (e.g. cigarettes and lighters, the more cigarettes sold the more lighters, so if one company's value increases so does the other). For example, our solution Tesla is complementary with Intel, as is Microsoft is with Intel.
- **Competitor companies**: similarly, if two companies are competitors, their value will be affected inversely. In our solution BP and Galp are competitor and Aldi and Primark are competitor.
- **Random shares**: some shares have random fluctuations in value.
- **Global events**: to better simulate the daily uncertainty of the stock market, caused by the possible effects that occurrences around the world may have on investors and companies (and, therefore, the stock prices), we created a few possible events designed that affect stock prices, inspired by real world events and scenarios. For example:
  1. COVID19: after a global pandemic the value of vaccines companies' increases, in our solution this happens with Moderna's shares.
  2. Tech Boom: at a certain point in time there may be a tech boom event in which all technological companies' values increases, and their subsequent shares.
  3. Oil Crisis: During and Oil Crisis stock price of Oil companies will decrease, in our solution BP and Galp are affected.

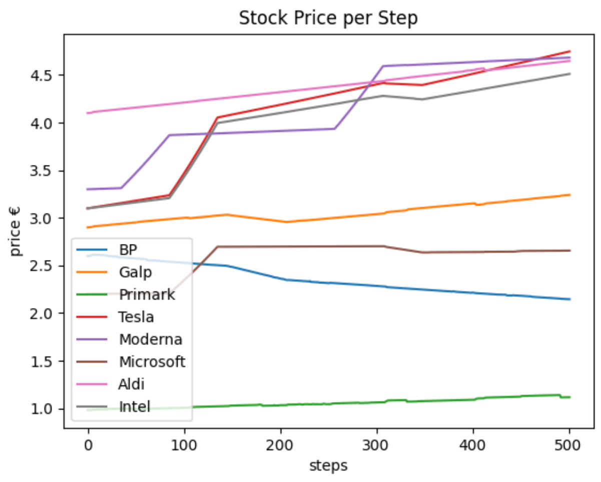
We can see and example of the expected evolution of the stock price in fig. 1 during run with 5000 steps. The stocks in the game can be changed and added easily in the code.

2.2.2 Scenarios. When simulating the game the user is able to choose a scenario. Some scenarios include:

Default the default scenario according to Section 2.2.1. In this scenario we have all the events turned on and the rules of Supply and demand, Complementary industries and Competitor companies are turned on.

Recession the value of all shares is more likely to decrease;

Figure 1. Example of stock price evolution in an run



Inflation the value of shares is more likely to increase;

These scenarios are relevant to learn more about the nuances of agent behavior in different circumstances. The user is able to change the scenarios in the "Options" menu in the GUI when playing the game.

2.3 Agent System

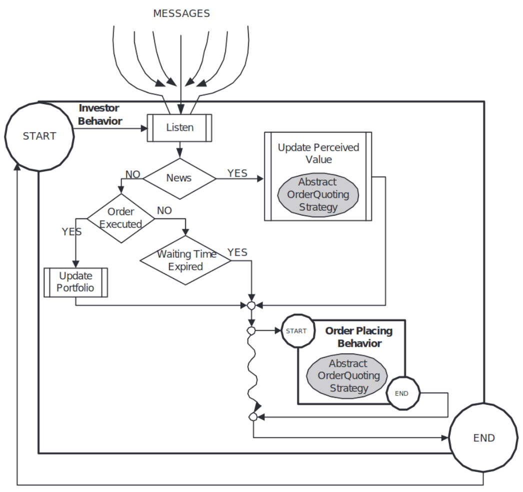
The players (agents) are **competitive** towards each other and can have several different strategies. This game can have from 2 to 8 players, but we simulated up to 5000.

The **challenge** in this environment is to determine which investments will return a profit. Another key challenge is understand when to sell shares, so as to not end up selling them for a value lower than they were brought.

2.3.1 Agent's Properties. The agents in our system have the following characteristics:

1. **Autonomous**: the agent acts independently and may learn from the environment.
2. **Adaptive**: the agent adapts to the environment, for example, stops buying in a recession.
3. **Rationality**: the agent acts in a way to to maximizes its value in cash and stocks owned.
4. **Curiosity**: sometimes the learning agent will explore for different way to improve its performance.
5. **Reactive**: the agent will react to changes in the environment accordingly.
6. **Proactive**: the agent will take the initiative to try different stocks and see it their prices increase.
7. **Believable**: the agents have an avatar and are presented to the users around a table as if playing the board game with each other.
8. **Not mobile**: this agent characteristic does not apply to this scenario.

