Intra-exchange Cryptocurrency Arbitrage Bot

Eric Han
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Recommended Citation
DOI: https://doi.org/10.31979/etd.6xze-y9xu
https://scholarworks.sjsu.edu/etd_projects/655
Intra-exchange Cryptocurrency Arbitrage Bot

A Project
Presented to
The Faculty of the Department of Computer Science
San José State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Eric Han
December 2018
The Designated Project Committee Approves the Project Titled

Intra-exchange Cryptocurrency Arbitrage Bot

by

Eric Han

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2018

Dr. Thomas Austin  Department of Computer Science
Dr. Mark Stamp  Department of Computer Science
Dr. Jon Pearce  Department of Computer Science
ABSTRACT

Intra-exchange Cryptocurrency Arbitrage Bot

by Eric Han

Cryptocurrencies are defined as a digital currency in which encryption techniques are utilized to regulate generation of units of currency and verify the transfer of funds, independent of a central governing body such as a bank. Due to the large number of cryptocurrencies currently available, there inherently exists many price discrepancies due to market inefficiencies. Market inefficiencies occur when the price of assets do not reflect their true value. In fact, these types of pricing discrepancies exist in other financial markets, including fiat currency exchanges and stock exchanges. However, these discrepancies are more significant in the cryptocurrency domain due to the low levels of government regulation, higher amounts of speculation, and human behaviors driven by investors seeking profit. These types of pricing discrepancies can be eliminated to some extent by executing arbitrages, which are defined as a sequences of trades beginning and ending with the same asset which result in more of that asset at the end of the trading sequence. Through executing arbitrages, the market should become more efficient.

This project was an attempt to execute intra-exchange arbitrage on the well-known cryptocurrency exchange Binance and generate profit, and as a side effect make the cryptocurrency exchange market more fluid. Although the project did not record phenomenal profits, it did successfully generate several hundred dollars over the course of several months, independent of market fluctuations.
ACKNOWLEDGMENTS

I want to thank my family: KY Han, Sue-Jane Han, and Jason Han for their undying support of me through this period of my life, along with Professor Thomas Austin, Professor Mark Stamp, Professor John Pearce, and Professor Katerina Potika from San Jose State University Computer Science Faculty. Their outstanding pedagogy has inspired me as a student to try novel ideas, no matter how absurd they sound.
I would like to dedicate this Project to all my friends and family who are going through struggles and feel like there’s no light at the end of the tunnel. Remember, any dream you have written down with a date becomes a goal. A goal broken down into steps becomes a plan. A plan backed by actions makes your dreams come true.
# TABLE OF CONTENTS

DEDICATION ........................................... vi

CHAPTER

1 Introduction ......................................... 1
   1.1 Overview ........................................... 1
   1.2 Research Objective ................................. 1

2 Background and Related Work .................... 3
   2.1 History and Background ........................... 3
   2.2 Types of Currency Arbitrage ....................... 4
   2.3 Current Cryptocurrency Arbitrage Implementations and Challenges 5
   2.4 Bellman-Ford and Asset Arbitrage ................ 8

3 System Architecture ................................ 11
   3.1 Technical Approach/Methodology .................. 11
   3.2 Implementation .................................... 12
      3.2.1 Program Setup (List of classes and other files used) ... 12
      3.2.2 Libraries Used ................................ 14
      3.2.3 Miscellaneous Data Structures Used ............ 14

4 Experiments and Results .......................... 16
   4.1 Bitcoin Only Experiment .......................... 16
   4.2 “amountBuffer” = 1 ................................ 19
   4.3 “amountBuffer” = 2 ............................... 20
   4.4 “amountBuffer” = 3 ............................... 21

vii
4.5 “amountBuffer” = 4 ............................................. 22

5 Discussion and Observations ................................. 28

5.1 Important Topics .............................................. 28

6 Conclusion and Future Work ................................. 30

LIST OF REFERENCES ........................................... 32
LIST OF FIGURES

1 Table showing Additional Bitcoins, Percent Increase, and Percent Increase from Beginning vs Arbitrage Executions. . . . . . . . . . 17
2 Total Account Bitcoin Value vs. Arbitrage Sequences Executed. . 18
3 Table showing Profitability with amountBuffer = 1. . . . . . . . 19
4 Graph showing arbitrage results with amountBuffer = 1. . . . . . 20
5 Table showing Profitability with amountBuffer = 2. . . . . . . . 20
6 Graph showing arbitrage results with amountBuffer = 2. . . . . . 21
7 Table showing Profitability with amountBuffer = 3. . . . . . . . 22
8 Graph showing arbitrage results with amountBuffer = 3. . . . . . 22
9 Table showing Profitability with amountBuffer = 4 (Run 1). . . . 23
10 Graph showing arbitrage results with amountBuffer = 4 (Run 1). 24
11 Table showing Profitability with amountBuffer = 4 (Run 2). . . . 24
12 Graph showing arbitrage results with amountBuffer = 4 (Run 2). 25
13 Table showing Profitability with amountBuffer = 4 (Run 3). . . . 25
14 Graph showing arbitrage results with amountBuffer = 4 (Run 3). 26
CHAPTER 1

Introduction

1.1 Overview

Cryptocurrencies present many new opportunities that can be attributable to its utilization of blockchain technology. With technologies such as smart contracts, distributed ledger, and currency virtualization behind it, it is no wonder that there are over 4000 cryptocurrencies (or alt-coins) in existence today [1]. As a side effect of this explosion in number of cryptocurrencies, a similar rise in cryptocurrency exchanges has also occurred. Coinbase, Kraken, Bitfinex, Bitrex, and Poloniex are just a few of the many existing cryptocurrency exchanges [2]. These sites provide Application Programming Interfaces (APIs) to their users so they can trade programmatically. Due to the existence of these APIs, market inefficiencies present in cryptocurrency markets can be capitalized on through trades that execute on these inefficiencies.

In fact, this technique, known as arbitrage, has been performed by banks and financial institutions for many years. Unfortunately, access to a high availability API often requires significant financial investment and access is typically only given to banks and financial institutions such as investment houses [3]. The main research question this project seeks to answer is: *Can arbitrage techniques be applied to cryptocurrency markets successfully?*

1.2 Research Objective

The research objective is to develop an automated bot that can successfully find and execute arbitrage opportunities, and ultimately yield profits on the Binance exchange and make the market more efficient. I propose to apply this arbitrage technique to Binance, currently the largest cryptocurrency exchange in the world, because I believe that larger price discrepancies exist due to the high number of currencies available, at the time of this writing over 150, as well as the inherent
volatility present in cryptocurrency values.

Building the foundation for a public-use library for an intra-exchange arbitrage bot will be part of the research objective. Ideally, if one single application could make profits, then the objective can be extended to experiment running multiple instances of the same application to extend profits and further stabilization of the cryptocurrency-exchange toward market efficiency. If the arbitrage bot works on Binance, steps can be taken to abstract the code so it can be applied to other exchanges as well.
CHAPTER 2

Background and Related Work

2.1 History and Background

Currency arbitrage involves buying and selling currency pairs from different brokers to take advantage of their different spreads. The currency-spread is defined as the difference between “bid” and “ask” prices of an asset. “Bid” refers to what a buyer offers to pay for a specific asset, while “Ask” refers to the selling price of an asset offered by a seller. The size of the currency spread is often used to measure the liquidity and efficiency of the market. A higher currency-spread means the market is less efficient, while a lower currency-spread indicates a more efficient market.

