Mid Point Project Check:
Cryptocurrency Algorithmic Trading (Arbitrage)

Team Code: CRY0

Date: 2-19-2017

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

VIDEO UPDATE OF PROJECT:
https://media.oregonstate.edu/media/t/0_gpfirjd4

GITHUB URL:
https://github.com/mnutsch/Crypto_Algorithmic_Trading

Description of Current Project Status:
We created an app which looks for arbitrage opportunities in cryptocurrencies.

The application does not need a graphical user interface but rather runs from the command line on a Linux server. The app repeatedly queries at the various exchanges rates between cryptocurrencies at recurring intervals. If it finds a situation where the exchange rates are not consistent, then it notifies us of the opportunity via SMS text message (sending notifications via email functionality but not yet incorporated into main).

For example, the app check the USD exchange rate for Bitcoin Core (BTC) at the exchange "GDAX" and also the exchange rate for Bitcoin Core (BTC) at exchange Gemini. If the difference in price exceeds a preset threshold, then the app sends us a message.

Accomplishments:

server setup = We set up a Linux server on AWS EC2 to host the project application. This was necessary due to limitations in scheduling and keep alive permissions on the Oregon State University "Flip" server.

main.py = This is the main file which is run by the Cron job and/or user. It remains a work in progress as to further incorporation of files that remain in development however is currently running on the server.

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

readExchangeRates[Exchange].py = This is a series of files. Each file contains 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. The application reads rates from two exchanges, GDAX and Gemini. Functions from these files have been imported into main.py, checkForSinglecurrencyOpps.py, and checkForMultipleCurrencyOpps.py.

checkForSingleCurrencyOpps.py = This file contain function(s) which check for arbitrage opportunities from trading a single currency between different exchanges. The file remains a work in progress from an optimization standpoint to generate better results.

checkForMultipleCurrencyOpps.py = This file contains function(s) which check for arbitrage opportunities from exchanging a chain of currencies between each other. It remains in development at this time. Currently, it evaluates a potential chain of currency transactions by first calculating the cost of converting $[amount] currency A to $[amount] of currency B, $[amount] of currency B to $[amount] of currency C, and $[amount] of currency C to $[amount] of currency A. A function will then iterate through all permutations (currently up to 1,320 permutations) of these cost combinations and return a list of potentially profitable chains of transactions.

calc.py = This file contains functions for calculating profit/loss, fees, etc. Currently, a lot of functionality is similar to checkForSingleCurrencyOpps.py, however this will likely be refactored as certain functions can be used by both that file and checkForMultipleCurrencyOpps.py as well as may have potential use in other file.

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

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

tropoSendSMS.py = This script runs on the Tropo (text SMS) service. It handles incoming calls from the sendSMS.py file and translates those connections into SMS messages.

logCryptoPrices.py = This script logs crypto-currency exchange rates in a CSV file. The log it creates was intended to be used to help us determine what is a safe margin of error for executing an arbitrage transaction.

Tasks Completed From Original Plan:

PHASE I Completed Tasks

A. Currencies and exchanges to use identified.
    a. Exchanges used: GDAX, Gemini
    b. Currencies used: Bitcoin cash (BCH), Ethereum (ETH), Litecoin (LTC), Bitcoin core (BTC)
B. Functionality for reading and parsing data from cryptocurrency exchange API's.
C. Functionality for sending emails created.

PHASE II Completed Tasks

D. Functionality for sending text (SMS) messages created: Messages sent through Tropo.
E. Functionality to check for arbitrage opportunities created for single currency opportunities and remains in progress for multiple currency opportunities.

Changes From Original Plan:

- Create functionality for executing transactions on cryptocurrency exchanges to be completed after some Phase II pieces.
- Server changed from OSU to AWS as it did not appear that we would be able to run Cron scripts to schedule tasks on the OSU server.
- We found that Bitcoin Cash (BCH) and Litecoin (LTC) were not available for "single currency" opportunities. This is because the Gemini exchange does not support transacting in these 2 currencies through an API.

Remaining Tasks:

The remaining tasks are targeted at automatically executing trades if a situation where the exchange rates are not consistent and trade execution would return a profit after associated fees are included as well as algorithmic improvements to find more arbitrage opportunities. 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.

checkForMultipleArbitrageOpps.py = This file remains in development. Currently it returns a list of potentially profitable chains of transaction costs up to a chain of three transactions of currency (buy $[amount] of currency B with $[amount] of currency A, buy $[amount] of currency C with $[amount] of currency B, and buy $[amount] of currency A with $[amount] of currency C). Upon further development, it will return the overall profit/loss or costs of such a transaction to complete the information about the potential arbitrage opportunity. Possible improvements include expanding the potential chain of interactions beyond three transactions, as well as possible optimization of a table of conversion costs via dynamic programming (if compatible with API and time permitting).

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.

withdrawToCryptoAddressFrom[Exchange].py = This will be a series of files which transfer cryptocurrency from a wallet at one exchange to a wallet at another exchange. Although not in the original project plan, we determined that this is needed to fully automate a full arbitrage trade.

PHASE I Remaining:

F. Register accounts at cryptocurrency exchanges.
    a. GDAX: https://www.gdax.com/
    b. Gemini: https://gemini.com/
G. Create functionality for executing transactions on cryptocurrency exchanges.
    a. Create the file execTransactionsGDAX.py.
    b. Create the file execTransactionsGemini.py.

PHASE II Remaining:

H. Finish functionality to check for arbitrage opportunities.
    a. Add transaction fees for each exchanges as a setting in configuration.py.
    b. Identify expected transaction settlement time.
        i. Add a risk factor related to settlement time as a setting in configuration.py.
I. Complete main body of code, which coordinates the various pieces.
J. 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 Remaining

K. Create cron instructions to run the application on a recurring basis.
L. 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.

Server:

Due to limitations we found with permissions on the Oregon State University "Flip" server, we set up a dedicated project server running Linux using Amazon Web Services.

Collaboration:

Version Control / Code Repository
A repository on Github.com has been used to store code during development. The repository has been made public for grading at this URL:
https://github.com/mnutsch/Crypto_Algorithmic_Trading

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

Responsibilities:

Matt -

Setup the Linux server for hosting the project application.
Wrote the readExchangeRates[Exchange].py files.
Wrote the sendSMS.py files.
Setup the Tropo account (for text SMS) and wrote the tropoSendSMS.py file.
Wrote the logCryptoPrices.py file.

Patrick -

Wrote the checkSingleCurrencyOpps.py file.
Wrote the sendEmail.py file.
Wrote the checkMultipleCurrencyOpps.py file (development ongoing at this stage).
Task H: Finish functionality to check for arbitrage opportunities.

Tim -

Wrote the main.py file.
Task G: Executing transactions on exchanges.
Task I: Complete main body of code to coordinate the various pieces.

(all team members) -

Task F: Register accounts at the cryptocurrency exchanges.
Task J: Testing of functionality.
Task L: Write instructions for setting up and using the application.