Figure 2. Example of an investor agent [1]



9. **Veracity**: the agents will respect the rules of the game, there are no byzantine or malicious agents.

2.3.2 Simple Reactive Agent.

To design the strategies our agents used we studied the work of Boer et al. [1] where they describe several reactive banking agent behaviors, like the one seen in Fig. 2. These agents make decisions based on new market data [1] and are adequate to our problem for that reason.

The first reactive agent does not have memory and is the *Simple trend follower* [1] or Simple Reactive Agent. This agent will try to follow the stocks prices trends by buying stock if its price increases and selling if its price decreases. This agent has no memory but communicates with the central bank to see the last price change. It is based on this immediate price change that the agent makes its reactive decision. As such the agents only “sees” one round.

If this agent decides to buy a certain stock it will respect the following: Let max be the maximum amount of stock it can buy, $stock_{tobuy}$ the amount he will buy then:

$stock_{tobuy} > 1 \land stock_{tobuy} \leq max$

This agent is close to the perfect agent in our environment. This is because this agent will react in a quick way to the environment. By being this quick this type of agent is used by stock trading companies and operates real-time [1]. Disadvantages of this agent are that it may spend money on stocks that do not present a positive long-term trend. It is close to the ideal agent but can wrongly identify a trend, sell a stock and it will increase in value after selling.

2.3.3 Careful trend follower.

The second type of reactive agent is the Careful trend follower [1] aka Careful Agent. This agent was inspired by the *Risk Averse Agent* described by Brandouy et al. [2] in which the amount of risk these agents take is minimized. This is a reactive agent with memory and has a few characteristics that differentiate him from the others:

1. He never invests his whole capital at once, this is he will not buy the maximum number of stocks he can buy, even if he sees a positive trend. This is another facet of his carefulness.
2. He will wait for a set of rounds until seeing a positive trends before investing as seen in fig. 5.

Although this agent is careful when buying stocks he is also careful when deciding to hold said stock. When a careful agent sees a negative trend in a stock he will not try to hold the stock and will sell it immediately (on the first round the stock goes down).

Disadvantages of this agent are that he is not as quick to respond to positive trends was the Simple Reactive Agent and as such may not increase in value as much in some scenarios (see Inflation scenario in fig. 10 where the Simpler Agents performs better).

2.3.4 Random agent.

The Random agent is essential to our solution and he will buy, sell and hold stock randomly though the game. This agent will increase the entropy of the systems as, by buying and selling stock he will influence market prices (see 2.2.1) and by having other agents react to market prices, the random agent will also influence other agents.

This agent is also important to provide a baseline for the Reinforcement Learning and Reactive agents.

2.3.5 Reinforcement Learning Agent.

Although simple reactive agents are quicker to react to changes in an environment and as such widely used in this type of environment [1], a learning agent has the possibility to react in a better to a random event in the environment.

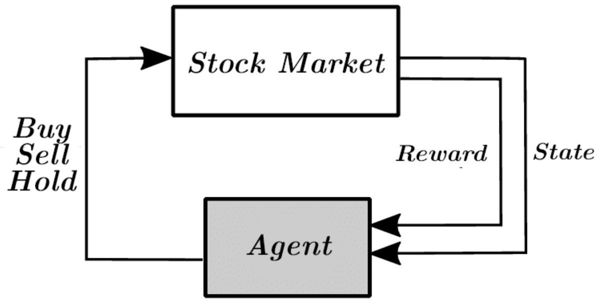
We choose a to use a Reinforcement Learning Agent because this framework assumes learning under uncertainty in a dynamic environment, something that our environment has.

