COMP4801_FITE4801 Final year project [Section FA]

Topic: **Optimal Order Placement in Cryptocurrency Markets**

Mohammad Muttasif (UID: 3035667778)
Kritik Satija (UID: 3035666645)
Raghav Agarwal (UID: 3035720697)

Date of Submission: 24-04-2023
Abstract
In financial markets, the execution of orders is a crucial aspect of trading activities. Currently, the process is manual and susceptible to human error, necessitating the implementation of algorithmic automation. This paper seeks to deliver a market simulator and propose a solution for order execution using Deep Q-Network (DQ-N), a form of Reinforcement Learning (RL), and micro-price. Due to constraints such as exploration-exploitation tradeoff and sparse rewards, the DQ-N algorithm fails to meet the Time-Weighted Average Price (TWAP) benchmark. In contrast, the micro-price algorithm outperforms the TWAP algorithm and generates alpha.
Acknowledgement

I would like to thank my supervisor, Dr. Liu Qi, for all the encouragement and help he's given me over the year.

Without the contributions of my group mates, Kritik Satija and Raghav Agarwal, as well as their dedication to the project and positive attitudes, we would not have been able to develop this idea. Talking with my teammates about our project or anything related to technology has always boosted my knowledge. Without these contributors, it simply could not have happened. It was inspiring how positive and encouraging you were always despite how busy you were.

My parents, Mr. Arif Haider and Mrs. Asna Rizvi, also deserve my gratitude. My two best friends in the whole world. There have been countless times when I have needed emotional support, and they have been there to provide it, as well as words of encouragement.

And finally, I would like to express my gratitude out to Dr. Matthew YEUNG from the Centre for Applied English Studies for his insightful suggestions that helped for writing this paper.
# Table of Contents

Abstract ................................................................................................................................. 2  
Acknowledgement .................................................................................................................. 3  
Abbreviations: .................................................................................................................... 6  
List of Tables: ....................................................................................................................... 7  
List of Figures ....................................................................................................................... 8  
List of Equations .................................................................................................................. 10  
Chapter 1: Introduction ........................................................................................................ 11  
  1.1 Objectives ...................................................................................................................... 12  
  1.2 Motivation .................................................................................................................... 12  
  1.3 Scope ............................................................................................................................ 13  
  1.4 Deliverables: .................................................................................................................. 13  
  1.4 Report Outline............................................................................................................... 13  
Chapter 2: Methodology: ..................................................................................................... 14  
  2.1 Input Data ..................................................................................................................... 14  
  2.2 Exchange Simulator ...................................................................................................... 14  
  2.3 Deep Q-Network (DQ-N) Algorithm ............................................................................. 15  
  2.3.1 States ......................................................................................................................... 17  
  2.3.2 Actions ....................................................................................................................... 17  
  2.3.3 Rewards ..................................................................................................................... 18  
  2.3.4 Hyperparameters ....................................................................................................... 18  
  2.3.5 Limitations ................................................................................................................ 19  
  2.3.6 Assumptions ............................................................................................................. 20  
  2.4 Data Analytics .............................................................................................................. 20  
  2.4.1 Time-Weighted Bid-Ask Spread ............................................................................... 20  
  2.4.2 Time-Weighted Orderbook Depth ............................................................................. 21  
  2.4.3 Time-Weighted Price Slippage ................................................................................ 21  
  2.5 Microprice .................................................................................................................... 22  
  2.5.1 Background ............................................................................................................... 22  
  2.5.2 Input data ................................................................................................................... 24  
  2.5.3 Algorithm .................................................................................................................. 30  
  2.6 Benchmarks ................................................................................................................... 33  
Chapter 3: Findings ............................................................................................................. 33
3.1 Output and Performance ............................................................................................................. 33
3.1.1 DQ-N Algorithm .................................................................................................................. 34
3.1.2 Micro-price ........................................................................................................................... 37
Chapter 4: Conclusion .................................................................................................................... 46
References ......................................................................................................................................... 49
Appendices ....................................................................................................................................... 50
Appendix A - Project Schedule ....................................................................................................... 51
Appendix B - Market Simulator ....................................................................................................... 52
Appendix C – UI Design (Suggestive) ............................................................................................. 53
Abbreviations:
API    Application Programming Interface.
BTC    Bitcoin.
DQ-N   Deep Q-Network.
ETH    Ethereum.
ML     Machine Learning.
RL     Reinforcement Learning.
TWAP   Time Weighted Average Price.
USD    United States Dollar.
VWAP   Volume Weighted Average Price.
List of Tables:

A1 ................................................................. Timeline of the project
List of Figures

Figure 1.1 The RL Model. The agent performs actions based on the current state, or observation. The ML environment gives the reward based on the action, and returns a new state. .......................... 12

Figure 2.1 Time-weighted Bid ask spread for NEER-USDT on Gate.io. It shows the relationship between the bid-ask spread with time. ............................... 21

Figure 2.2 Time-weighted order book depth for NEER-USDT on Gate.io. It shows the relationship between the notional with time ............................. 22

Figure 2.3 Time-weighted price slippage for NEER-USDT on Gate.io. It shows the relationship between the slippage with time ............................. 22

Figure 2.4 Orderbook depth of YFII-USDT ...................... 26

Figure 2.5 Slippage of YFII-USDT ......................... 26

Figure 2.6 Spread of YFII-USDT ....................... 26

Figure 2.7 Orderbook depth of GRAIL-USDT .............. 29

Figure 2.8 Slippage of GRAIL-USDT ................. 29

Figure 2.9 Spread of GRAIL-USDT ............. 30

Figure 2.10 The algorithm for buying ...................... 32

Figure 2.11 The algorithm for selling ...................... 33

Figure 3.1 Plot of Rewards-Iteration with first set of hyperparameters .......... 35

Figure 3.2 Plot of Rewards-Iteration with second set of hyperparameters .............. 36

Figure 3.3 Plot of Rewards-Iteration with third set of hyperparameters ......... 37

Figure 3.4 Plot of Rewards-Iteration with first set of hyperparameters .......... 39

