Arbitrage Trading Systems for Cryptocurrencies. Design Principles and Server Architecture

Cristian PĂUNA
The Bucharest University of Economic Studies, Romania
cristian.pauna@ie.ase.ro

When dot.com has become a quaint idea, when electronic shops have lost the mass attention, while classical and margin trading has become obsolete, something new is coming: cryptocurrencies. Hundreds of virtual coins have been invented for a single reason: the profit. The high price volatility of these new markets and the fact that the virtual coins price is not regulated by a central bank or a single exchange, gives us opportunities for arbitrage trading. The existence of important price differences makes possible the profit when an automated system buy cheaper and sell more expensive in the same time. This paper will present the general principles underpinning the implementation of arbitrage trading software for virtual coins market. The very large number of cryptocurrencies and exchanges fundamentally change the server architecture of the trading software. The distributed price data in hundreds of sources and the technical differences of each of these data providers make all the things difficult to be implemented in a single application. The low-latency order calculation needed for the fast delivery before a significant price change, in the presence of thousands of price quotes coming from hundreds of distributed servers makes everything special.

Keywords: *Cryptocurrency (CRYC), Arbitrage trading (ART), Algorithmic trading (AT), High Frequency Trading (HFT), Automated trading software (ATS)*

1 Introduction

Bitcoin (BTC), Litecoin (LTH), Ethereum (ETH), Ripple (RIP), thousands of virtual coins already exists. The Bitcoin "introduced in 2009", "caught the interest of the mainstream media in 2012" [1]. After that, an undefined number of virtual coins were invented. All these coins are named virtual just because they exist only in the electronic representation. They are also called cryptocurrencies because they are designed to work as a currency like a medium of value exchange and the "cryptography is used to secure financial transactions" [2], to control the creation of new units, to verify the transfer of funds and to set the owner for each unit.

The reason involving real money to buy this kind of virtual assets is of course the profit. The virtual coins value is fluctuating on the free markets as any asset, powered by speculative and marketing actions and sustained at least hypothetical by some economical processes involved in the background of each crypto coin. However, even it is a free market, the virtual coin market has its own characteristics, especially when it is about the volatility and "the flexibility in the supply schedule" [3].

We are not so far away the days when Bitcoin almost touched 20.000 USD in a relative short period of time starting from couple of bucks and after that plunged back to 7.000 - 8.000 USD in an even shorter interval. This kind of movements makes the investment in cryptocurrencies an attractive one today. However, the high price volatility makes this investment to be a very risky one, drawdowns of more than 61.8% being very easily assimilated with a gambling game. For all that, keeping the money into a virtual currency hoping the value will increase after a while can be a profitable idea, but it is not the only one. All these assets are operating independently of a central bank or a central exchange.

Without a central price regulator the quotes can be different from an exchange to another. That means we can find cases when and where we can buy cheap and sell more expensive the same thing in the same moment of time. This will be called an arbitrage trade. The unbalanced price of the same asset in two different places will give us an instant profit.

DOI: 10.12948/issn14531305/22.2.2018.04

The question is how to find these opportunities and how can we profit from all of these in the real time. The theory is easy, just find two exchanges with different quotes for the same equity, buy where the equity is cheaper and sell where it is more expensive.
The problems start when we will consider the facts that there are hundreds of crypto pairs, hundreds of thousands of combination of these coins and hundreds of exchanges where you can buy and sell all of these assets. That means a human being cannot compare all these prices in order to find the best trade opportunities, to build the orders and to send them to those exchanges before the price is changed. Only a computer can to manage all of these actions.
This paper will present how an arbitrage trading system for cryptocurrencies can be built, will reveal the main principles and practices and will also present some results in order to open or to grow the interest for this kind of software systems.