There are several Reinforcement Learning algorithms and we chose to use one of the most well-known, *Q-learning* [6]. It is an example of an off-policy control method because it directly approximates the optimal action-value function independently of the agent’s actions. Through a process of trial-and-error (see fig. 3), the agent learns an optimal policy, by approximating the action-value function: $Q(s, a)$ as seen below. Where s_t is the state at t, a_t the action, α is the learning rate and γ is the discount rate.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[reward_{t+1} + \gamma max_b Q(s_{t+1}, b) - Q(s_t, a_t)] \quad (1)$$

To update its Q a learning agent explores the environment and receives a reward. Our Reinforcement Learning will measure its success from each round based on the function

Figure 3. Diagram of Reinforcement Learning Agent



reward (see bellow eq. (2)), where s_t is the state at t, a_t the action, and $agentValue()$ the function in eq. 3.

$$reward(s_t, a_t) = agentValue(s_t) - agentValue(s_{t-1}) \quad (2)$$

Before and after updating its Q our agent selects an action based on an ϵ-greedy action selection function. This action selection strategy chooses actions with maximal estimated value but with probability ϵ elects a random available action. As time goes on we decrease the parameter ϵ to allow higher exploration rate at the beginning of learning and more exploitation in the end.

Advantages of this agent are that he will be able to learn and a such react to a wider variety of environments due to being curious and exploring his options. Disadvantages are that this agents requires a large amount of training in order to begin performing well.

2.4 Actuators and Sensors

Each of the agents has a set of Actuators and Sensors. Actuators:

- *buy(amount, shareName)*: sell a certain share
- *sell(amount, shareName)*: buy a certain share
- *hold()*: do nothing in a certain round

Sensors:

- *valueOfShare(shareName)*: returns the value of a share;
- *availableShares(shareName)*: returns array of shareNames that are available for buy.

3 Empirical evaluation

After the implementation of the systems a series of intensive test were conducted in order to derive conclusion from the environment.

3.1 Metrics

The main goal agents have is maximize their value. To measure the agents' value we used a metric similar to *wealth* used by Souissi et al. [5] where the agent value is the sum of the cash they have with how much each share they own is worth, like so:

$$agentValue = moneyValue + \sum_{s=firstShareName}^{lastShareName} valueOfShare(s) \quad (3)$$

In each round we can use the following metrics:

- agent value (money in wallet+share value, eq. (3));
- agent share value;
- agent money in wallet;

3.2 Methodology

To improve the agents behaviour understanding according to the market variations, a GUI was built to ease the manipulation of time steps and the data driven conclusions. In order to build the GUI a cross-platform set of Python modules designed for writing video games was used - ***Pygame***. With this modules implemented we were able to review the behaviour of the different agents step by step within a clear UI. It also improved the possibility of refinement of the agents during the development process. The implementation of this GUI was also one of the Functional Requirements (see Section 2.1).

Python plotting library was also used - ***MatPlotLib***, to test the implementation after each round and set of rounds. With this tool we were able to plot the results of the different agents and compare them with each other on the different scenarios. Each test was conducted 10 times for each agent and game mode. Each point in the following graphs is the average of 10 runs of the same agent. In most figures we use the performance of the Random Agent as a baseline to compare with other agents' performance.

3.3 Simple Reactive Agent Results

This is almost a perfect agent and performs well in all three scenarios (see subsection 2.2.2). In the following figure (fig. 4 we can the the average of 10 rounds, and as we can see the Simple Reactive Agents performs better than the random agents. The random agent can be seen as a baseline agent with which we compare the simple reactive agent.

3.4 Careful Reactive Agent Results

The careful reactive agent will only invest when he sees a trend but in order for it to identify a trend it has to be present in n rounds. In fig. 6 we can see its performance against two random agents. This agent clearly performs better then the baseline random agent.

Nonetheless this agent loses against the simple reactive agent in an low entropy scenario like in fig. 7. We can conclude that because the simple reactive agent is quicker to respond to changes in the environment and is less careful when investing, a positive in this environment.