Figure 3.5 Plot of Rewards-Iteration with first set of hyperparameters ...... 40

Figure 3.6 Plot of Rewards-Iteration with first set of hyperparameters. 41

Figure 3.7 Plot of Rewards-Iteration with first set of hyperparameters. 42

Figure 3.8 Plot of Rewards-Iteration with first set of hyperparameters. 43

Figure 3.9 Plot of Rewards-Iteration with first set of hyperparameters. 44

Figure 3.10 Plot of Rewards-Iteration with first set of hyperparameters. 45

Figure 3.11 Plot of Rewards-Iteration with first set of hyperparameters. 46

Figure B.1 A screen capture of the market simulator. The orderbook is for BTC/USD and the depth is 5 levels. There is a change whenever a buy/sell order is placed .......................... 52
Figure C.1 Wireframe for the place order tab. The user can choose between a market and a limit order ............................. 53

Figure C.2: Wireframe for the market order tab. The user can place a market order after choosing from a buy/sell order and giving a notional and asset.......................... 54

Figure C.3: Wireframe for the limit order tab. The user can place a limit order after choosing from a buy/sell order and giving a price, quantity and asset......................... 55

Figure C.4 Wireframe for the history with the order details with the option of cancelling incomplete orders............................. 56
List of Equations

Equation 2.1 Orderbook structure ......................... 15
Equation 2.2 Reward function ............................. 18
Equation 3.1 hyperparameters1 ......................... 34
Equation 3.2 hyperparameters2 ......................... 35
Equation 3.3 hyperparameters3 ......................... 36
Chapter 1: Introduction

For seasoned investors, cryptocurrencies have emerged as a novel and enticing asset class. The widespread use of cryptocurrencies is anticipated to have a significant impact on global economies [1]. Several studies of the cryptocurrency market are centred on optimal order placement. For example, models that maximise limit order price, limit order market share, or allocation among exchanges are proposed [2]. However, it may be difficult to implement these strategies in the actual world because they require large approximations. Users have access to simple algorithms such as Time Weighted Average Price (TWAP) and Volume Weighted Average Price (VWAP), as well as more powerful algorithms such as Reinforcement Learning (RL). It is necessary to define RL in this context. It is a Machine Learning (ML) technique that investigates how systems can learn to predict the outcomes of their actions and enhance their behaviour during state transitions based on rewards and punishments (See Figure 1.1). [3]. A state represents the current condition of the environment within which an agent operates. The agent takes action to influence the outcome of the environment, which results in either positive or negative reinforcement. A reward is the reinforcement, either positive or negative, provided to an agent to indicate whether their actions were favourable or not.

The project also incorporates the micro-price algorithm. It is an asset fair value estimator that uses orderbook data at level 1. Microprice offers several benefits and has been empirically demonstrated to be a superior predictor of future prices, in contrast to simpler methods such as weighted mid-price. As a consequence, we anticipate that integrating Microprice into our execution strategy will yield significant benefits.

In addition, previous research has investigated the application of reinforcement learning (RL) in trading, specifically for optimising
order placement. Schaul et al. developed an RL-based trading agent that outperformed existing strategies for the stock market.

![RL Model Diagram](image)

**Figure 1.** The RL Model. The agent performs actions based on the current state, or observation. The ML environment gives the reward based on the action, and returns a new state.

### 1.1 Objectives

In order to give traders the best possible strategy, the project aims to provide a useful model to users by trying a number of different strategies. In addition to this, it aims to create practical models based on tested methods and customize execution algorithm parameters by taking various factors such as orderbook depth and bid-ask spread into consideration. With this agenda, the project aims to accomplish three objectives: First, train an RL model to determine the size and cost of a limit order. Second, develop an intuitive user interface that enables users to enter various variables as needed. Third, compare the performance of different algorithms based on various metrics.

### 1.2 Motivation

Existing literature on optimal execution in cryptocurrencies only addresses extremely liquid markets, such as the Bitcoin to United States Dollar (BTC/USD) and Ethereum to United States Dollar (ETH/USD) pairs, which is a problem. In addition, traders cannot access complex algorithms such as RL models on the majority of platforms. Numerous studies have investigated the application of RL in trading, specifically in the context of optimal order placement. Schaul et al. [4] developed a trading agent for the stock market that outperformed several existing trading strategies using an RL-based approach. To address these issues, our team strives to provide frequent traders with improved execution.
service and usability. Additionally, we strive to improve the efficiency of the bitcoin market.

1.3 Scope
This initiative focuses on enhancing trade execution. Several design decisions must be made for effective project administration. In order to appeal to a larger audience, we use the Gate.io exchange and two alternative tokens, YFII-USDT and GRAIL-USDT, that reflect varying degrees of liquidity and volatility.

1.4 Deliverables:
The assignment has been completed on time (refer to table A.1 in Appendix A). We have completed the development of the cryptocurrency market simulator, on which market and limit orders can be placed, as well as the TWAP and VWAP algorithms. We have also designed the wireframe for the user interface (See Appendix C) and read numerous research papers on testable algorithms. In addition, between 5 January 2023 and 15 February 2023, we implemented and tested an RL algorithm on this simulator, and between 1 March 2023 and 15 March 2023, we completed the user interface design. The deliverables of this report include the market simulator (see Figure B.1 in Appendix B), the RL algorithm, the wireframe for the user interface, and the implementation of the micro-price algorithm.

1.4 Report Outline
The remaining sections of the report are as follows: Chapter 2 discusses the methodology employed, Chapter 3 presents and analyses the results, and Chapter 4 concludes the report. The report is finished off with chapter 6. It provides a summary of the entire report as well as a description of the progress that has been made up until this point.
Chapter 2: Methodology:

This chapter seeks to introduce briefly the machine learning algorithm, input data, exchange simulator, data analytics, accuracy benchmark, model assumptions, and pertinent theory.

2.1 Input Data

The market microstructure data served as input for the tailored RL algorithm. Market microstructure data is the data that financial exchanges maintain and disseminate that provides the specifics of limit order books from the past [3].

To simulate the market, these data were obtained from two assets with distinct liquidity and volatility levels. Choosing the appropriate cryptocurrency exchange is also crucial. Gate.io was the best option for the purposes of this undertaking.