In the fiat currency market, banks often use arbitrage to take advantage of pricing discrepancies between multiple currency pairs to make a profit. Bots that trade on conventional traditional markets such as Bloomberg and NASDAQ exist but are available exclusively to investment houses and brokers. These conventional market bots need access to exchange data from the market. This exchange data is typically not available to laymen [4].

The transparent nature of the blockchain gives cryptocurrency traders access to an exchange’s order book and design trading bots that act on this data. The blockchain is essentially a distributed ledger with all transaction histories, maintained by users and miners of the network. As a result of this, there are already many bots that target cryptocurrency exchanges, such as “Crypto Trader”, “Haasbot”, “Zenbot”, “Gekko”, and “BTCRobot.” Many of these bots have a monthly subscription fee ranging from $60USD to $3,500USD. (At the time of this writing Bitcoin was over 8000USD) Most utilize an inter-exchange arbitrage strategy [5].
2.2 Types of Currency Arbitrage

As a result of the large number of cryptocurrency exchanges, two types of arbitrages can be performed to capitalize on market inefficiencies. The first is “inter-exchange” arbitrage, and the second is “intra-exchange” arbitrage. The former refers to executing arbitrage sequences within an exchange, while the second means executing arbitrages across different exchanges. Each has its own benefits and challenges, which will be discussed below.

• The first technique, inter-exchange arbitrage, was explored by Norry [5] in his article on Bitcoin trading bots. This type of arbitrage surveys cryptocurrency prices on many different exchanges, and finds pricing discrepancies between them. When a pricing discrepancy is found, a lower priced cryptocurrency will be bought on one exchange, then transferred and sold on the other. By doing these arbitrages across many different exchanges, the pricing of cryptocurrencies should stabilize more to a consistent market value, and the executor of the arbitrages can gain profits directly linked to the pricing differences found [4].

• A second type of arbitrage technique, which is similar to the first one, also takes advantage of pricing spreads between two exchanges. In the trading bot “Blackbird,” when an spread is discovered that is large enough to cover fees of short-selling and buying on the two respective exchanges, a short-sell is performed on the exchange where the price is higher, and a long is executed on the exchange where the price is lower. When the two prices eventually converge, the positions are settled (sold for the long position, and bought in the short-sell position). The net profit is basically half the spread minus the trading fees, multiplied by the volume of cryptocurrency traded [6]. This technique claims to be market-neutral in that it doesn’t expose the user to any risks associated with market fluctuations. It also eliminates the need to transfer assets between
different exchanges.

- A third technique was discovered while the exploring MIT homework set solutions by Demaine and Goldwasser [7]. This third technique, “intra-exchange” arbitrage, is basically making a sequence of trades starting and ending with a specific currency, but you end up with more of the starting currency. The Bellman-Ford algorithm [8] can be used to discover pricing discrepancies in an exchange, and these arbitrage opportunities can then be executed by a bot. This is the type of arbitrage explored in this paper. [7]

2.3 Current Cryptocurrency Arbitrage Implementations and Challenges

Osipovich and Jeong [9] discuss the case of Stefan Qin, a 21-year-old Australian based in California, who has built a business out of cryptocurrency arbitrage. In 2016, he founded Virgil Capital, a hedge fund specializing in cryptocurrency arbitrage. He put his studies at San Francisco’s Minerva Schools on hold to run the fund, which returned over 400% last year after fees and now manages $23.5 million. [9] Qin is doing inter-exchange arbitrage, or buying on one exchange, and selling on another. This technique presents several challenges. The most significant challenge is withdrawal times. Because when we buy a currency on an exchange, and want to withdraw it to a different exchange, it needs to be processed by the blockchain and mined. Unless the miner’s fee is high, it is likely that the transaction will take a significant length of time, after which the arbitrage opportunity would disappear. The other challenge is that you often need significant amounts of capital to really make any significant impact on equalizing the rates on the two target exchanges.

In an article written by “scrawl” (username), he says that intra-exchange arbitrage, presents its own set of challenges. First off, many exchanges have a rate-limit on their API, which limits the number of API requests you can make on it for a given time
period. These rate-limits vary from exchange to exchange, but they tend to start from around 60 requests a minute, and may go up to 120 requests per minute [10]. A second challenge this technique presents is the very small window in which you have to execute these arbitrages. In the FOREX market, arbitrage opportunities exist for at most one second, by which time the opportunity has already been taken advantage of [3]. A third challenge is that in cryptocurrency exchanges, a flat commission fee is charged for each trade. These fees can differ from exchange to exchange, but tend to range from 0.05% to 0.5% per trade. These are all factors that need to be taken into consideration when building an arbitrage bot [5].

From Norry’s article on cryptocurrency trading bots, many implementations of trading bots currently exist on the market. “Blackbird”, “Haasbot”, “Zenbot”, “Gekko”, “CryptoTrader”, and “3Commas” are a several available cryptocurrency trading bots available on the market currently.

- “Blackbird” - Blackbird is a supposedly an inter-exchange, market neutral arbitrage bot written in C++ because it actually doesn’t sell, but short-sells a currency on one exchange when a significant price difference is observed. When the prices eventually equalize, it settles the position and reaps in the profit. In addition, The buy/sell and sell/buy trading activities are done in parallel on two different exchanges, independently. Advantage: no need to deal with transfer latency issues [6].

- “Haasbot” - Haasbot offers 3 different pricing plans- one for beginners at a price of 0.073BTC, one for advanced traders at 0.208BTC, and one for growing investors priced at 0.127BTC. These prices are for one year. Haasbot is written in C# and offers a many customizations for bot trading strategies. Haasbot can use technical indicators like RSI, MACD, and Fibonacci. There are proprietary safeties and insurances to keep your investments safe. There is also an auto-tune
feature you can use to optimize your trading strategy. Historical/Real time testing, advanced notifications and reporting are also available, and the platform is developer friendly: meaning you can write your own code to develop and customize your own bots. Most importantly, the platform offers a plethora of technical indicators you can use to your advantage when developing a bot trading strategy [11].

- “Zenbot” - Zenbot is a command-line inter-exchange cryptocurrency trading bot using Node.js and MongoDB. Features it includes are: Fully-automated technical-analysis-based trading approach, full support for GDAX, Poloniex, Kraken, Bittrex, Quadriga, Gemini, Bitfinex, CEX.IO and Bitstamp; plugin architecture for implementing exchange support, or writing new strategies; simulator for backtesting strategies against historical data; “Paper” trading mode which operates on a simulated balance while watching the live market; configurable sell stops, buy stops, and (trailing) profit stops; and flexible sampling period and trade frequency - averages 1-2 trades/day with 1h period, 15-50/day with 5m period [12].

