Project Proposal: Cryptocurrency Algorithmic Trading (Arbitrage)

Team Code: CRY0

Date: 1-21-2017

Team Members:

- Timothy Bramlett, email: bramlett@oregonstate.edu; Github: TimothyBramlett
- Patrick Mullaney, email: mullanep@oregonstate.edu; Github: mullaneyp
- Matt Nutsch; email: nutschm@oregonstate.edu; Github: mnutsch

Introduction:

We will create a cryptocurrency wallet app with the ability to perform algorithmic trading. Specifically, our app will look for arbitrage opportunities.

The app will repeatedly query the various exchanges rates between cryptocurrencies. If it finds a situation where the exchange rates are not consistent, then it automatic executes a trade. In other words, create a bot which finds arbitrage opportunities between different crypto-currencies.

For example, the app will check the USD exchange rate for BCH at exchange A and the exchange rate for BCH at exchange B. If the associated fees are less than the difference in price, then the app will automatically execute a trade.

General Description:

1. Use the Oregon State "Flip" web server.
2. Create a process which finds arbitrage opportunities.
    a. The process should read the exchange rates for a currency from multiple cryptocurrency exchanges. This will initially include the currencies Bitcoin Cash (BCH), Litecoin (LTC), Ethereum (ETH), and Bitcoin Core (BTC). The cryptocurrency **exchanges used will initially be GDAX and Gemini**.
    b. The process should then identify possibilities where a currency can be bought at one exchange, transferred to a wallet at another exchange, and sold for a profit.
    c. The process should include considerations for a minimum profit amount when determining if a trade is profitable.
    d. The process should send an email notification to the user with all opportunities above a set profitability threshold. The notification should include the currency, each exchange rate pricing, fee pricing, and the estimated profit.

    e. The process should also sent a text (SMS) notification to the user with similar content to the email. The service to use for **text (SMS) notifications will be Tropo**.
3. Create a polling process which runs the processes defined in steps 2 and 3 repeatedly throughout the day on a recurring basis at fixed intervals.
4. Create code which will can programmatically initiate transactions in an identified transaction chain if the profitability is above a set threshold.
    a. This process should send a notification to the user(s) about the transfers.

Language and Style Guides:
We will use Python for this application. The application code will follow the PEP 8 Style Guide, which can be found here: https://www.python.org/dev/peps/pep-0008/.

Server:
The application will run on the OSU Flip server. The URL to connect to that server is "flip.engr.oregonstate.edu", port 22. Team members will use their ONID's to connect.

Interface:
The application does not need a graphical user interface. Instead it be run from the command line on a Linux server. The application will however generate emails to communicate trading opportunities that it finds. The application will be scheduled to run automatically at recurring intervals. This scheduling may be performed via a Cron script on the server.

File Structure:

main.py = This will be the main file which is run by the Cron job and/or user.

configuration.py = This file will contain configuration information. It will define API keys for different cryptocurrency exchanges. It may define minimum or maximum thresholds when checking for arbitrage opportunities. It may define limits on the amounts of transactions for perform.

readExchangeRates[Exchange].py = This will be a series of files. Each file will contain functions that read data from a cryptocurrency exchange and then parse the data into a common object format. Replace [Exchange] with the name of the exchange. For example, there may be files called readExchangeRatesGDAX.py and readExchangeRatesGemini.py. Functions from these files will be imported into main.py.

execTransactions[Exchange].py = This will be a series of files. Each file will contain functions that send transaction requests to a cryptocurrency exchange. Replace [Exchange] with the name of the exchange. For example, there may be files called execTransactionsGDAX.py and execTransactionsGemini.py. Functions from these files will be imported into main.py. Transactions which may be performed include: buying a given currency, selling a given

currency, and transferring a currency to an external wallet address. Executing a transaction may be complicated and will include the following considerations:

sendEmail.py = This will be a file which contains functions for generating an e-mail to the user(s). Functions from this file will be imported into main.py.

sendSMS.py = This will be a file which contains functions for generating a text (SMS) message to the user(s). Functions from this file will be imported into main.py.

checkForSingleCurrencyOpps.py = This file will contain function(s) which check for arbitrage opportunities from trading a single currency between different exchanges.

checkForMultipleCurrencyOpps.py = This file will contain function(s) which check for arbitrage opportunities from exchanging a chain of currencies between each other.

Tasks:

PHASE I

A. Identify the currencies and exchanges to use.
    a. Exchanges to use:
        i. GDAX
        ii. Gemini
    b. Currencies to use:
        i. Bitcoin cash (BCH)
        ii. Ethereum (ETH)
        iii. Litecoin (LTC)
        iv. Bitcoin core (BTC)
B. Register accounts at cryptocurrency exchanges.
    a. GDAX: https://www.gdax.com/
    b. Gemini: https://gemini.com/
C. Create functionality for reading and parsing data from cryptocurrency exchange API's.
    a. Create the file readExchangeRatesGDAX.py.
    b. Create the file readExchangeRatesGemini.py.
D. Create functionality for executing transactions on cryptocurrency exchanges.
    a. Create the file execTransactionsGDAX.py.
    b. Create the file execTransactionsGemini.py.
E. Create functionality for sending emails.
    a. Set up an account with an email service to send the emails.
    b. Add the email settings to the configuration.py file.
    c. Write the file sendEmail.py with a function for generating emails.