It has a vast selection of coins and provides access to historical orderbook data for training models.

Python's web sockets library was used to retrieve the information. This provided real-time data from Gate.io for the RL model's training and testing.

2.2 Exchange Simulator

Using two ordered lists of (price, size) pairs, the order book for the exchange simulator was generated in Python (See Equation 2.1). These are the asking and asking prices.
\[ Bids : [(P^b_i, N^b_i) \mid 0 \leq i < m], \; P^b_i > P^b_j \text{ for } i < j \]
\[ Asks : [(P^a_i, N^a_i) \mid 0 \leq i < n], \; P^a_i < P^a_j \text{ for } i < j \]  

\( P^b_i \) represents the offer price of the \( i^{th} \)-level buy limit order in the orderbook, while \( N^b_i \) is the size of the order. \( P^a_i \) represents the ask price of the \( i^{th} \)-level sell limit order in the orderbook, whereas \( N^a_i \) represents the quantity of that order.

On the simulator, a buy market order of size \( N \) removes sell limit orders from the top of the order book until all \( N \) shares are purchased. Similar to a buy limit order, a sell market order removes buy limit orders from the top of the order book until all shares are sold.

### 2.3 Deep Q-Network (DQ-N) Algorithm

Numerous studies have investigated the application of RL in trading, specifically in the context of optimal order placement. Schaul et al. [4] developed a trading agent for the stock market that outperformed several existing trading strategies using an RL-based approach.

Consequently, the algorithm employed for the task is an RL model. Deep Q-Network (DQN) is the algorithm utilised by the RL model implemented in this undertaking. The DQN algorithm is a deep learning-based RL algorithm that approximates the Q-value function using a neural network. The Q-value function maps the present state and action to the expected cumulative reward. It has been demonstrated that the DQN algorithm is effective at solving a variety of RL problems, including Atari games and robotics tasks.

The objective of Q-learning is to learn the optimal action-value function \( Q(s, a) \), which provides the expected reward for taking action \( a \) in state \( s \) and subsequently adhering to the optimal policy. Selecting the action that maximises the action-value function at each state yields the optimal policy. Q-learning employs an iterative update rule to discover the Q-values by minimising the mean-squared error between the current estimate and the target value, which is the discounted sum...
of future rewards obtained by taking the current action and subsequently adhering to the optimal policy.

Due to the necessity of maintaining a table of Q-values for every conceivable state-action pair, Q-learning becomes impractical for large state spaces. To address this, DQ-N approximates the Q-values using a neural network instead of a table. As input, the neural network receives the state and outputs the Q-value for each action. The algorithm then stabilises the learning process with experience replay.

Experience replay entails storing the agent's experiences (i.e., state, action, reward, next state) in a buffer and randomly sampling experiences from the buffer during training. This helps disrupt the correlation between successive experiences and reduces the update variance.

The DQN algorithm implemented in this undertaking has undergone a few performance-enhancing modifications. An example of such a modification is the implementation of an epsilon-greedy policy, which selects a random action with probability epsilon and the action with the highest Q-value with probability 1-epsilon. The Q-values are then updated utilising the Bellman equation, which estimates the true Q-values based on the current Q-values and the observed reward. This promotes exploration of the environment in the early phases of training and implementation of the policy learned in later stages.

Using a target network, which is a copy of the primary network, reduces the correlation between the target and predicted Q-values. Periodically, the target network is updated with the main network's weights to prevent the target Q-values from becoming overly correlated with the present Q-values and destabilising the learning process.

The following subsections describe the states, actions, and rewards.
2.3.1 States
The DQ-N algorithm's state is an enumeration of features. getfeatures() is employed to calculate the features. The characteristics that comprise the feature list, or state, which are:

1. Mid price: is the average of the highest and lowest bids and is a vital indicator of the current market price.

2. Spread: The difference between the lowest offer price and the highest bid price, which indicates the market's liquidity. The broader the spread, the lower the liquidity, and vice versa.

3. Best bid-ask imbalance: The ratio of the volume difference between the best bid and the best ask to the total volume of the best bid and the best ask. As an indicator of market sentiment, it is pertinent to the RL algorithm.

4. Best bid and ask depth: are the volumes of the highest bid and ask prices, respectively. They are indicators of the market's profundity and its capacity to accommodate large orders.

5. Bid price distance and ask price distance: are the gaps between the highest bid and ask prices and the average price. They are significant because they reveal the market's strength and the degree of support or resistance at various price levels.

2.3.2 Actions
At each timestep, the agent has the option of either maintaining the present position in the asset or placing a market sell order for the entire quantity held.

The action space is therefore discrete and binary, with 0 signifying a hold action and 1 representing a market sell action.
2.3.3 Rewards
This algorithm's reward function is intended to motivate the agent to maximise its cumulative profit over a specified time period. Specifically, the reward at each timestamp $t$ is proportional to the change in the agent's position's value from time $t - 1$ to time $t$. The reward is calculated mathematically as follows:

$$ r_t = p_t - p_{t+1} $$

(2.2)

where $p_t$ and $p_{t+1}$ are the prices of the asset at times $t$ and $t + 1$, and $q_{t-1}$ is the quantity of the asset held by the agent at time $t - 1$.

If the agent places a sell order, the reward for that time step is computed as described previously, and the episode ends. If the agent maintains its current position, the episode advances to the subsequent time step.

If the price difference between consecutive timestamps is sufficient for the DQ-N agent to learn, the reward function should be Equation 2.2. In practise, however, the disparity is typically insignificant. Therefore, $p_{t+100}$ is used to train the agent rather than $p_{t+1}$.

2.3.4 Hyperparameters

2.3.4.1 gamma
Future discount factor for rewards. A value of 1 would indicate that the agent considers all future rewards to be of equal importance, whereas a smaller value assigns future rewards less importance.

2.3.4.2 epsilon
The rate of exploration, which determines the likelihood of choosing a random action.

This begins at 1.0 and is annealed to an ultimate value of epsilon min over time. Initial epsilon values that are high encourage exploration, while ultimate epsilon values that are low encourage exploitation.
2.3.4.3 epsilon_decay
The annealing rate of epsilon over time. A lower value indicates that epsilon decays more slowly, resulting in an extended exploration phase.