- “Gekko” - Gekko is a free and open source inter-exchange Bitcoin TA trading and backtesting platform that connects to popular Bitcoin exchanges. It is written in JavaScript and runs on Node.js. It offers a “Paper” Trader and “Tradebot” that performs actual trades. Gekko offers plug-in support and the ability to develop your own trading strategies. Gekko supports 3 different exchanges (including Bitfinex, Bitstamp and Poloniex) [13].

- “CryptoTrader” - CrytpoTrader is an cloud-based inter-exchange arbitrage bot that requires no software installation. All major crypto-currency exchanges are supported for both backtesting and live trading. It also offers a strategies marketplace where strategies can be bought and sold. It offers backtesting
trading strategies to see how a chosen strategy would work in different market conditions [14].

• “3Commas” - 3Commas is a trading platform that offers several interesting features, including concurrent stop-loss and take-profit trades. It also includes a feature called Trailing features which allows you to rake in profits at a higher threshold than you initially set. It also has long (longer time frame) and short (shorter time frame, typically day trading) bots available. It also offers a QFL bot that performs well in stable markets. The QFL trading strategy is another version of trading based on "price support" and the focus on finding "dead cat bounce" trading opportunities. It also offers a composite bot option which lets you have a list of coins you want the bot to trade and it will then manage your balance automatically. This option allows for optimal balance usage, and is much easier to use then multiple usual bots. [15].

It is not known whether or not these trading bots can discover or execute on arbitrage opportunities. Most bots come with the option to design your own trading strategies and even provide test environments to test your trading bot strategy out. However, it was noted by Norry that some of the trading bots with better reviews would require up to 0.32BTC (2048 USD at the time of this writing) per month licensing fee [5, 10].

2.4 Bellman-Ford and Asset Arbitrage

In this section, we will go over the mathematical foundations behind calculating arbitrage opportunities, which specially utilize the negative-cycle detection property of the Bellman-Ford algorithm. The Bellman-Ford algorithm is a shortest-paths algorithm similar to Dijkstra’s. It utilizes relaxation to find the shortest distance from any one node to another node in a connected graph. Note that the graph we are dealing with in this thesis is not fully-connected, e.g. not every node is connected
directly with another. The nodes are typically connected through 4 main nodes, “BTC” (Bitcoin), “ETH” (Ethereum), “BNB” (Binance Coin), and “USDT” (Tether: Cryptocurrency backed 1:1 by the US Dollar - Not Verified!)

In the Homework 7 solutions by “Demaine” and “Goldwasser” [7], they presume a situation where there is a suitable weighted, directed graph $G = (V, E)$, which we form as follows. $G$ is composed of $n$ vertices which comprise $V$, this is the number of cryptocurrencies available to trade. An edge $e_{i,j}$ from $v_i$ to $v_j$ and an edge $e_{j,i}$ from $v_j$ to $v_i$ exist if there is a trading pair between two vertices, and these edges comprise $E$. The full set of edges, $E$ and vertices, $V$, comprise our graph, $G$. An arbitrage opportunity is defined as a case where:

$$R_{i_1, i_2} \times R_{i_2, i_3} \times R_{i_3, i_4} \times \cdots \times R_{i_{k-1}, i_k} \times R_{i_k, i_1} > 1$$

where $R_{i, j}$ is the exchange rate going from a currency $i$ to a different currency $j$. They further note that this scenario is only true when:

$$\frac{1}{R_{i_1, i_2}} \times \frac{1}{R_{i_2, i_3}} \times \frac{1}{R_{i_3, i_4}} \times \cdots \times \frac{1}{R_{i_{k-1}, i_k}} \times \frac{1}{R_{i_k, i_1}} < 1$$

Taking an advantage of the multiplicative to addition property of logarithms, they note that if you take the logarithm of both sides you can instead represent this condition as a sum:

$$\ln \left( \frac{1}{R_{i_1, i_2}} \right) + \ln \left( \frac{1}{R_{i_2, i_3}} \right) + \ln \left( \frac{1}{R_{i_3, i_4}} \right) + \cdots + \ln \left( \frac{1}{R_{i_{k-1}, i_k}} \right) + \ln \left( \frac{1}{R_{i_k, i_1}} \right) < 0$$

Using this intuition, we can then represent our an edge weight from $v_i$ to $v_j$ simply as:

$$e_{i,j} = \ln \left( \frac{1}{R[i, j]} \right) = -\ln R[i, j]$$

[7]

Thus, this representation allows us to calculate, depending on current market exchange rates polled, all the negative cycles present and also the most profitable
one. The arbitrage sequence is determined by tracing the predecessor when a negative cycle is found, and a set can be handily used to determine what the exact sequence is. Although the run-time of Bellman-Ford is $f(n) = n^3$, our $n$ is small enough that the cubic property of the run-time does not really impact program performance; rate-limiting would by an exchange’s API would be a bigger concern than the run-time of the Bellman-Ford algorithm in our program. From my experience the Binance REST API is not rate-limiting. I have polled Binance’s REST API every 0.0001ms and did not get any errors.
CHAPTER 3
System Architecture
3.1 Technical Approach/Methodology

An object-oriented approach to this problem was taken. Currencies were represented as vertices and exchange rate pairs as edges. The Bellman-Ford algorithm, a single-source shortest-path algorithm that can detect negative cycles, was utilized to discover whether or not arbitrage opportunities exist on the Binance exchange, and when they did show up, an execution on the most profitable one was made. Steps to implement this are detailed below:

1. Poll the REST API Binance to obtain current active trading pairs and their respective current exchange rates.

2. Dynamically generate the vertices and edges from data obtained in step 1.

3. A graph is constructed from the edges and vertices from step 2. Bellman-Ford will be executed on these edges and vertices.

4. Store the highest profitable trade sequence generated from Bellman-Ford as our “best arbitrage sequence”.

5. Execute trades on the “best arbitrage sequence” stored in previous step with the Binance API.

6. Write following values to .csv file immediately after trade, which includes:
   - Time-stamp
   - Arbitrage sequence
   - Portfolio BTC balance
   - Snapshot value (at current exchange rates- exchangeRatesMid)
   - Actual portfolio value (calculated with exchangeRatesMid)
• “amountBuffer” value
• “sigDigBuffer” value
• Exchange (Binance in this project)

7. Repeat previous steps indefinitely.

The steps outlined above should steady stream of profits while making the Binance exchange more liquid and efficient.

3.2 Implementation

This project was implemented in Java(1.8+).

3.2.1 Program Setup (List of classes and other files used)

- Main - Used to execute interaction with Binance’s REST API. Has static HashMaps of exchange rates, code that interacts and extracts data from the API for construction of our graph for use by Bellman-Ford. All dialogs for user interaction are also in the Main class.
- Trader - Used for calculating account balances (without arbitrage and holding), doing actual currency conversion from either BTC/ETH to all others, or the other way around, determining amount and pricing for each trade in a arbitrage sequence, and executing trades. Also used for taking account snapshots.
- ShouldTrade - This is a class used by the Trader class to filter out trades that had an amount requiring a precision of significant digits < 2. I tried higher values than 2 but no trades were executed. This makes sense because most pairs had a requirement of 2 or less for amount precision.
- Trade - Used to return a LimitOrder object provided by XChange’s library. This LimitOrder object has a price that is determined by the best sell price if we are "buying", and the best "bid" price if we are selling. The amount is determined through a dialog when the program first runs, but I typically
used 0.0025BTC ≈ $16USD at the time of this writing. 1BTC ≈ $6400USD in November, 2018. I chose this value of Bitcoin to use because it is slightly over the minimum order value, which is approximately $10 USD. We want to use the least amount of BTC available, so that all our orders will fill immediately and not be left as an open order to be filled later. As the total value of a trade goes up, the likelihood that it will not be filled immediately goes up. I noticed good results with about $15 USD. It is interesting to note, however, that all my open trades did fill. Some took a few months, but they all did complete.