2 Arbitrage trading

As we already presented, the arbitrage trading means to buy cheap and to sell more expensive the same equity in the same moment of time in two places where the quote prices are different. The arbitrage trading is not a new idea. It was used since years to trade shares. "Stocks are matched into pairs with minimum distance between normalized historical prices" [4] and arbitrage trading strategies can be used with any stock exchange in order to make profit.
Usual, the classical arbitrage trade is involving the same volume of equity in the both trades. For each trade (i), the profit (P_i) is equal with the difference between the sell price ($sell_i$) and the buy price (buy_i) multiplied by the volume traded (V_i).

$$P_i = V_i (sell_i - buy_i) \qquad (1)$$

For this simple trade idea, the trading software must compare continuously the quotes from different exchanges and find the highest price differences. But the classical arbitrage trading is not the only one trading opportunity which can be found in the cryptocurrency markets. The virtual coins are traded in pairs. The price of a pair, for example BTCETH if we will consider Bitcoin and Ethereum, means how much ETH we need to sell in order to buy a specified unit of BTC. When an exchange ask you a specified price to buy BTCETH and another exchange offers you a higher price for the same pair, a classical arbitrage trade can be executed and the profit will result from the price difference. But sometimes the dependency between two coins is not expressed in the same way in all exchanges. There are cases when an exchange will quote the price for BTCETH and another exchange will quote the price for ETHBTC. The virtual coins involved are the same but the price in built in the reversed way. For this case, to find an arbitrage trade the software must compare the price from the first exchange with the reversed price into the second. Now we are talking about a reversed arbitrage trade. In this case the profit will be:

$$P_i = V_i \left(sell_i - \frac{1}{buy_i} \right) \qquad (2)$$

The execution of the trades is usual simple. A trading order means to buy or sell the specified pair at a specified price with an established volume. Of course an order can be executed by the exchange only if there are enough coin units to sell in that exchange in order to buy the ordered volume from the first coin. This validation must to be done by the trading software before to send the orders to the exchanges, based on the real-time capital data provided by each exchange. An arbitrage trade means at least two orders in two different exchanges.
Another type of arbitrage trades is involving more pairs. Sometimes the product of two, three or more pair quotes is imbalanced between two exchanges or even in the same exchange. For understanding, let's consider the next inequalities:

$$BTCETH*ETHLTC<BTCLTC \quad (3)$$

and

$$BTCETH*ETHLTC*LTCRIP<BTCRIP \quad (4)$$

DOI: 10.12948/issn14531305/22.2.2018.04

In cases like these, multiple arbitrage trades can be possible in order to make profit. In the first exchange will be executed more trades and in the second only one. The software will buy the lowest price for multiple pairs and sell the last one on a higher price. The profit for multiple arbitrage trades will be expressed in this case by the formula:

$$P_i = V_i sell_i - \left(\sum_{k=1}^{nBUY} V_k buy_k \right)_i \qquad (5)$$

For the next level of multi arbitrage trades, when we buy more pairs and sell more pairs in order to earn the difference between those operations, the profit will be expressed by the next equation:

$$P_i = \left(\sum_{j=1}^{nSELL} V_j sell_j \right)_i - \left(\sum_{k=1}^{nBUY} V_k buy_k \right)_i \qquad (6)$$

where (nBUY) is the number of buy trades and (nSELL) is the number of sell trades made simultaneous. This kind of trades can be found using two different exchanges but sometimes the difference can be found even in the same exchange. The multiple arbitrage trades are more complex to be validated and executed and a serious computational power is needed in order to deliver all the trading signals and orders in real time. Only low-latency trading software can manage this process in order to deliver the orders in that short time needed to execute them.

3. Cryptocurrency dictionaries

One of the most important parts of an arbitrage trading system for cryptocurrencies is the dictionary. Due to a very high number of cryptocurrencies quotes involved and because there is not a standard for naming the coins and pairs, a lot of false trading signals can be built without a strict codification methodology. Each exchange has its own coin and pair codes. We can find cases when the same coin can be named or coded differently.

For example BTC can be the symbol for Bitcoin in the most exchanges but some of them will code the Bitcoin with XBC or BTX. In order to compare the price of the same currencies, a codification dictionary must to be implemented. The trading software must have its own codification rules and all codes from all exchanges must to be included into the trading software database.

In addition there are cases when the code of a pair into an exchange can be the code of another pair into another exchange. For example SCAPPC is representing the pair of SC reported to the APPC coin in the most exchanges but it others it is the pair between SCA and PPC. All these codes SC, SCC, PPC and APPC are valid crypto coin codes. This is the case when a profitable arbitrage trade can be found if we will compare the price of the SCAPPC pair into two exchanges where that code represent different coin pairs. To avoid these type of false signals, in the trading software dictionary must to be inserted each pair code from each exchange related with the real pairs that are quoted. A codification using a separator is a very good implementation. With this additional rule, the code will looks like SC-APPC and SCA-PPC and the confusion is avoided.