2.3.4.4 learning_rate
The learning rate of the optimizer used to alter the weights of the neural network. A higher learning rate enables the agent to learn more quickly, but may result in erratic updates or optimal value overshoot.

2.3.4.5 batch_size
For each training iteration, the number of transitions sampled from the replay buffer.

A larger batch size can improve the stability of updates, but it requires more memory.

2.3.4.6 memory_capacity
Maximum number of transitions that can be stored in the buffer. A larger memory size enables the agent to learn from a greater number of experiences, but may impede down the learning process.

2.3.4.7 target_update_freq
Update frequency (in episodes) for the target network weights. Less frequent updates to the target network can accelerate training, but may result in a less stable policy.

2.3.5 Limitations

2.3.5.1 Exploration-Exploitation Trade-off
The exploration-exploitation tradeoff is one of the model's most significant limitations. While the model is effective at learning a policy for a given environment, it has difficulty exploring new regions of the state space. This is due to the model's reliance on a deterministic policy in which actions are selected based on the highest predicted Q-value. This may cause the model to become stuck in local
optima, failing to explore other regions of the state space that may lead to greater rewards.

2.3.5.2 Learning from Sparse Rewards
The capacity of the DQN model to learn from sparse rewards is another limitation. In some environments, the reward signal may be sparse or delayed, making it challenging for the model to acquire a maximize-rewards policy. This is due to the fact that the model must investigate a large number of state-action pairs to learn an effective policy, which can be difficult when the reward signal is sparse or delayed.

2.3.5.3 Limited Generalization
Additionally, the DQN model can grapple with generalisation. This is due to the fact that the model was trained on a limited number of experiences, which may not be representative of the entire state space. Consequently, the model may struggle to generalise to unobserved states, resulting in poor long-term performance.

2.3.6 Assumptions
The fees levied by exchanges for trading are disregarded. It is supposed that the user incurs minimal trading costs.

2.4 Data Analytics
Our team has developed orderbook analytics tools to collect and analyse pertinent data from Gate.io in order to derive meaningful metrics.

2.4.1 Time-Weighted Bid-Ask Spread
It is the difference between the best limit bid price in the orderbook and the best limit ask price, weighted by the
amount of time that the particular order book snapshot existed on the exchange (See Figure 2.1). The user can determine the space between each data point.

Figure 2.1: Time-weighted Bid ask spread for NEER-USDT on Gate.io. It shows the relationship between the bid-ask spread with time.

2.4.2 Time-Weighted Orderbook Depth
From the mid-price, it is the notional value of all limit orders issued at specific depths in the orderbook. The selected bit rates are 100, 200, and 300 bps. The notional values are then weighted according to the amount of time that the particular order book snapshot existed on the exchange (See Figure 2.2). The interval between each data point is customizable by the user.

2.4.3 Time-Weighted Price Slippage
It is the disparity between a coin's executed price and its mid-price. The slippage is computed based on a fictitious value specified by the user, such as $1,000. It is weighted based on how long a particular order book snapshot existed on the exchange (see Figure 2.3). The interval between each data point is customizable by the user.
Figure 2.2: Time-weighted order book depth for NEER-USDT on Gate.io. It shows the relationship between the notional with time.

Figure 2.3: Time-weighted price slippage for NEER-USDT on Gate.io. It shows the relationship between the slippage with time.

2.5 Microprice

2.5.1 Background

In the context of high frequency trading, the micro price concept has arisen as a promising asset pricing tool. In his 2017 seminal paper, Sasha Stoikov introduced the concept of micro price as a high-frequency predictor of future prices [5]. The research of Stoikov indicates that the best bid and ask quantities contain highly
informative data that can be used to predict the future price of an asset with high precision. The microprice is calculated as a function of the orderbook imbalance and spread using data from the level 1 orderbook.

The limitations of mid-price and weighted mid-price are well-known, with the former being a medium-frequency signal that does not utilise volume, and the latter becoming extremely chaotic as the spread widens. Micro-price, on the other hand, has several advantageous characteristics that make it the preferred method for pricing assets. First, it is a martingale construction, which means it satisfies the martingale property and thus provides a more accurate price forecast. Second, it can be computed incredibly quickly, which is a crucial characteristic for high frequency trading. Lastly, it outperforms mid price and weighted mid price and functions equally well for small and large tick equities.

In recent years, the use of micro price-based algorithms has garnered considerable traction due to their ability to accurately predict the future prices of assets.

The micro price is derived from the highest bid and ask quantities, which are highly informative and provide a more accurate estimate of the assets' fair value. The level 1 orderbook data is used to calculate the micro-price as a function of the orderbook imbalance and spread, making it a dependable and robust instrument for high frequency trading.

In addition, the micro price offers a number of benefits over the traditional mid price and weighted mid price structures. The median price is a signal with a medium frequency that does not utilise the volume, whereas the weighted median price is chaotic as the spread widens. The micro price, on the other hand, is a martingale construction that is quick to compute and outperforms conventional constructions. It is effective for both small and large tick equities, making it a flexible asset pricing tool for high frequency trading.
2.5.2 Input data

The orderbook data from the hour prior to the trading period is essential for predicting the next hour of trading because it reflects recent market developments and trends. Using an extended or shorter time frame for training data may result in inaccurate predictions due to a lack of current and relevant data. A one-hour time frame achieves a balance between incorporating recent market data and capturing market dynamics at their core. The orderbook data is comprised of the YFII-USDT token and the GRAIL-USDT token, as described in sections 2.5.2.1 and 2.5.2.2.

2.5.2.1 YFII-USDT

DFI.MONEY, also referred to as YFII, is a DeFi platform that was introduced in July 2020 as a fork of yearn.finance (YFI), with the primary objective of optimising returns for investors in the DeFi ecosystem while implementing changes proposed in the YFI Improvement Proposal-8 (YIP-8) upgrade plan. In addition to protocol modifications, DFI.MONEY has introduced new products, the most notable of which is the Vault, which seeks to provide investors with the opportunity to earn high yields with minimal risk. The native token of the platform, YFII, is a token with a fixed supply that liquidity providers earn based on their network interaction. YFII is a vital component of the platform's ecosystem because it is used for governance, enabling users to vote on important decisions pertaining to the platform's development. Incentivising users to contribute liquidity to the platform, YFII is also used for staking and farming.