Let n be the number of round the careful agent waits before recognizing a trend:

- if $n = 0$, he never recognizes a trend;
- if $n = 1$ behaves the same way as the simple reactive agent;

Figure 4. Average performance of Simple React Agent in Default Scenario (in 10 runs)

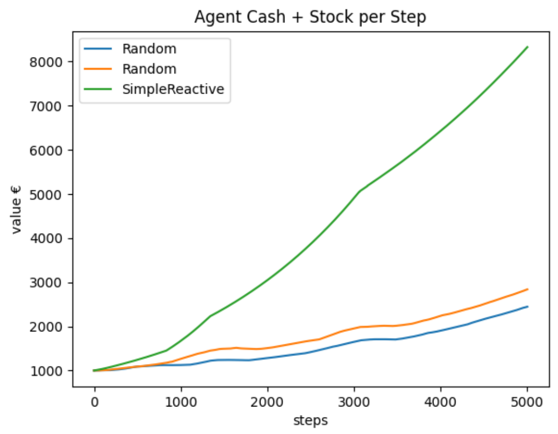
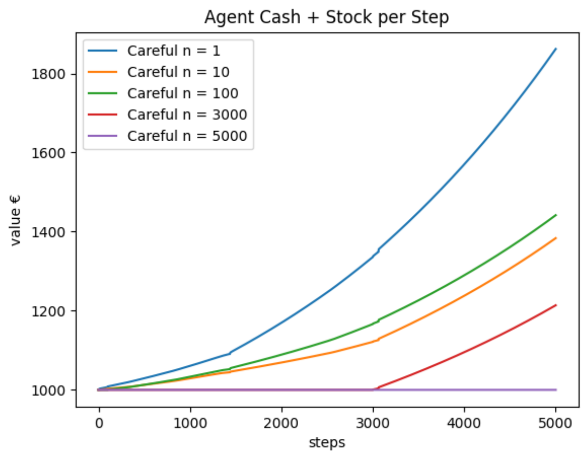


Figure 5. Careful agent performance comparison according to n (the number of rounds before recognizing a trend) in Inflation Scenario



- and as $n \longrightarrow +\infty$ this agent will wait more time before making the decision to buy a certain stock after seeing a positive trend.

The default value of this modifier is 5 (as a normal game has 40 rounds). Nonetheless we tested this agent with different values of n. And found that the empirical conclusion corroborated our previous conclusions about the n value.

We can see such a comparison in figure 5. As $n \longrightarrow \infty$ the agent need more rounds before committing to invest in a stock, to the extreme of never investing. In fig. 5 $n = 5000 \equiv \infty$ because the last step was 5000.

Figure 6. Average performance of Simple React Agent in Default Scenario (in 10 runs)

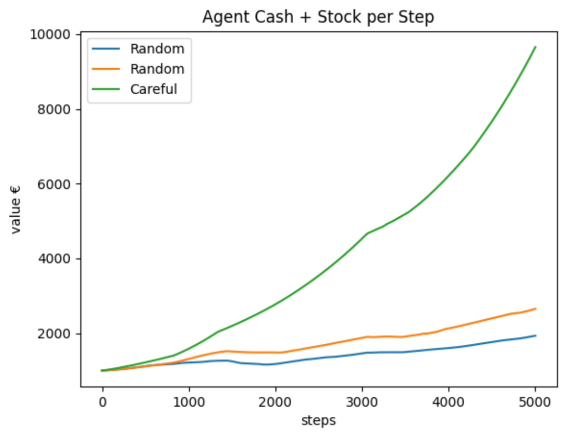
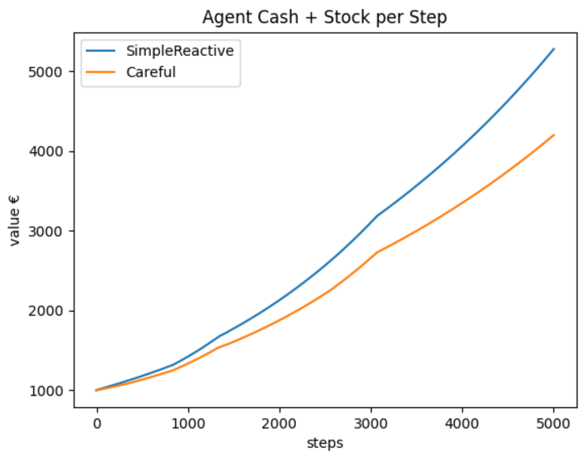


Figure 7. Comparison between Careful and Simple Reactive Agents in Default Scenario



3.5 Reinforcement Learning Agent Results

Our reinforcement agent, as stated in Section 2.3, uses a *Q-learning* algorithm with an ϵ-greedy action selection function. As times goes on we decrease the parameter ϵ to allow a higher exploration rate at the beginning of learning and more exploitation in the end.