- **Vertex($v_i$)** - Used to model each cryptocurrency as a node in a graph

- **Edge($e_{i,j}$)** - Used to model exchange rates between these cryptocurrency nodes.

  *Note that when creating the Edge class, we use the best buy price for going from one currency to another (e.g. BTCUSDT symbol, going from BTC to USD), since that would be the price we could immediately sell BTC for USDT at. Conversely, we would use the best sell price when buying a currency with another, since that would be the cheapest we could immediately buy BTC with USDT in this example at. (e.g. BTCUSDT symbol, going from USDT to BTC)*

- **Graph($G$)** - Used to model the collection of Vertices and Edges, and execute the Bellman-Ford algorithm. Also used to find the negative cycle with the largest weight (highest profitability), and find the path associated with that cycle. When tracing the route with predecessor, remember to reverse the route since we are tracing backwards.

- **CurrencyConverter** - Used to convert currencies between each other at current exchange rates, and ensure correct precision (number of significant digits) in price and quantity fields when creating limit orders. I read the number of significant digits from a .csv file but I know there is a REST API endpoint provided by Binance that provides this data.
Utilities - Used to log data in results .csv file.

- BinanceTradingRule-Master.csv - used to store the precision of quantity
- BinanceTradingRule-MinPrice.csv - used to store the precision for pricing
- binanceConfig.properties - This file stores your apiKey and apiSecret, make sure not to push it to Github! (i.e. store it outside your project folder)

### 3.2.2 Libraries Used

- XChange by Knowm - This was a very handy library that provided easy to use classes and builder methods for interaction with Binance’s API. They have implemented their library for more than 60 exchanges, and their code can be viewed on Github at XChange Github Repository.
- GSON - a JSON interaction library by Google. This library made it easy to parse and interact with JSON data returned by Binance’s REST API.
- Unirest - a simple Java library for making HTTP verb requests, such as GET and POST in this project.
- XChange-stream - a Java library allowing for interaction with websockets. Allows you to subscribe to updates in different websocket channels, such as ticker, trades, and orderbook, and also unsubscribe from a channel.

### 3.2.3 Miscellaneous Data Structures Used

- ArrayList<Vertex> vertices - Used to store all the cryptocurrencies available on the Binance exchange.
- ArrayList<Edge> edges - Used to store all the edges and their weights calculated for use in Bellman-Ford.
- HashMap<String, Double> exchangePrices - This would store the prices for an instant sell (highest buy), and instant buy (lowest sell).
- HashMap<String, Double> exchangeRates - This was used by the Currency-
Converter class to make conversion rates between currencies easier and more intuitive.

- HashMap<String, Double> exchangeRatesMid - This was used to get an accurate portfolio value. I noticed that I couldn’t use exchangeRates because it would often underestimate my account value. When I averaged the lowest "ask" and highest "sell", and stored it in exchangeRatesMid, I obtained a much more accurate picture of my portfolio value. This was verified by checking with my portfolio balance on Binance.

- LinkedHashMap<String, Integer> sigDigs - This was used to store the precision required for the quantity when creating LimitOrder objects to execute.

- LinkedHashMap<String, Integer> sigDigsForPricing - This was used to store the precision required for the price when creating LimitOrder objects to execute.

- HashMap<String, Edge> edgeMap - This was used in construction of edges

- HashMap<String, Vertex> vertexMap - This was used in construction of vertices.

- HashSet<Vertex> setOfVertices - This was used in construction of vertices.

- HashSet<Edge> setOfEdges - This was used in construction of edges

- Properties prop - used to store apiKey and apiSecret for authentication to Binance REST API.
CHAPTER 4

Experiments and Results

In the table below, it shows the ending amount of bitcoins after each arbitrage
sequence is executed. I chose to execute trades starting and ending with bitcoin
because I felt this strategy would be a good way to verify whether the arbitrage was
working or not. As you can see from the table, the average amount of bitcoin increased
a little less than 0.1% after each arbitrage sequence. This makes sense, given that
the trading commission Binance takes is approximately 0.05% for each trade made.
This commission was calculated before depending on the length of the trade sequence.
For example, for 4 trades, approximately 0.2% (4 * 0.05) would be initially added as
commission to the potential arbitrage; we will call this “potential arbitrage”. After
adding the commission to the arbitrage ratio opportunity, an additional parameter
“amountBuffer” was then added on top of this “potential arbitrage”, to compensate
for the limitations on precision for the price and amount fields imposed by the Binance
REST API. Different values were tested for “amountBuffer” and the results will be
shown below. “AmountBuffer” is an integer value used in the Trader class, though
decimal values would work too; it adds a buffer on top of the commission for a wider
arbitrage profit “space”. Through my experimentation, I discovered that Binance is a
highly liquid cryptocurrency exchange. However, arbitrage opportunities still do exist
and can be capitalized on.

4.1 Bitcoin Only Experiment

After verifying that the arbitrage sequence is correct in execution, I removed
the “BTC” only filter so that I would be able to execute on significantly more
opportunities.

I collected data using different values for “amountBuffer” and “sigDigBuffer”.
“amountBuffer” refers to the buffer on top of the estimation of the trading fees Binance
Figure 1: Table showing Additional Bitcoins, Percent Increase, and Percent Increase from Beginning vs Arbitrage Executions.

<table>
<thead>
<tr>
<th>Bitcoin With Arbitrage</th>
<th>Additional Bitcoins</th>
<th>Percent Increase</th>
<th>Increase Ratio from Beginning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03130317</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0.03134911</td>
<td>0.00004594</td>
<td>0.147</td>
<td>1.001467583</td>
</tr>
<tr>
<td>0.0313673</td>
<td>0.00001819</td>
<td>0.058</td>
<td>1.002048674</td>
</tr>
<tr>
<td>0.03140468</td>
<td>0.00003738</td>
<td>0.119</td>
<td>1.003242803</td>
</tr>
<tr>
<td>0.03141825</td>
<td>0.00001357</td>
<td>0.0432</td>
<td>1.003676305</td>
</tr>
<tr>
<td>0.03142991</td>
<td>0.00001166</td>
<td>0.0371</td>
<td>1.004048791</td>
</tr>
<tr>
<td>0.03144331</td>
<td>0.0000134</td>
<td>0.0426</td>
<td>1.004476863</td>
</tr>
<tr>
<td>0.03145775</td>
<td>0.00001444</td>
<td>0.0459</td>
<td>1.00438158</td>
</tr>
<tr>
<td>0.03146342</td>
<td>0.00000567</td>
<td>0.018</td>
<td>1.00511929</td>
</tr>
<tr>
<td>0.03147615</td>
<td>0.00001273</td>
<td>0.0405</td>
<td>1.005525958</td>
</tr>
<tr>
<td>0.03149407</td>
<td>0.00001792</td>
<td>0.0569</td>
<td>1.006098424</td>
</tr>
<tr>
<td>0.03156073</td>
<td>0.00006666</td>
<td>0.212</td>
<td>1.008227921</td>
</tr>
<tr>
<td>0.03160409</td>
<td>0.00004336</td>
<td>0.137</td>
<td>1.009613084</td>
</tr>
<tr>
<td>0.0316491</td>
<td>0.00004501</td>
<td>0.142</td>
<td>1.011050957</td>
</tr>
<tr>
<td>0.03168059</td>
<td>0.00003149</td>
<td>0.0995</td>
<td>1.012056926</td>
</tr>
</tbody>
</table>