PHASE II

F. Create functionality for sending text (SMS) messages.
    a. Set up an account with the API service Tropo.
    b. Add the text (SMS) settings to the configuration.py file.

    c. Write the file sendSMS.py with a function for generating text (SMS) messages.
G. Create functionality to check for arbitrage opportunities.
    a. Identify transaction fees for each exchange. (see G - a below)
        i. Add this as a setting in configuration.py.
    b. Identify expected transaction settlement time. (see G - b below)
        i. Add a risk factor related to settlement time as a setting in configuration.py.
    c. GDAX <-> Gemini
        i. Bitcoin Cash (BCH)
        ii. Litecoin (LTC)
        iii. Ethereum (ETH)
        iv. Bitcoin Core (BTC)
H. Write the main body of code, which coordinates the various pieces.
I. Thoroughly test each piece of functionality.
    a. Test **reading and parsing data** from exchange API's.
    b. Test **executing transactions** through exchange API's.
    c. Test **calculating single currency** opportunities.
    d. Test **calculating multiple currency** opportunities.

PHASE III

J. Create cron instructions to run the application on a recurring basis.
K. Write Instructions for setting up and using the application.
    a. How to set up user accounts at the exchanges.
    b. How to install the script on a server.
    c. How to configure the script (configuration.py).
    d. How to schedule the script to run automatically.

Collaboration:

Version Control / Code Repository

A private repository on Github.com will be used to store code during development. The repository can be found by logged in team members at this URL: https://github.com/mnutsch/Crypto_Algorithmic_Trading

Communication

The team members will primarily communicate over Slack, using the private channel "crypto_project_461_w" in the osu-cs.slack.com group. When needed the team will meet over video conference using Google Hangout.

Responsibilities:

Each team member is expected to contribute 100 hours to the project.

Matt -

Task C: Reading and parsing data from exchanges.
Task F: Text (SMS) generation for notifications.

Patrick -

Task E: Email generation for notifications.
Task G: Functionality to check for arbitrage opportunities.

Tim -

Task D: Executing transactions on exchanges.
Task H: Main body of code to coordinate the various pieces.

(all team members) -

Task A: Identify currencies and exchanges to use.
Task B: Register accounts at the cryptocurrency exchanges.
Task I: Testing of functionality.
Write progress reports.
Write instructions for setting up and using the application.

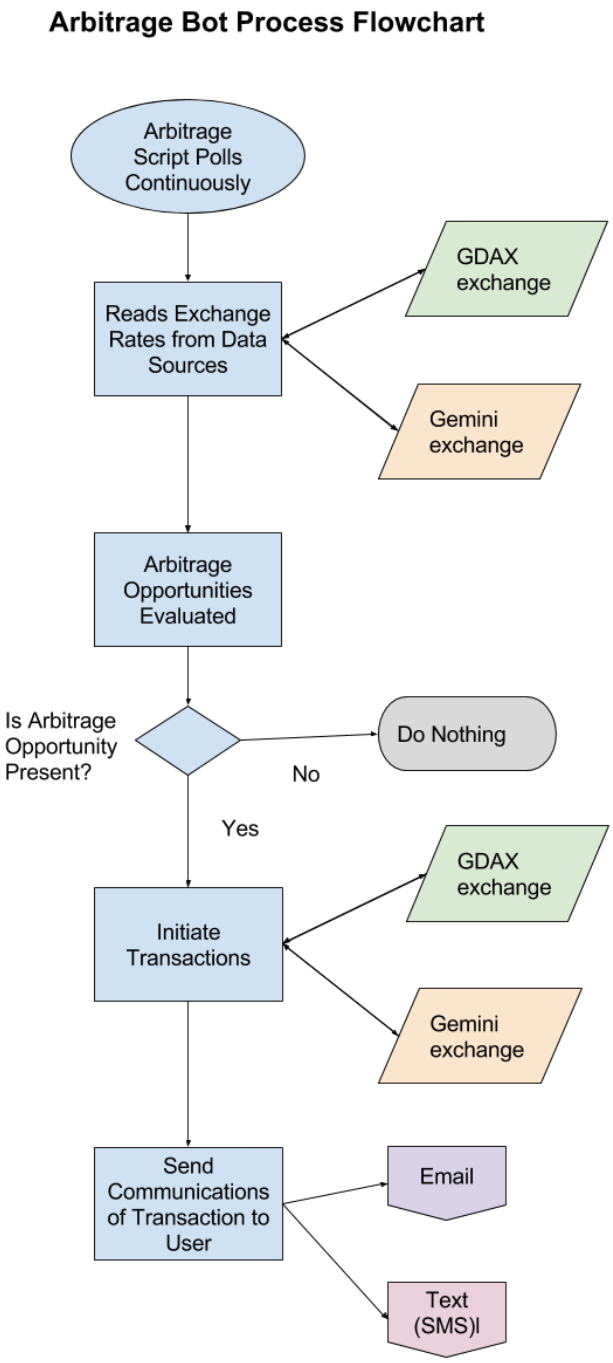
Useful Links:

- Information at this URL for useful arbitrage calculations: https://99bitcoins.com/bitcoin-arbitrage/
- GDAX cryptocurrency exchange API info: https://docs.gdax.com/?python#introduction
- Gemini cryptocurrency exchange API info: https://docs.gemini.com/rest-api/?python#introduction
- Tropo API info (for sending text SMS messages): https://www.tropo.com/docs/rest

Conclusion:

This algorithmic trading project will teach us a lot about conducting financial transactions programmatically. It will help us build a model that could later be reused or extended to things like stock transactions, fiat currency trading, or commodity trading.

Appendix A:


Arbitrage Bot Process Flowchart
Arbitrage Script Polls Continuously
Reads Exchange Rates from Data Sources
GDAX exchange
Gemini exchange
Arbitrage Opportunities Evaluated
Is Arbitrage Opportunity Present?
Do Nothing
No
Yes
Initiate Transactions
GDAX exchange
Gemini exchange
Send Communications of Transaction to User
Email
Text (SMS)l

Appendix B:

Exchange and Currency Diagram