From this reason, the cryptocurrencies dictionary implementation and continuous optimization is a very significant part of the arbitrage trading software for virtual coins. More, the current practice added to this reason another important one. In fact more exchanges are changing the codification of the crypto coins and pairs occasionally. This means reliable trading software must have the possibility for an easy change of a coin code or pair code, in order to save time and to trade the most significant profit opportunities without a significant change in the code of that software.

In this moment (2018 year) there are more than 3000 virtual coins on the free market listed in more than 300 exchanges and we are still counting. Not on the last place, the fact that new crypto coins are invented every day and they are added often into the exchanges quotes makes the dictionary to be a significant tool in order to keep the trading software updated with the reality of cryptocurrency markets.

DOI: 10.12948/issn14531305/22.2.2018.04

4 Data structure

The price quotes are received from the exchanges or from the brokerage companies in real-time in different data structures. Each exchange has its own API application and the data structure of each exchange is different. Even all data sources are available by an API access, the differences between data structures delivered by each exchange make everything more complex. The arbitrage software must be adapted to import all the piece quotes from each exchange according each technical specifications.

Another difference is given by the fact that some exchanges will deliver all the price quotes on a single interrogation of their API, others exchanges will deliver the quotes in different interrogations, one request for each currency pair price. The third level of difficulty is occurred for those exchanges that accept a limited number of requests in a specified time interval. All of these facts make connection with each exchange to be a dedicated and laborious job.

Whatever are the difficulties to implement the data connection with the exchanges, once the price quote is received in real-time, it must to be recorded into the application database. Because the update interval is random, each quote will be inserted as a separate record related with the exchange code, with the pair code and with the updated time. For those processes that need the historical price quotes, because of the large number of the records, usual the history is simplified and represented separately as historical records which will include the next components, that become a standard for a historical price data series [5]: open price of a time interval (OPEN), maximal price of a time interval (MAX), minimal price of a time interval (MIN), close price of a time interval (CLOSE), where the time interval has usual one of the next values: one minute (1M), five minutes (5M), fifteen minutes (15M), thirty minutes (30M), one hour = 60 minutes (1H), four hour = 240 minutes (4H), one day = 1440 minutes (1D), one week = 10080 minutes (1W), one calendar month (1M), one calendar year (1Y).

Usual start time interval and data about the traded volumes are also available together with the price quotes. Advanced software will record also the current spread for the equities, which is the difference between the buy and the sell price at a moment of time.

All price data quotes from hundreds of exchanges and for hundred of thousand of cryptocurrency pairs must to be made in real time. In addition the data history must to be recorder for future operations. Using a relational database is a fact. In addition, the huge number of CRUD operations and the low-latency needed for the trading signals and orders push the application to use multi server architecture presented in Figure 1. In the current practice, more servers will do the acquisition processes of the real time price data from the exchanges, a different server will makes all CRUD (create, record, update, delete) operations and a different one will compute the trading signals and will build and deliver the low-latency trading orders. All of these operations must to be completed in real-time, before the price is changing significantly the values in order to make possible the arbitrage trades.