Average Percent Increase Per Trade: 0.08562142657

Different values for “amountBuffer” were used, ranging from 1 to 6. “sigDigBuffer” refers to precision we would like to have on our trades. For ‘sigDigBuffer’, a value of 2 would mean that any symbol with a precision less than 2, i.e. 1 and 0, would be filtered out. Of course, we would want to increase the precision of our arbitrage execution, but by increasing the “sigDigBuffer”, but the number of arbitrage opportunities found is decreased. By decreasing “sigDigBuffer” to 1 or 0, the number of arbitrages executed increase. The precision of the arbitrage is directly proportional to “sigDigBuffer”. Increasing “sigDigBuffer” past of a value of 2 results in nearly no trades being executed. Hence, for Binance, a “sigDigBuffer” of 2 is necessary to execute intra-exchange arbitrage. The tables below show the results for runs with different values for “amountBuffer” and “sigDigBuffer”. Note that these results are specific to Binance and would likely be different for a different exchange.
After verifying that my arbitrage logic was correct, I decided to remove my BTC only filter and to execute on all arbitrage sequences. An interesting question came into mind: How would I capture whether or not I was making profits through arbitrage if I was executing on all coins? To address this issue, I decided to take a snapshot of my balances at a certain point in time, consisting of a HashMap of coins and their respective balances. I would then convert all those coins into Bitcoin at the current market rate on the exchange, using a data structure called exchangeRatesMid (This is the average of the best sell and best buy price). This calculation gave me a reasonable estimate of my account value if I held onto my coins and had not executed arbitrage. I could then compare my actual Binance account value, using the same rates stored in exchangeRatesMid, to this snapshot value to gauge whether or not I was gaining or losing value through executing arbitrages. In using this methodology, I could also negate market movements of the highly volatile cryptocurrency market and obtain a
neutral picture of the success of my program.

The following section shows 6 tables and graphs demonstrating the impact of “amountBuffer” on the profitability of my arbitrage program. The line graphs shown simply shows whether or not my portfolio value increases or decreases depending on the value of “amountBuffer”. The following table is for “amountBuffer” = 1.

### 4.2 “amountBuffer”= 1

<table>
<thead>
<tr>
<th>Date and Time</th>
<th>Arbitrage Sequence</th>
<th>Portfolio Value (Arbitrage)</th>
<th>Portfolio Value (Btc)</th>
<th>Difference</th>
<th>Difference Adjusted to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018/09/05 12:26:04</td>
<td>BTC ZEN ETH STRAT BTC</td>
<td>2549.570718</td>
<td>2519.160255</td>
<td>30.41049326</td>
<td>0</td>
</tr>
<tr>
<td>2018/09/05 12:26:41</td>
<td>BTC ZJUM ETH BTC</td>
<td>2544.537563</td>
<td>2513.466247</td>
<td>31.07131777</td>
<td>0.663735045</td>
</tr>
<tr>
<td>2018/09/05 12:28:53</td>
<td>BTC HC ETH NANO BTC</td>
<td>2567.842267</td>
<td>2517.044943</td>
<td>30.7822996</td>
<td>0.367769479</td>
</tr>
<tr>
<td>2018/09/06 12:32:01</td>
<td>BTC ZEN ETH MDM BTC</td>
<td>2549.510824</td>
<td>2517.286848</td>
<td>32.22987567</td>
<td>1.865157341</td>
</tr>
<tr>
<td>2018/09/05 12:33:15</td>
<td>BTC MTL ETH USDTC BTC</td>
<td>2548.495893</td>
<td>2517.040256</td>
<td>32.16869949</td>
<td>1.57136231</td>
</tr>
<tr>
<td>2018/09/05 12:36:24</td>
<td>BTC ZEN ETH MTL BTC</td>
<td>2549.504876</td>
<td>2518.010368</td>
<td>31.49478495</td>
<td>1.08824351</td>
</tr>
<tr>
<td>2018/09/05 12:36:45</td>
<td>BTC MTL ETH PUX BTC</td>
<td>2557.043204</td>
<td>2527.477775</td>
<td>30.51492455</td>
<td>0.59992601</td>
</tr>
<tr>
<td>2018/09/05 12:39:10</td>
<td>BTC OMT ETH ZEN BTC</td>
<td>2558.481107</td>
<td>2528.560265</td>
<td>30.9204226</td>
<td>0.553876291</td>
</tr>
<tr>
<td>2018/09/05 12:43:38</td>
<td>BTC WAM ETH REP BTC</td>
<td>2567.891455</td>
<td>2526.190459</td>
<td>31.7009919</td>
<td>1.290531925</td>
</tr>
<tr>
<td>2018/09/05 12:44:54</td>
<td>BTC WAM ETH USDTC BTC</td>
<td>2555.745352</td>
<td>2525.664034</td>
<td>30.0818185</td>
<td>-0.36246117</td>
</tr>
</tbody>
</table>

Figure 3: Table showing Profitability with amountBuffer = 1.

As you can see from the table and graph below, an actual loss is shown with “amountBuffer” = 1. A higher frequency of arbitrages is noticed, but the profitability is most likely due to loss in precision due to the limitations imposed by Binance’s REST
API on both the price and amount. After executing nearly 50 arbitrage sequences with an average amount of 0.002 BTC, I noticed a loss of approximately 6USD equivalent, so I decided to move on to higher values of “amountBuffer”. The initial difference in portfolio value is due to previous experiments with arbitrage at higher values that had resulted in profit. I went back to an “amountBuffer” of 1 to log data in order to verify that it is indeed a losing proposition.