As previously stated the Reinforcement Learning Agent begins by exploring without any information about the environment. One challenge arising in Reinforcement Learning is the trade-off between exploration and exploitation: an agent must prefer actions tried in the past and found to be effective in producing reward (exploitation); however, to discover such actions, it has to try actions that it has not selected before(exploration). Efficient Reinforcement Learning thus requires a balance between these concepts.

Figure 8. Average performance of Reinforcement Learning Agent in Default Scenario (in 10 runs)

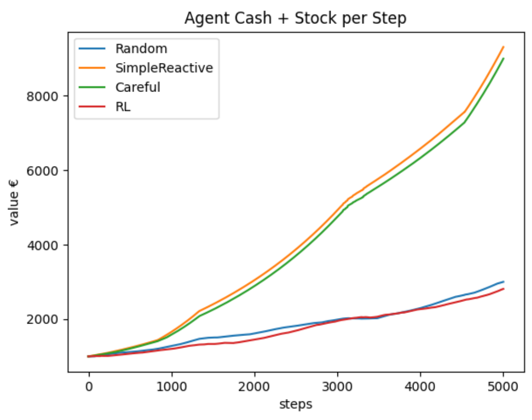
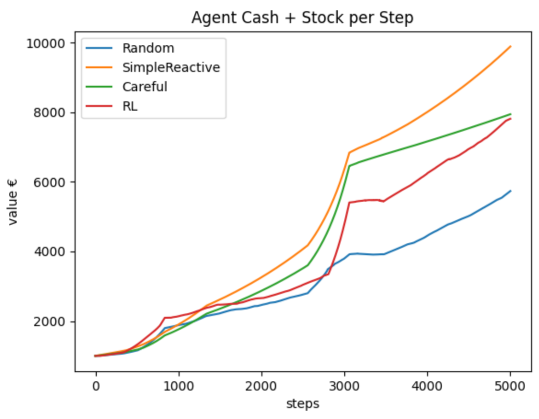


Figure 9. Performance of a single run with all Agents in Default Scenario



The performance of our Reinforcement Learning Agent is equivalent to a Random Agent in games with a small amount of steps. This is clear in our video demonstration and on an normal run of the game with 40 steps .

Nevertheless after a few thousand rounds we can start to see and improvement on the performance of the Reinforcement Learning Agent against the Random Agent. This is clearly demonstrated in the Recession Scenario in 11, and in the Inflation Scenario in fig. 10. Although we have positive results in the previous scenarios in the Default Scenario, with more entropy and random events, in average, the Reinforcement Learning Agent is equivalent to a baseline agent (e.g. random) as seen in fig. 8.

3.6 Overall results

In this section of the report we hope to address each of the scenarios and each of the agents competing with each other.

Figure 10. Average performance of Agents in Inflation Scenario (in 10 runs)

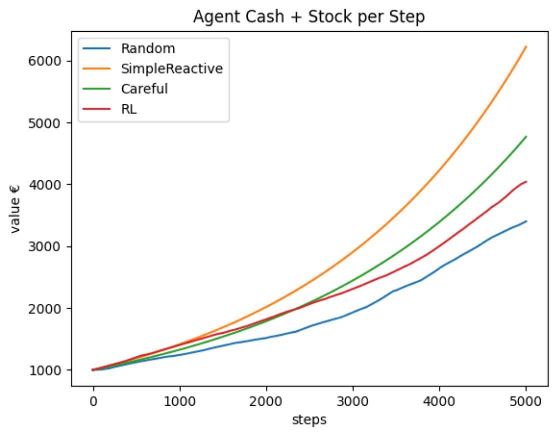
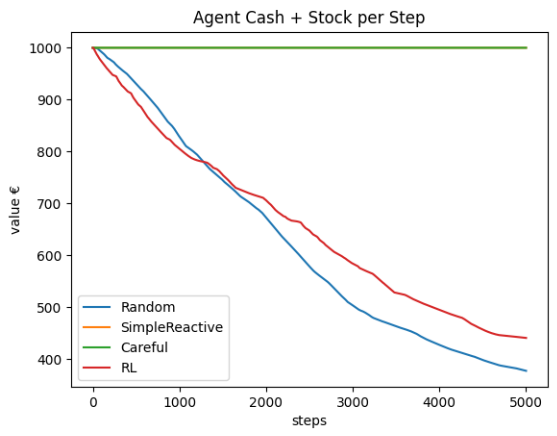