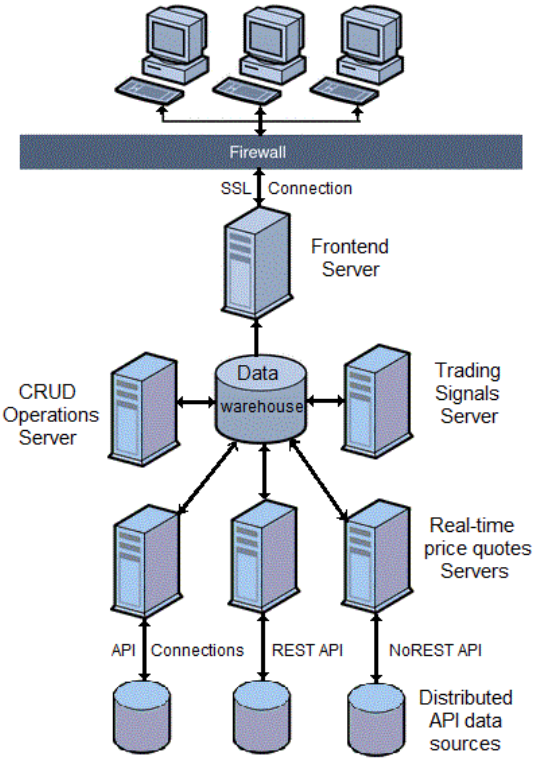

Fig. 1. Server architecture for a cryptocurrency arbitrage system

DOI: 10.12948/issn14531305/22.2.2018.04

5 SQL needed

Due to the large amount of data, first reasons to use a relational database are to control data redundancy, to provide the data integrity and to avoid data inconsistency. The concurrency control needed by using several servers to update the same data warehouse with the real-time price quotes will be also provided by a relational database. In addition, to restrict the data access to different levels of authorization and to provide reliable data backup and recovery, the usage of an SQL relational database will be the solution, even today when NoSQL databases are in trend.

In the context of a large volume of CRUD operations and data requests, SQL will provide the low-latency data responses. The most significant example is the signal generator server. To see how much can be the time response difference we have compared a repetitive process to build the arbitrage trading signals and an SQL procedure to extract the same signals from the database. In the first case, a classical procedure to build the trading signals was wrote to find better prices for each price quote in the database using only repetitive code statements.

For a database including more than 1,000,000 real-time quotes, the process to build the arbitrage trading signals was longer than two minutes on a dedicated server with 2x3GHz processor core and 4 GB RAMM Burst. This time is too long for reliable trading software. In more than two minutes the quotes can be significantly changes, meaning those signals have a low chance to be executed after the delivery on the brokerage account.

In the second case it was implemented an SQL procedure to build the arbitrage trading signals with the same amount of data. The results using SQL are conclusive; the time response was about 0.25 seconds which is a realistic time for the trading software. In Figure 2 is presented a sample code for node.js using callback functions and SQL to build the arbitrage trading signals.

```
//Classical arbitrage trading signals module
const mysql = require("mysql");
const mysqlOptions = {host: "localhost", user: "...", password: "...", database: "..."}
const myConnection = mysql.createConnection(mysqlOptions);
function arbitrageSignals(){
        setInterval(function(){//repeat each minute
        myConnection.query("SELECT q.quote_id, q.exchange_id, q.pair_id, q.buyprice FROM quotes q
                            INNER JOIN (SELECT j.pair_id, COUNT(*) FROM quotes j GROUP BY j.pair_id
                            HAVING COUNT(*) > 1) innerjoin ON q.pair_id=innerjoin.pair_id",
                            function (err, result, fields) {if (err){throw err;}
                            result.forEach(function(row)
                                  {var quote_id=row.quote_id, buyexchange_id=row.exchange_id
                                   var pair_id=row.pair_id; buyprice=row.buyprice;
                                   if(buyprice>0){buildSignal(quote_id, buyexchange_id, pair_id, buyprice);
                                  }}); }); }, 60000);
}
function buildSignal(quote_id, buyexchange_id, pair_id, buyprice){
    myConnection.query("SELECT q.exchange_id, q.sellprice FROM quotes q WHERE q.quote_id<>'"+quote_id+"'
                        AND q.exchange_id<>'"+buyexchange_id+"' AND q.pair_id='"+pair_id+"'
                        AND q.sellprice>'"+buyprice+"'",
    function (err, subresult, fields) {if (err){throw err;}
    subresult.forEach(function(subrow){var sellexchange_id=subrow.exchange_id, sellprice=subrow.sellprice;
                                       var profit=(sellprice-buyprice)/buyprice*100;
    if((sellprice>0)){insertSignal(pair_id, buyexchange_id, buyprice, sellexchange_id, sellprice, profit);
    } }); });
}
function insertSignal(pair_id, buyexchange_id, buyprice, sellexchange_id, sellprice, profit){
    var signaltime=Date.now();
    myConnection.query("INSERT INTO signals (pair_id, buyexchange_id, buyprice, sellexchange_id, sellprice,
                        profit, signaltime) VALUES ('"+pair_id+"','"+buyexchange_id+"','"+buyprice+"','"+
                        sellexchange_id+"','"+sellprice+"','"+profit+"','"+signaltime+"')",
    function (err, result, fields) {if (err){throw err;} });
}
module.exports=arbitrageSignals;
//Classical arbitrage trading signals module
```

Fig. 2. node.js callback functions for the classical arbitrage trading signals module.