Overall, the YFII token from DFI.MONEY has garnered considerable interest in the DeFi industry due to its potential to generate high returns for investors while mitigating risk. The liquidity of the YFII token is elaborated upon in Figures 2.4, 2.5, and 2.6.

In relation to the orderbook depth (as depicted in Figure 2.4), the results indicate a substantial depth that can be interpreted as a sign of robust liquidity during the observed time period. In particular, the depth is such that the distinctions between bid-ask spreads of 50, 100,
and 200 basis points are negligible. This indicates that market participants can transact at comparable costs across a range of bid-ask spreads, indicating market efficiency.

In addition, the average orderbook depth across all levels is observed to range between 12,000 and 13,000 notional. This degree of market participation and interest is noteworthy, as it contributes to the market's overall efficacy. It also implies that there are sufficient buy and sell orders at different price levels to absorb incoming transactions without causing significant price fluctuations.

In this instance (refer to Figure 2.5), the slippage is minimal, ranging from 0.1% to 0.4% of the mid-price, indicating the asset in question has a high level of liquidity. This conclusion is derived from the fact that the notional value used to calculate the slippage is $500, and the observed low slippage indicates that there is little variance between the expected and actual execution prices of the order.

A low slippage indicates that traders can execute trades at prices that are extremely close to the expected price without substantially affecting the market price as a whole. This finding provides additional support for the claim that the asset is relatively liquid.

Moreover, it is intriguing to note that the slippages for sell and purchase orders move in opposite directions. When the purchase slippage rises, the sell slippage falls, and vice versa.

As shown in Figure 2.6, the bid-ask spread has been observed to be narrow, with an average range of 0.1% 0.2% of the mid-price. This is a significant discovery because it indicates that the asset in question is relatively liquid.

The difference between the highest price a buyer is willing to pay (the bid) and the lowest price a vendor is willing to accept (the ask) is minimal when the bid-ask spread is low. This indicates that dealers are able to transact the asset at comparable prices without experiencing significant price fluctuations.
In conclusion, these findings suggest that the studied market possesses a robust level of liquidity, which is essential for efficient price discovery and trade execution. However, additional research may be required to validate these findings in various market conditions or over a longer period of time.

Figure 2.4: Orderbook depth of YFII-USDT

Figure 2.5: Slippage of YFII-USDT

Figure 2.6: Spread of YFII-USDT
2.5.2.2 GRAIL-USDT
Camelot DEX is a decentralised exchange that operates on the Arbitrum network and provides an automated market maker (AMM) as its primary feature, as well as a number of supplementary functionalities to support the protocol. Notably, Camelot has instituted unique features such as dual-liquidity types, catering to both volatile and stable swaps (WBTC-USDT and USDC-USDT, respectively), and dynamic trading fees, allowing each pool to set its own trading fee that varies based on the trade direction. These features enhance the user experience by providing flexibility in trading and optimising the provision of liquidity.

The representation of staked liquidity pool positions via non-fungible tokens (NFTs) known as spNFTs is an additional innovative feature provided by Camelot. Similar to traditional liquidity provision, these tokens enable users to earn yield and fees, and they can also be locked to increase the yield on the positions. Some pools mandate the utilisation of closed positions for additional yield, thereby incentivizing participants to contribute to liquidity provision.

Camelot has its own native token, GRAIL, which can be converted into xGRAIL. These tokens can be used to generate returns across multiple Camelot plugins, including protocol fee sharing, enhancing pool yields, and accessing the launchpad. These plugins provide additional opportunities for users to increase their returns and partake in the Camelot ecosystem.

Using numbers 2.7, 2.8, and 2.9, one can analyse the liquidity.

The liquidity fluctuates over time, as determined by analysing the depth of the GRAIL-USDT token on Gateio (Figure 2.7). Initially, market liquidity is robust, as indicated by the market's substantial depth. Nonetheless, as time progresses, the liquidity at the 200 basis points level rests at a notional value of 12,000, which is still regarded as adequate liquidity. At 100 basis points, the liquidity resolves at approximately 5,000 notional, which is significantly less than 200
basis points. In addition, at a level of 50 basis points, the liquidity depth is highly variable and rapidly decreases from its initial value.

It is essential to note that these variations in liquidity depth may be influenced by a number of variables, including market conditions, trading activity, and the availability of market participants. It is essential to monitor these trends and adapt trading strategies accordingly in order to maximise returns and minimise risks.

Analysis of the asset's slippage (see Figure 2.8) reveals that purchasing the asset is relatively simpler than selling it. The average buy slippage for a notional value of $500 is approximately 0.6% - 0.4%, which is lower than the average sell slippage of approximately 1.1%.

Numerous variables, such as market conditions, trading volume, and liquidity, can account for the observed difference in slippage between purchasing and selling the asset. Higher slippage during selling may indicate less liquidity or a greater demand for selling the asset compared to purchasing, resulting in a greater price impact when selling.

Slippage trends must be monitored and the trading strategy must be adjusted accordingly to maximise returns and minimise risks. When executing buy or sell orders, traders should consider slippage analysis to minimise price impact, particularly when dealing with larger notional values.

The bid-ask spread of the GRAIL-USDT token is significantly larger than that of the YFII-USDT token, as shown in Figure 2.9. This indicates that trading the GRAIL-USDT token is more expensive, as the difference between the bid and ask prices is greater, and therefore may be more difficult. This indicates that GRAIL-USDT may be classified as a token with low to moderate liquidity.

Observing the bid-ask spread provides valuable insight into the performance of trading algorithms when applied to a token with a wider spread. Trading algorithms are extremely sensitive to market
conditions, and a wider spread can have a significant impact. When designing and implementing a trading algorithm for GRAIL-USDT or any other token with a similar liquidity profile, it is essential to consider the bid-ask spread.

The liquidity profiles of GRAIL-USDT and YFIUSDT encompass a variety of assets and situations. When developing trading strategies and making trading decisions, it is essential to consider the liquidity characteristics of various tokens in order to effectively manage the costs and risks associated with trading.