Figure 11. Average performance of Agents in Recession Scenario (in 10 runs)



Default Scenario. in this scenario events are enabled as well as all the stock modifiers (e.g. Law of Supply and Demand). The best performing agent, in average, for this scenario is the Simple Reactive Agent. In fact the Reinforcement Learning Agent, although sometimes presents a good results, in average it does not have a better performance than the Random Agent (see fig. 8). https://www.overleaf.com/project/609e4bd04b4f482

Recession Scenario. in this scenario events are disabled as are all the stock modifiers (e.g. Law of Supply and Demand). All stocks go down, and this is an extreme scenario design with the purpose to test our agents. Here, in average, the best performing agents are the reactive ones. These never see an up-warts trend and as such never invest and lose value. Nonetheless we can see an improvement of the performance of the Reinforcement Learning Agent after a few thousand steps (see fig. 11).

Inflation Scenario. this is the last available scenario and one where all the stocks grow. In this scenario the events and normal stock price modifiers are also disabled. The results are similar to the Recession Scenario where the Reinforcement Learning Agent performance begins to clearly improve after a few thousand steps and where the reactive agents perform the best (see fig. 10).

3.7 Limitations and future work

A limitation of our scenario is the static nature of our environment. While one agent is deliberating all others wait for his decision. As such for a more realistic approach, in the future, we would like to implement our solution in a more dynamic environment. Another vein of future work that would be interesting to explore is the implementation of more types of agents, particularly a deliberative agent, and other more complex ones with different learning capabilities and techniques, such as the ones seen in the work of Boer et al. [1] and [2].

4 Conclusion

The first conclusion we take from our results is that Reactive agents, risk averse or not, perform very well in this environment. Even in a high entropy system the simple reactive agent performed better (see table 1). This is in agreement with the literature as reactive agents are used in real-time stock trading systems [1]. To corroborate this we present the average agent value after running the Default scenario 10 times with 5000 steps in table 1.

The careful agent has a more passive strategy and as concluded by Keim et al. [3] the cost of missing an opportunity when trading is high for more careful agents.

Lastly the Reinforcement Learning Agent performed well in simple scenarios (see Recession and Inflation in fig. 10 and fig. 11). And we can see an improvement of its performance in runs with a large amount of step (e.g. in the thousands).

We hope that our work can be used to better understand reactive and reinforcement learning agents in high and low entropy environments. Our work and its very detailed UI can also be used to study stock price evolution and interaction. To conclude, we want to add that we have achieved all the original functional requirements (see Section 2.1) and reached our goal to gather conclusion about different agent strategies playing with each other.

Table 1. Average agent value after 5000 steps in Default Scenario in 10 games

Rank	Agent	Avg. Value
#1	Simple Reactive	9914
#2	Careful	9686
#3	RL	3652
#4	Random	3481

References

[1] Katalin Boer-Sorban. 2008. *Agent-based simulation of financial markets: a modular, continuous-time approach.* Number EPS-2008-119-LIS.

[2] Olivier Brandouy, Philippe Mathieu, and Iryna Veryzhenko. 2013. On the Design of Agent-Based Artificial Stock Markets. *Communications in Computer and Information Science* 271. https://doi.org/10.1007/978-3-642-29966-7_23

[3] Donald B Keim and Ananth Madhavan. 1995. Anatomy of the trading process empirical evidence on the behavior of institutional traders. *Journal of Financial Economics* 37, 3 (1995), 371–398.

[4] Miguel Prado. 2021. Wall Street Bets: o fenómeno que está a agitar as bolsas dos EUA. https://expresso.pt/economia/2021-01-28-Wall-Street-Bets-o-fenomeno-que-esta-a-agitar-as-bolsas-dos-EUA-visto-de-Portugal

[5] Mohamed Souissi, Khalid Bensaid, and Rachid Ellaia. 2018. Multi-agent modeling and simulation of a stock market. *Investment Management* (2018).

[6] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.