![Graph showing arbitrage results with amountBuffer = 1.](image)

**Figure 4**: Graph showing arbitrage results with amountBuffer = 1.

### 4.3 “amountBuffer” = 2

<table>
<thead>
<tr>
<th>Date and Time</th>
<th>Arbitrage Sequence</th>
<th>Portfolio Value (Arbitrage)</th>
<th>Portfolio Value (Hold)</th>
<th>Difference</th>
<th>Difference Adjusted to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018/09/05 14:34:57</td>
<td>BTC PPT ETH KMD BTC</td>
<td>2553.464868</td>
<td>2530.252937</td>
<td>23.211931</td>
<td>0.04052566</td>
</tr>
<tr>
<td>2018/09/05 14:42:32</td>
<td>BTC DASH ETH REP BTC</td>
<td>2549.412117</td>
<td>2524.973963</td>
<td>24.438154</td>
<td>0.04052566</td>
</tr>
<tr>
<td>2018/09/05 15:00:18</td>
<td>BTC NAV ETH XMR BTC</td>
<td>2551.928206</td>
<td>2526.491793</td>
<td>25.431026</td>
<td>2.03721144</td>
</tr>
<tr>
<td>2018/09/05 15:09:24</td>
<td>BTC PPT ETH BTC</td>
<td>2558.567465</td>
<td>2534.087602</td>
<td>24.479863</td>
<td>1.80531969</td>
</tr>
<tr>
<td>2018/09/05 15:18:04</td>
<td>BTC DASH ETH ZEN BTC</td>
<td>2556.599946</td>
<td>2533.492606</td>
<td>23.092340</td>
<td>-0.295107714</td>
</tr>
<tr>
<td>2018/09/05 15:20:31</td>
<td>BTC NAV ETH DGD BTC</td>
<td>2556.694474</td>
<td>2523.223934</td>
<td>24.371135</td>
<td>0.981382144</td>
</tr>
<tr>
<td>2018/09/05 15:27:33</td>
<td>BTC SALT ETH USDT NEO BTC</td>
<td>2556.777071</td>
<td>2532.456394</td>
<td>24.32068298</td>
<td>0.981382144</td>
</tr>
<tr>
<td>2018/09/05 15:32:50</td>
<td>BTC EDO ETH NEO BTC</td>
<td>2557.418173</td>
<td>2533.353082</td>
<td>24.050690</td>
<td>0.673391226</td>
</tr>
<tr>
<td>2018/09/05 15:33:14</td>
<td>BTC MCO ETH ETC BTC</td>
<td>2557.430404</td>
<td>2533.566379</td>
<td>23.8640245</td>
<td>0.470729152</td>
</tr>
</tbody>
</table>

![Table showing Profitability with amountBuffer = 2.](image)

**Figure 5**: Table showing Profitability with amountBuffer = 2.
The table above shows experiments with an “amountBuffer” = 2. Results were similar to “amountBuffer” = 1, but with an actual small gain of approximately 0.5USD after just 10 arbitrage executions. The reason for so little data on “amountBuffer” = 2 is because I found the program was much more profitable at higher values of “amountBuffer”. The graph below shows the table in a line bar format.

![Graph showing arbitrage results with amountBuffer = 2.](image)

Figure 6: Graph showing arbitrage results with amountBuffer = 2.

4.4 “amountBuffer” = 3

The table above shows experimental results with an “amountBuffer” value of 3, I started noticing some actual gains with respect to arbitrage execution. After 20 arbitrage executions, a net gain of approximately 7.5USD was noticed. This was exciting and I decided to bump up the value of “amountBuffer” even higher. The graph shown below shows the results from using an “amountBuffer” value of 3.
4.5 “amountBuffer” = 4

The tables below show my experimental results of 3 runs with an “amountBuffer” value of 4, and the tables and graphs are listed in sequence. I noticed that at this value, the profitability of my program was at its highest. In Run 1, after nearly 70 executions, the program had made a net profit of approximately 20USD equivalent.
### Figure 9: Table showing Profitability with amountBuffer = 4 (Run 1).

<table>
<thead>
<tr>
<th>Date and Time</th>
<th>Arbitrage Sequence</th>
<th>Portfolio Value (Arbitrage)</th>
<th>Portfolio Value (Unit)</th>
<th>Difference</th>
<th>Difference Adjusted to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-09-26 09:55</td>
<td>USD/JPY USD/JPY</td>
<td>2625.106388</td>
<td>2588.206924</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-26 23:35</td>
<td>BTC DASH ETH NAS BTC</td>
<td>2622.84327</td>
<td>2588.206924</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-27 09:27</td>
<td>ETH/USD ETH/USD</td>
<td>2587.68932</td>
<td>2595.18239</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-27 23:34</td>
<td>BTC ICX ETH ICX ETH</td>
<td>2592.70685</td>
<td>2592.70685</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-28 30:56</td>
<td>BTC ICX ETH ICX ETH</td>
<td>2589.52375</td>
<td>2592.70685</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-26 44:42</td>
<td>TRX BTC OAX ETH TRX</td>
<td>2622.83485</td>
<td>2592.70685</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-21 07:27</td>
<td>NANO BNB AION ETH NEO BNB</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-21 49:31</td>
<td>BTC KZC BNB BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-21 18:19</td>
<td>BTC MCO BTC MCO BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-21 59:23</td>
<td>BTC PPT ETH MCO BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-20 18:41</td>
<td>ETH VIA BTC BNB ETH</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-20 58:18</td>
<td>UNI ETH INV BTC ETH</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-19 05:18</td>
<td>BTC BNB UNISWAP ETH</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-18 35:20</td>
<td>BTC UNISWAP ETH</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-17 23:23</td>
<td>BTC BNB USDC BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-17 14:45</td>
<td>BTC BNB USDC BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-16 11:50</td>
<td>BTC BNB USDC BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-16 07:10</td>
<td>BTC BNB USDC BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-15 08:03</td>
<td>BTC BNB USDC BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-15 14:43</td>
<td>BTC BNB USDC BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-15 17:08</td>
<td>BTC BNB USDC BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2018-09-15 20:56</td>
<td>BTC BNB USDC BTC</td>
<td>2997.24800</td>
<td>2997.24800</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

23
This results of Run 1 are clearly shown in the graph below. Run 2’s table and graphs are shown further below, with a net gain of almost 6USD. Run 3 is displayed lastly; over 26 arbitrage executions, a net portfolio value increase of approximately 6USD was recorded.

Figure 10: Graph showing arbitrage results with amountBuffer = 4 (Run 1).