DOI: 10.12948/issn14531305/22.2.2018.04

6 API data server

A special particularity of the arbitrage trading systems is the API server that collects real-time quotes from each data source. Each exchange has organized an API application in order to permit restricted and real-time access to the data. To interrogate that API, to receive low-latency data and to insert it in the local database or warehouse, we need to organize a special server process for each exchange. In the Figure 3 it is presented the code for a data acquisition process using an API. The "Exchange" object represents the Rest API procedure module distributed by each exchange. "bookTickers" is the function from this module that exports the price data. Each exchange has its own exporting functions, the name of this function and the "data" structure delivered by this function are different, and consequently the code function must to be adapted for each particular case.

The data delivered by each exchange is usual a JSON object [6] including different data related with the current state of the market. Because the structure of this JSON object is different from an exchange to another, a general procedure cannot be built. In the example presented, the "setInterval" function is a predefined node.js function to repeat the process. In our case we set to repeat the price quotes import each minute (60 sec * 1000 = 60000).

Setting this time is also a procedure depending on each exchange. There are exchanges that allows one per minute requests, other exchanges will allow interrogations one time at two or five minutes and others at ten or larger interval. Usually when the request interval is not respected, the API can cancel the connection with the server and no data will be received from that source. In the example above, the "updateQuote" function is a common function to update the records with the piece quotes, this function will be used for any exchange server, once the data is reduced to an established data structure.

```
1  //Exchange quotes module
2  function exchangeQuotes(exchange_id){if(exchange_id>0){
3  setInterval(function(){Exchange.bookTickers((error, data)=>{
4                          for(item in data)
5                             {var quote=data[item];
6                              var code='', buyprice=0, sellprice=0;
7                              for(item1 in quote)
8                                 {if(item1='symbol'){code=quote[item1];}
9                                  if(item1='bidPrice'){buyprice=quote[item1];}
10                                 if(item1='askPrice'){sellprice=quote[item1];}}
11                                 if((buyprice>0)&&(sellprice>0))
12                                   {updateQuote(code, exchange_id, buyprice, sellprice);
13                             }} }); }, 60000);
14 }}
15 module.exports=exchangeQuotes;
16 //Exchange quotes module
```

Fig. 3. node.js call back functions for the real-time price data acquisition server

Important is the fact that the real-time repetitive price process is a server process, organized for each exchange and depending on the particularities of each API data source. The update time is also depending on the data source. The usage of the node.js server "is quite lightweight and efficient" [7] and permit a fast and simple implementation of these entire server processes. When the number of these modules is a significant one, using node.js we can distribute the repetitive procedures on more distributed servers, as a measure to keep the data delivery time under those limits to ensure the low-latency response.

7 Arbitrage trading signals

In order to consider why we have to do all these technical efforts, in this chapter it will be presented a real case of arbitrage trading signals with cryptocurrencies. Using theCryptoTrader [8] the next signals were obtained at 25 May 2018 using 787 cryptocurrencies listed on 7 different exchanges.

DOI: 10.12948/issn14531305/22.2.2018.04

Table 1. Arbitrage trading signals made by theCryptoTrader on 25.05.2018

Arbitrage trading signal	Buy exchange	Buy price	Sell exchange	Sell price	Profit [%]
BTG-BTC	Cryptopia	0.00101454	Binance	0.00604100	495.44
CTM-BTC	Cryptopia	0.00000929	Binance	0.00004469	381.05
BTC-STEEM	Poloniex	0.00029422	Bittrex	0.00036467	23.94
BTC-SBD	Bittrex	0.00023894	Poloniex	0.00026653	11.55
BTC-OMNI	Bittrex	0.00338255	Poloniex	0.00367822	8.74
BTC-SYS	Poloniex	0.00004503	Bittrex	0.00004895	8.71
BTC-GBP	Kraken	5398.10	Gdax	5653.55	4.73
BTC-NAV	Bittrex	0.00012728	Poloniex	0.00013294	4.45
BTC-DOGE	Poloniex	0.00000046	Bittrex	0.00000048	4.35
BTC-PINK	Bittrex	0.00000250	Poloniex	0.00000258	3.20

As we can see in the Table 1, the profit expectation is a considerable one. The signals presented are the best ten arbitrage trading signals from a list of about 186 available signals found at that moment by the trading software. Any investor who has available money into those exchanges can trade that signal in order to make profit. Usual the signal is not available for a very long period of time. This is the reason to use trading software that has to find the trading opportunities and to trade them with low-latency orders sent direct to the exchanges.