Figure 2.7: Orderbook depth of GRAIL-USDT

Figure 2.8: Slippage of GRAIL-USDT
2.5.3 Algorithm

In recent years, the use of the microprice as an estimate for the fair price of an asset has become increasingly common, particularly in contexts involving high frequency trading.

The microprice is determined by utilising the data from the level 1 orderbook, which provides information regarding the state of the market for a certain asset at the time of calculation. The microprice is able to provide useful insights into the potential future movement of the asset's price by doing an analysis of the bid and ask quantities included inside the orderbook.

When developing our trading strategy, we make use of the microprice as an indicator for future price changes. This enables us to make judgements regarding the placement of buy or sell orders that are more informed. To be more specific, we construct our trading algorithm based on the data obtained from the orderbook's statistics, which include imbalance and spread, and then we use this data to determine the microprice. We hope that by include the microprice in our trading strategy, we will be able to improve the way in which our orders are executed and, as a result, achieve better outcomes overall.

The Time-Weighted Average Price (TWAP) method is one that has seen widespread application in algorithmic trading, and it serves as the basis for our trading algorithm.
Within our algorithm, we have included the microprice both as an element that plays a significant role in the decision-making process and as a forecast of future price changes. To be more specific, we carry out trades at an interval of one hundred iterations, and the microprice tells us whether to place a limit order or a market order. When the microprice is greater than the current price and we have the intention of selling the asset, we will put a limit order in the hopes that there will be an increase in the value of the asset. If our limit order is satisfied, we will go to the following iteration; but, if it is not satisfied, we will place a market order in order to satisfy our quantity quota (See Figure 2.11). On the other hand, if the microprice is lower than the current price and we have the intention of purchasing the asset, we will put a limit order in the hopes that the price would go down further. When the 100 iterations are complete, if the limit order has not been filled, we will place a market order (see Figure 2.10). This is analogous to the selling process.

Our algorithm enables us to surpass TWAP or at the very least achieve equivalent performance in the event that the microprice forecasts an increase in price. On the other hand, we stick to the traditional TWAP strategy if the microprice does not indicate that it expects an upward movement. With this method, we are able to take advantage of the predictive capacity of the microprice while still preserving the benefits of the tried-and-true TWAP strategy.

The addition to our trading algorithm of the insight that consisted of standardising the difference in price that existed between the microprice and the midprice has proven to be a worthwhile addition. By standardising the signal, we were able to develop a common reference point that was applicable to all assets and trading circumstances. This was made possible by the fact that we standardised it. Because of the way we approached the problem, we were able to develop a signal that was somewhere in the range of [0,1], which made it much simpler for us to analyse the signal and make decisions based on it.
In addition to this, standardising the signal helped us to generalise our code so that it could account for any assessment of the microprice and for a variety of changes in the price of the asset. Because of this, it was much simpler for us to modify our trading algorithm so that it could work with a variety of market conditions and assets. We were able to minimise the complexity of our code and make it more modular by providing a standardised signal. This, in turn, made it easier for us to update and modify the code.

In general, the incorporation of the insight that consisted of standardising the difference between microprice and midprice into our trading algorithm has proven to be a beneficial addition to the programme. It has made our code more adaptable to a variety of market conditions and assets, allowing us to develop a common reference point that can be utilised across all assets and trading scenarios.

Figure 2.10: The algorithm for buying
2.6 Benchmarks
Within the context of this project, the TWAP algorithm serves as the standard against which the performance of models is evaluated. The term time-weighted average price, or TWAP, is a financial word that refers to the average price of a security over a specific amount of time. This algorithm has a long history of serving as the benchmark algorithm in a variety of research papers [6][7].

Chapter 3: Findings
Upon completion, the rewards versus iterations graph is presented in this section, along with an evaluation of the performance of the model and a discussion of the anticipated constraints.

3.1 Output and Performance
The actions that were carried out at each timestamp are what the model is forecasted to provide as its output. These actions could
include putting a market sell order or not placing any order at all. Graphs depicting rewards in relation to iterations will be utilised in order to present the findings. After a significant number of repetitions, more than 500, our expectation is that the rewards will steadily increase as they move up the y-axis.

This would be evidence that the model training is effective and that the agent is learning how to take the appropriate action regardless of the circumstances. We get things started in a naive fashion by simply putting the hyperparameters in place without giving them much thought. After that, we begin gradually adjusting the model's hyperparameters in an effort to improve the performance of the model.

3.1.1 DQ-N Algorithm

3.1.1.1 Initial hyperparameters

Initially, we set the hyperparameters as follows:

\[
\begin{align*}
\gamma &= 0.99 \\
\epsilon &= 1.0 \\
\epsilon_{\text{decay}} &= 0.98 \\
\epsilon_{\text{min}} &= 0.01 \\
\text{batch size} &= 64 \\
\end{align*}
\]  

(3.1)

The set of hyperparameters 3.1 yields a reward-iterations graph (see Figure 3.1) in which rewards are distributed randomly.
To enhance performance, the hyperparameters should be calibrated according to the intended task.

3.1.1.2 Adjusted hyperparameters
Initially, we set the hyperparameters as follows:

\[
\begin{align*}
\text{gamma} &= 0.8 \\
\text{epsilon} &= 1.0 \\
\text{epsilon}\_\text{decay} &= 0.95 \\
\text{epsilon}\_\text{min} &= 0.01 \\
\text{batch}\_\text{size} &= 64
\end{align*}
\]  

(3.2)

The set of hyperparameters 3.2 yields a graph of reward iterations (see Figure 3.2) with more rewards above the x-axis.
3.1.1.3 Further hyperparameter tuning
After more experimentation, we set the hyperparameters as follows:

\[
\begin{align*}
gamma &= 0.6 \\
\epsilon &= 1.0 \\
\epsilon_{\text{decay}} &= 0.93 \\
\epsilon_{\text{min}} &= 0.01 \\
\text{batch size} &= 64
\end{align*}
\] (3.3)

The set of hyperparameters 3.3 yields a graph of reward iterations (see Figure 3.3) in which rewards are distributed at random.
3.1.2 Micro-price
The results of the studies that were carried out demonstrated that micro-price is an accurate estimator of the movement of prices in the future. Under the parameters of our market simulator, the micro-price algorithm has repeatedly demonstrated superior performance compared to TWAP as well as randomly placing market and limit orders. As can be seen in the algorithm below, Micro-price has the ability to generate a competitive advantage over other algorithms that are extensively employed.