<table>
<thead>
<tr>
<th>Date and Time</th>
<th>Arbitrage Sequence</th>
<th>Portfolio Value (Arbitrage)</th>
<th>Portfolio Value (Held)</th>
<th>Difference</th>
<th>Difference Adjusted to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018/09/10 14:59:52</td>
<td>EOS BTC ZEN ETH EOS</td>
<td>2383.417756</td>
<td>2390.403799</td>
<td>66.986243</td>
<td>0</td>
</tr>
<tr>
<td>2018/09/10 15:44:48</td>
<td>ETH USDT ONT ETH</td>
<td>2374.910912</td>
<td>2292.453679</td>
<td>82.45723366</td>
<td>-0.5567235563</td>
</tr>
<tr>
<td>2018/09/10 15:45:36</td>
<td>BTC MTL ETH BCD BTC</td>
<td>2374.685034</td>
<td>2292.997969</td>
<td>82.587065</td>
<td>-0.4268922227</td>
</tr>
<tr>
<td>2018/09/10 15:46:51</td>
<td>BTC MTL ETH USDT BTC</td>
<td>2377.348023</td>
<td>2295.639393</td>
<td>81.7086433</td>
<td>-1.305872888</td>
</tr>
<tr>
<td>2018/09/10 15:49:51</td>
<td>BTC TRIG ETH LTC BTC</td>
<td>2378.306616</td>
<td>2296.29068</td>
<td>82.059308</td>
<td>-0.750212376</td>
</tr>
<tr>
<td>2018/09/10 15:50:36</td>
<td>TRIG ETH BCD BTC TRIG</td>
<td>2379.207488</td>
<td>2297.098895</td>
<td>82.1085984</td>
<td>-0.4254908844</td>
</tr>
<tr>
<td>2018/09/10 16:02:53</td>
<td>ONT BTC USDT ONT</td>
<td>2378.666992</td>
<td>2294.175786</td>
<td>84.49120323</td>
<td>1.47724801</td>
</tr>
<tr>
<td>2018/09/10 18:45:37</td>
<td>BNB WAVES ETH NAV BTC USDT BNB</td>
<td>2404.026442</td>
<td>2322.27622</td>
<td>81.7508198</td>
<td>-1.26137421</td>
</tr>
<tr>
<td>2018/09/10 18:59:35</td>
<td>BNB NANO BTC USDT BNB</td>
<td>2413.575337</td>
<td>2330.912059</td>
<td>82.66327903</td>
<td>-0.3006791927</td>
</tr>
<tr>
<td>2018/09/10 19:03:01</td>
<td>BTC WAVES ETH TRIG BTC</td>
<td>2416.07138</td>
<td>2331.262239</td>
<td>84.80914046</td>
<td>1.795183244</td>
</tr>
<tr>
<td>2018/09/10 19:10:35</td>
<td>BTC DASH ETH XMR BTC</td>
<td>2413.049129</td>
<td>2328.537036</td>
<td>84.5039288</td>
<td>1.489135655</td>
</tr>
<tr>
<td>2018/09/10 20:26:25</td>
<td>BCC ETH WAVES BNB BTC BCC</td>
<td>2409.195742</td>
<td>2320.332779</td>
<td>88.8634256</td>
<td>5.84905365</td>
</tr>
<tr>
<td>2018/09/10 21:03:50</td>
<td>BTC EOS BNB BTC</td>
<td>2404.362492</td>
<td>2316.551145</td>
<td>87.8313743</td>
<td>4.81736206</td>
</tr>
<tr>
<td>2018/09/10 21:14:45</td>
<td>BTC WAVES ETH BTC</td>
<td>2402.203444</td>
<td>2313.456619</td>
<td>88.7462436</td>
<td>5.738671616</td>
</tr>
</tbody>
</table>

Figure 11: Table showing Profitability with amountBuffer = 4 (Run 2).
Figure 12: Graph showing arbitrage results with amountBuffer = 4 (Run 2).

<table>
<thead>
<tr>
<th>Date and Time</th>
<th>Arbitrage Sequence</th>
<th>Portfolio Value (Arbitrage)</th>
<th>Portfolio Value (Held)</th>
<th>Difference</th>
<th>Difference Adjusted to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018/09/11 15:11:03</td>
<td>BTC ICX USDT ETH WTC BTC</td>
<td>2288.64/1235</td>
<td>2197.04/1628</td>
<td>91.59/6070</td>
<td>-1.57/588714</td>
</tr>
<tr>
<td>2018/09/11 15:20:41</td>
<td>BTC BCD ETH USDT OLM BTC</td>
<td>2291.63/6378</td>
<td>2201.04/1687</td>
<td>90.02/90036</td>
<td>-2.97/808388</td>
</tr>
<tr>
<td>2018/09/11 15:28:24</td>
<td>ETH WTC BTC NAS ETH</td>
<td>2296.17/6096</td>
<td>2204.24/053</td>
<td>93.93/56637</td>
<td>2.35/59295</td>
</tr>
<tr>
<td>2018/09/11 15:31:19</td>
<td>SALT BTC ETH SALT</td>
<td>2300.89/9521</td>
<td>2205.94/683</td>
<td>95.19/26896</td>
<td>3.58/961897</td>
</tr>
<tr>
<td>2018/09/11 15:32:07</td>
<td>GVT ETH BTC GVT</td>
<td>2301.54/339</td>
<td>2206.03/177</td>
<td>94.14/6227</td>
<td>3.01/455916</td>
</tr>
<tr>
<td>2018/09/11 15:33:04</td>
<td>ETH GVT BTC USDT ETH</td>
<td>2302.87/1756</td>
<td>2207.46/253</td>
<td>95.40/55226</td>
<td>3.80/541855</td>
</tr>
<tr>
<td>2018/09/11 15:35:31</td>
<td>LTC BTC GVT ETH LTC</td>
<td>2302.50/7069</td>
<td>2207.32/207</td>
<td>95.18/5348</td>
<td>3.58/928412</td>
</tr>
<tr>
<td>2018/09/11 15:35:41</td>
<td>ETH NANO BTC GVT ETH</td>
<td>2302.41/0621</td>
<td>2207.76/001</td>
<td>94.64/2605</td>
<td>3.04/391298</td>
</tr>
<tr>
<td>2018/09/11 15:35:42</td>
<td>GVT ETH USDT BTC GVT</td>
<td>2302.25/05</td>
<td>2207.98/567</td>
<td>94.55/4788</td>
<td>2.95/87324</td>
</tr>
<tr>
<td>2018/09/11 15:36:13</td>
<td>ETH USDT BTC GVT ETH</td>
<td>2301.95/3274</td>
<td>2207.58/069</td>
<td>94.37/23316</td>
<td>2.77/27648</td>
</tr>
<tr>
<td>2018/09/11 15:38:09</td>
<td>MCO ETH BTC MCO</td>
<td>2301.96/5335</td>
<td>2207.75/337</td>
<td>94.21/09813</td>
<td>2.61/37464</td>
</tr>
<tr>
<td>2018/09/11 15:42:06</td>
<td>NAV BTC ZEN ETH NAV</td>
<td>2305.07/5096</td>
<td>2212.02/628</td>
<td>93.05/6062</td>
<td>1.45/201164</td>
</tr>
<tr>
<td>2018/09/11 15:44:18</td>
<td>BTC HC ETH SKY BTC</td>
<td>2304.52/668</td>
<td>2210.98/653</td>
<td>93.53/7156</td>
<td>1.93/70653</td>
</tr>
<tr>
<td>2018/09/11 15:45:40</td>
<td>GVT ETH BTC GVT ETH</td>
<td>2305.62/185</td>
<td>2211.24/176</td>
<td>94.40/6091</td>
<td>2.80/420101</td>
</tr>
<tr>
<td>2018/09/11 15:46:09</td>
<td>BTC GVT ETH NEO BTC</td>
<td>2305.34/241</td>
<td>2210.86/404</td>
<td>94.48/8056</td>
<td>2.82/198586</td>
</tr>
<tr>
<td>2018/09/11 15:50:47</td>
<td>ETH BTC USDT BTC GVT ETH</td>
<td>2306.48/728</td>
<td>2205.43/562</td>
<td>95.03/76613</td>
<td>3.43/75506</td>
</tr>
<tr>
<td>2018/09/11 15:59:08</td>
<td>ARK BTC WTC ETH ARK</td>
<td>2297.78/967</td>
<td>2200.99/029</td>
<td>97.79/16703</td>
<td>5.19/262963</td>
</tr>
<tr>
<td>2018/09/11 16:11:01</td>
<td>BTC BCD BTC GVT ETH</td>
<td>2303.48/912</td>
<td>2205.83/4107</td>
<td>97.60/05029</td>
<td>6.05/119217</td>
</tr>
<tr>
<td>2018/09/11 16:12:32</td>
<td>WTC ETH MTL BTC WTC</td>
<td>2303.51/993</td>
<td>2206.76/951</td>
<td>97.23/93756</td>
<td>5.63/710491</td>
</tr>
<tr>
<td>2018/09/11 16:18:44</td>
<td>ETH HC BTC SKY ETH</td>
<td>2299.97/011</td>
<td>2202.49/511</td>
<td>97.47/60014</td>
<td>5.88/784073</td>
</tr>
<tr>
<td>2018/09/11 16:40:14</td>
<td>ETH KMD BTC NAS ETH</td>
<td>2282.96/216</td>
<td>2198.40/239</td>
<td>94.59/02099</td>
<td>2.96/031392</td>
</tr>
<tr>
<td>2018/09/11 16:51:00</td>
<td>ETH SKY BTC NEBL ETH</td>
<td>2295.37/717</td>
<td>2197.61/953</td>
<td>97.75/781491</td>
<td>6.15/8207842</td>
</tr>
</tbody>
</table>