The arbitrage trading software permits a very fast order delivery for each signal direct to the exchanges into the investor's accounts. If the orders arrived before the significant change of the price quotes, they will be executed and the profit will be immediately received into the investor's account. However, time to time, some signals are false. Some exchanges deliver price quotes for some cryptocurrency pairs but they refuse to execute orders at those prices. This is usual the fact when the profit expectation of a signal is too high. Even so, signals with profit expectation under 999% can be well executed in the real practice. The limit is imposed by the exchange by the traded volume which cannot exceed a specified value. Some exchanges limit the trading volume depending on the investor's profile. However, with all that limits, scheduling the execution of a signal to be repeated time to time as a function of the trading software will permit the profit for capitals higher than the trading limits imposed by each exchange. Usual signals with the profit expectation around 20% are well traded.

8 Conclusions

Cryptocurrency market price is not regulated by a central bank. Arbitrage trading opportunities can be found on the free markets in order to make profit. Because of the large number of virtual coins and exchanges, finding and computing the arbitrage trading signals must to be done by the automated trading software. A crypto dictionary is a must to have in the trading software with cryptocurrencies because of the important codes differences between exchanges. In addition, this module will permit all frequent changes in the codification and new virtual coins to be managed.

The technical differences between data sources impose an individual server process to be organized for each data source. A high number of data sources mean a high number of server processes. This implies implementation of more data acquisition process servers distributed in different technical resources.

Due to the large amount of relational data, relational database and SQL is a reliable solution in order to ensure the low-latency response of the trading software. Arbitrage trading software with cryptocurrencies delivers a significant profit.

DOI: 10.12948/issn14531305/22.2.2018.04

References

[1] N. Gandal, H. Halaburda, "Can We Predict the Winner in a Market with Network Effects? Competition in Cryptocurrency Market Design and Implementation", 2016, *NET Institute Working Paper* No. 14-17, Available: https://ssrn.com/abstract= 2506463, May 9, 2018

[2] Wikipedia Encyclopedia - "Financial Cryptography" - https://en.wikipedia.org/wiki /Financial_cryptography, May 10, 2018

[3] M. Iwamura, Y. Kitamura, T. Matsumoto, K.Sailo, "Can We Stabilize the Price of a Cryptocurrency?: Understanding the Design of Bitcoin and Its Potential to Compete with Central Bank Money", *Institute of Economic Research Hitotsubashi University Kunitachi, Tokyo, 186-8603 Japan*, 2014 Available: https://ssrn.com/abstract =2519367, May 9 2018

[4] E. Gatev, "Pairs Trading: Performance of a Relative-Value Arbitrage Rule" - *The Review of Financial Studies*, Volume 19, Issue 3, 2006, Pages 797–827, Available: https://doi.org/10.1093/rfs/hhj020, May 10, 2018.

[5] MetaTrader 4 (MT4) - Historical price data structure - Available: https://docs.mql4.com/ constants/structures/mqlrates, May 10, 2018.

[6] JavaScript Object Notation (JSON) – "The javascript object notation (json) data interchange format" - Available: https://buildbot.tools.ietf.org/html/rfc8259, May 10, 2018.

[7] K. Ley, Y. Ma, Z. Tan, "Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js", 2014 - *Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference*, Available: https://ieeexplore.ieee.org/ document/7023652, May 11, 2018

[8] Cryptocurrency Trading software Platform - theCryptoTrader - Available: https://pauna. biz/thecryptotrader, May 12, 2018

Cristian PĂUNA graduated the Faculty of Cybernetics, Statistics and Economic Informatics of the Academy of Economic Studies in 1999 and also he is a graduate of the Faculty of Aircrafts of the Polytechnic University of Bucharest in 1995. He got the title of Master of Science in Special Aerospace Engineering in 1996. In the last decades he had a sustained activity in software development industry, especially applied in the financial trading domain. Based on several original mathematical algorithms he is the author of more automated trading software. At present he is the Principal Software Developer of Algo Trading Service Ltd. and he is involved as PhD student in Economic Informatics Doctoral School of the Academy of Economic Studies.

DOI: 10.12948/issn14531305/22.2.2018.04