3.1.2.1 YFII-USDT trade – 1
In the course of our research, we used a microprice and time-weighted average price (TWAP) algorithm to carry out a back-test in order to evaluate the efficacy of buy and sell limit orders. According to the findings of our historical analysis, the fill rate of buy limit orders was
78.94%, which indicates that these orders were carried out with a significant amount of success.

Despite this, the percentage of sell limit orders that were fulfilled was only 54.16.

In addition, we discovered that the overall amount earned across all of the trades for purchase orders was 40.98 dollars, whereas the total amount obtained for sell orders was just 7.49 dollars.

According to this evidence, the buy orders in our analysis generated far more revenue than the sell orders did. In order to conduct a more in-depth analysis of the performance of these orders, we developed graphs (3.4 and 3.5) that depict the differences between each trade using the microprice algorithm and the TWAP method, respectively. With the use of these graphs (3.4 and 3.5), we were able to graphically analyse the performance of each trade and spot any patterns or trends that emerged.
Figure 3.4: Plot of Rewards-Iteration with first set of hyperparameters.
3.1.2.2 YFII-USDT trade 2

Building on the findings of our earlier research, we carried out a second back-test to assess how well the microprice and time-weighted average price (TWAP) algorithm performed when evaluating the performance of buy and sell limit orders. The outcomes of our investigation showed that the fill rate of the buy limit orders in this second back-test was 60.66%, which was lower than the fill rate that was seen in the first back-test that we conducted. In a manner parallel to this, the percentage of successfully executed sell limit orders was lower in the second back-test, coming in at 46.31.

In addition, our investigation indicated that the total amount won throughout the trades for buy orders was $7.98, which was a considerable amount less than the amount gained in the initial back-
test that we ran. The total amount earned from sale orders was only 0.94 dollars, demonstrating a decline in profitability for both buy and sell orders. This was also the case with the total amount gained from buy orders. Based on these data, it appears as though the performance of buy and sell limit orders utilising the microprice and TWAP algorithm might not be constant regardless of the market conditions that are present.

We created graphs (3.6 and 3.7) to illustrate the differences between each transaction for both the microprice and the TWAP algorithm so that we could have a clearer picture of how well these orders performed and obtain a deeper comprehension of how they worked. These graphs gave us the ability to visually observe the performance of each trade and to recognise any patterns or trends that may have emerged.

![Graph showing price difference per trade](image)

Figure 3.6: Plot of Rewards-Iteration with first set of hyperparameters.
3.1.2.3 GRAIL-USDT trade – 1

In the course of our research, we used a microprice and time-weighted average price (TWAP) algorithm to carry out a back-test in order to evaluate the efficacy of buy and sell limit orders. According to the results that we uncovered, the percentage of purchase limit orders that were successfully executed was 94.39%, showing a high degree of success overall in the process. However, the percentage of sell limit orders that were filled was only 7.76 percent, which indicates that there may be space for improvement in the way these orders are being executed.

In addition, the results of our analysis showed that the total amount earned throughout the trades for buy orders was 29.95 dollars, which was a significant amount more than the amount gained for sale orders, which was only 6.11 dollars. This was the case because buy orders were executed more frequently than sell orders. According to the
results of this research, it appears that the performance of purchase orders was more profitable than the performance of sell orders.

We developed graphs (3.8 and 3.9) to depict the differences between each transaction for both the microprice and the TWAP algorithm so that we could have a deeper comprehension of the performance of these orders. According to the findings of our examination of these graphs, the effectiveness of buy and sell orders was found to vary substantially between different types of trades. While some trades brought about huge gains, others brought about significant losses.

Figure 3.8: Plot of Rewards-Iteration with first set of hyperparameters
3.1.2.4 GRAIL-USDT trade - 2

For the purpose of determining whether or not buy and sell limit orders are effective, our research included a back-test that utilised the microprice and time-weighted average price (TWAP) algorithm. According to the findings of our investigation, the percentage of buy limit orders that were fulfilled was quite high at 90.69 percent, however the percentage of sell limit orders that were fulfilled was quite a bit lower at 28.22 percent.

This would seem to imply that there is opportunity for improvement in the way that sell limit orders are carried out.

The results of our investigation also showed that the overall profit achieved by executing purchase orders was 41.45 dollars, which was a significant amount lower than the profit gained by executing sell
orders, which was 1187.17 dollars. This suggests that executing sell orders may be a more profitable strategy than executing buy orders, despite the fact that the number of successful sell orders was lower, which suggests that there may be a larger risk involved in executing sell orders. This is because the number of successful buy orders was higher. We developed graphs (3.10 and 3.11) to depict the differences between each trade utilising the microprice algorithm as well as the TWAP method so that we could conduct a more in-depth analysis of the performance of these orders. The performance of buy and sell orders varied dramatically between trades, as shown by our examination of these graphs, which showed that some trades resulted in significant profits while others resulted in losses. Some trades produced significant profits while others produced significant losses.

Figure 3.10: Plot of Rewards-Iteration with first set of hyperparameters
Chapter 4: Conclusion

The research that has been done on the topic of optimal order execution in the bitcoin market is the primary subject of this paper. Despite the availability of more sophisticated algorithms such as RL, traders commonly use simpler algorithms like as TWAP and VWAP. This is the primary challenge that this research intends to address. In addition, the published works that are currently available exclusively discuss highly liquid markets such as BTC/USD.

The purpose of this project is to identify the most effective tactic that can be used when making orders in the bitcoin market. The initial step is to train an RL model to either carry out a market sell order or take no action at all, after which the performance of the model is compared to that of competing algorithms. The trading algorithm was developed to
participate in a variety of cryptocurrency marketplaces, each of which exhibits its own unique amount of liquidity and volatility.