Figure 13: Table showing Profitability with amountBuffer = 4 (Run 3).
Figure 14: Graph showing arbitrage results with amountBuffer = 4 (Run 3).

With higher values of “amountBuffer”, such as ones greater than 6, very few arbitrages are executed. This is due to the fact that Binance is the most highly traded exchange in the world and as such is highly fluid. As such, and due to extraneous circumstances, data for “amountBuffer” = 5 and 6 were unable to be collected and displayed. I did conclude that an “amountBuffer” value of 3 to 4 was ideal, however. With lower values of “amountBuffer”, more but less profitable arbitrages are executed. With higher values of “amountBuffer”, the profitability of each arbitrage executed increases, but an arbitrageur needs to take into account time. Values of 5 and 6 were not as profitable as 3 or 4, and anything higher than 6 would result in nearly nil arbitrage executions. From my experimental results, I concluded that “amountBuffer” is a critical and necessary parameter. Also, the total trade amount value you choose to execute with in the beginning is another factor that you need to consider. With higher trade values, you will likely end up with more open unfilled orders that will
take time to complete. Thus, I suggest anyone trying to build an intra-exchange arbitrage bot to use as minimal a trade value as possible.
CHAPTER 5
Discussion and Observations

On an exchange like Binance, or any highly volatile market, one needs to take into consideration many factors. The first obviously is how you can measure the success of your program. I solved this issue by taking an account snapshot of a past account balance, and comparing it with the estimated portfolio value of my actual account balance after executing a series of arbitrages.

Another thing to take into consideration is that precision matters when it comes down to arbitrage. If we could have just 7-9 significant digits for both amount and pricing, I’m positive “amountBuffer” wouldn’t be needed, and if it was needed, a value of 1 would work just fine.

5.1 Important Topics

• Don’t use market orders. When you use a market order, it fills at the best possible price available. Because we are executing arbitrage based on best available exchange pricing, executing a market order leaves you susceptible to filling your market order by filling multiple orders in the book with higher prices in a buy-side order, and lower prices in a sell-side order, in effect nullifying the validity of the arbitrage.

• There is a specific precision (number of significant digits) required for the amounts on an order for each trading pair. Many symbols have a precision requirement of 0, which means that the amount must be an integer. These symbols were filtered out when finding arbitrage opportunities to execute because the integer precision would likely result in inaccurate executions of arbitrage.

• There is also a specific precision required for the price, but because we are executing limit orders, there is no filtering or buffering required for the price of an order.
• In this project’s implementation, an initial investment of cryptocurrency equivalent to $3000 USD was made, with an additional injection of $2000 USD later on. This amount was divided by the number of coins on Binance, and a specific amount of each coin was bought. The reason for diversification is because we are executing limit orders, we want the arbitrage to go all the way through at the time of execution. If we were executing market orders, this would not be necessary. (But don’t use market orders.)

• It is also interesting to note that all limit orders will usually fill at some point in the future. This was found to be the case in this project. All limit orders filled within a timespan of 2 months. Due to this inherent property of working with crypto-exchanges, or any stock market in particular, an unfilled limit order will have the side effect of giving you an inaccurate snapshot of your arbitrage. Ideally, you would want all the trades to complete after executing the order. Hence, that is why you saw spikes and jumps in my graphs. These are most likely due to past limit orders filling in the future and increasing my portfolio value through the delayed arbitrage profit.

• Due to inherent volatility of cryptocurrency market, you should take an account snapshot of your starting coin balances, and store it into a hashmap with each coin as the key and balance as the value. You can then use this hashmap to calculate the account value if you had not executed arbitrage. This same method was used to calculate present value of coin balances. Because Binance’s API returns coins that are not listed on the exchange with the “getBalances” method provided by the wallet class in the XChange library, I only polled the server for the coin if it is contained in the list of vertices in my Main class.
CHAPTER 6

Conclusion and Future Work

In this project, an intra-exchange arbitrage bot for the Binance cryptocurrency exchange was created and successfully deployed. During the implementation of the project, two main challenges were encountered, one due to the precision offered by the REST API offered, and the other due to the volatility of the cryptocurrency market. To try to combat the limited precision imposed on amount and price when executing trades, a parameter called “amountBuffer” was introduced to give the arbitrage more breathing room. To combat market volatility, account snapshots were taken of previous account balances and used to compare to current portfolio value; both these values were calculated at the current market exchange rates during the time of execution. Overall, it can be said that it is possible to create an intra-exchange arbitrage bot for the cryptocurrency market, though I believe a significantly larger investment of capital would be required for a sit back and watch your cryptocurrency portfolio grow type of situation.

It is interesting to note that for future work, websockets provided by an exchange could be an area that could be explored further. My exploration of websocket usage use in this project was very limited.

Another exciting direction this project could go is undergoing modification for an exchange like Hitbtc, which actually pays out a commission of 0.01% when you place an order that adds liquidity to the market. A market maker is an entity, such as a person or a bot, that does not buy and sell at the best price offered. In other words, the entity will sell at a price higher than the current best market "ask" price, and buy at a price lower then the best "bid" price. Because we have this data when we poll any exchange’s API, we can easily adjust our program to perhaps adjust our price to 99.5-99.9% or 100.1-100.5% of the best market price available, depending on if
you were buying or selling, respectively, and get paid commission for doing so. From my experience with Binance, all my limit orders eventually filled within 1-2 months. Hitbtc is a very good choice for executing arbitrage because you then can directly execute the arbitrage and rake in arbitrage profit along with the trading commission received. However, due to the large number of coins on Hitbtc, about 450 at the time of this writing, the capital required to perform a project like this would require a significant amount of capital. I look forward to implementing this for the Hitbtc exchange in the near future.
LIST OF REFERENCES