The information regarding the market microstructure is utilised as the input data for the specialised RL algorithm as well as the Deep Q-Network (DQ-N) algorithm. In order to retrieve the data, we made use of the websockets package that is included in Python.

The RL model's states, actions, and rewards have all been mapped out by our team. Orderbook data, volatility, order completion %, and a number of other metrics are all examples of potential state variables. The actions that the model would take are to place limit or market orders, depending on the situation. We have not yet settled on a reward function and are still in the process of testing out various functions to determine which one would provide the best results. Our model's overall performance will be measured against the TWAP algorithm, which will serve as the benchmark.

We are able to evaluate which model works the best by comparing the final prices of three different strategies: the TWAP strategy, the market order strategy, and the RL algorithm approach. We are going to see the end result with the help of metrics modules such as confusion matrices and random forest classifiers.

When conducting the evaluation of the model, both positive and negative trends in the market will be taken into consideration. During an upswing, the RL agent would place a limit buy order at a price that was higher than the market price, and during a downtrend, the limit purchase order would be placed at a price that was lower than the market price. Based on the conclusions of this study, it appears that an RL model would be the most appropriate representation of this use case. Because it would demonstrate how powerful algorithms may affect the actual world and generate income, an RL model that is applied to market microstructure data and evaluated against industry standards is valuable.

In addition, the study investigated the application of high-frequency signals, such as micro-price, which has been demonstrated to be an
extremely efficient method for the generation of profits. Another area of interest for this project was the order book analytics tool since it gives traders the ability to analyse order books and make trades based on L3 order book data, which eventually helps them generate alpha and increases their profits from execution.

We were able to identify two drawbacks to using this method: To begin, there is no way to accurately simulate the effect on the market. Second, if we were to use an RL model, we would not get the same level of efficiency as we would with a domain specialised method.

People may adapt our algorithm to operate with new cryptocurrencies as this area of research evolves, and they may test their own algorithms using our market simulator to see how well they perform. It would make it possible for ordinary traders to carry out sophisticated algorithms.
References


Appendices
Appendix A - Project Schedule

<table>
<thead>
<tr>
<th>Date/Time Period</th>
<th>Milestone/Achievements</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-10-2022 to 31-10-2022</td>
<td>Research and implement of market simulator</td>
</tr>
<tr>
<td>01-11-2022 to 15-11-2022</td>
<td>Placing orders on the market simulator and study its results</td>
</tr>
<tr>
<td>15-11-2022 to 10-12-2022</td>
<td>Algorithm (TWAP and VWAP) implementation</td>
</tr>
<tr>
<td>10-12-2022 to 31-12-2022</td>
<td>Researching RL and analysing it along with other available algorithms</td>
</tr>
<tr>
<td>05-01-2023 to 15-02-2023</td>
<td>Writing code for the algorithms used</td>
</tr>
<tr>
<td>15-02-2023 to 12-03-2023</td>
<td>Ranking performance of different types of algorithms and introducing new concept of microprice</td>
</tr>
<tr>
<td>12-03-2023 to 05-04-2023</td>
<td>Connecting all the relevant part together to make it workable and functional</td>
</tr>
<tr>
<td>05-04-2023 to 18-04-2023</td>
<td>Testing and debugging the code</td>
</tr>
</tbody>
</table>

A1. Timeline of the Project
## Appendix B - Market Simulator

<table>
<thead>
<tr>
<th>t</th>
<th>s</th>
<th>bid_px</th>
<th>bid_qty</th>
<th>ask_px</th>
<th>ask_qty</th>
<th>...</th>
<th>ask_5_px</th>
<th>ask_5_qty</th>
<th>bid_10_px</th>
<th>bid_10_qty</th>
<th>ask_10_px</th>
<th>ask_10_qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BTC_USD</td>
<td>22774.01</td>
<td>0.003</td>
<td>22789.57</td>
<td>0.8823</td>
<td>...</td>
<td>22835.79</td>
<td>5.7362</td>
<td>22905.95</td>
<td>8.2075</td>
<td>23047.21</td>
<td>5.8779</td>
</tr>
<tr>
<td>1</td>
<td>BTC_USD</td>
<td>22774.01</td>
<td>0.003</td>
<td>22789.57</td>
<td>0.8823</td>
<td>...</td>
<td>22835.79</td>
<td>5.7362</td>
<td>22904.61</td>
<td>4.9868</td>
<td>23047.21</td>
<td>5.8779</td>
</tr>
<tr>
<td>2</td>
<td>BTC_USD</td>
<td>22774.01</td>
<td>0.003</td>
<td>22789.57</td>
<td>0.8823</td>
<td>...</td>
<td>22835.79</td>
<td>5.7362</td>
<td>22903.27</td>
<td>9.1864</td>
<td>23047.21</td>
<td>5.8779</td>
</tr>
<tr>
<td>3</td>
<td>BTC_USD</td>
<td>22774.01</td>
<td>0.003</td>
<td>22789.57</td>
<td>0.8823</td>
<td>...</td>
<td>22834.72</td>
<td>5.6799</td>
<td>22902.27</td>
<td>9.1864</td>
<td>23058.63</td>
<td>4.9456</td>
</tr>
<tr>
<td>4</td>
<td>BTC_USD</td>
<td>22774.01</td>
<td>0.003</td>
<td>22789.57</td>
<td>0.8823</td>
<td>...</td>
<td>22847.21</td>
<td>5.8779</td>
<td>22903.27</td>
<td>9.1864</td>
<td>23058.63</td>
<td>4.9456</td>
</tr>
</tbody>
</table>

[5 rows x 82 columns]

Figure B.1: A screen capture of the market simulator. The orderbook is for BTC/USD and the depth is 5 levels. There is a change whenever a buy/sell order is placed.
Appendix C – UI Design (Suggestive)

Figure C.1 Wireframe for the place order tab. The user can choose between a market and a limit order
Figure C.2: Wireframe for the market order tab. The user can place a market order after choosing from a buy/sell order and giving a notional and asset.
Figure C.3: Wireframe for the limit order tab. The user can place a limit order after choosing from a buy/sell order and giving a price, quantity and asset.
Figure C.4 Wireframe for the history with the order details with the option of cancelling incomplete orders