triangular arbitrage, a proposed algorithm

Fernando Augusto Bender
Greenfield Informacoes em Controle e Negocios Financeiros LTDA

triangular arbitrage, a proposed algorithm

fernando a. bender[1]

https://greenfield-br.com

Abstract. a well known trade, arbitrage has been practiced by renowned traders all over the years, amongst them my preferred one, mr. jesse livermore. triangular arbitrage, a slightly more sophisticated variation, is well known in forex. a more recent subclass of such assets, cryptocurrencies provide a whole new arena for this trade in the form of open digital exchange direct access to retail. under this spot, in this paper, we discuss the triangular arbitrage and present an algorithm for trading it profitably. the algorithm assumes that the best book offers of each market can be retrieved seamlessly from an exchange, and they're made available in data arrays readily accessible by the algorithm code. an example is provided to illustrate the algorithm, later on a generic version is obtained. scalability, implementation and performance issues are discussed alongside the design.

Keywords: cryptocurrency · arbitrage · algorithm.

1 introduction

bitcoin started in 2008, but the efforts for a decentralized digital currency date back to the early 80's [1]. although a lot of applications can be derived or be based on such digital assets [2], not much other than coin trading has occurred so far.

regardless which road such technologies might take, they're likely to provide a wide set of assets for trading. it is our understanding that this is enough to allow arbitrage opportunities between them.

1.1 active and passive trading

on our own taxonomy, arbitrage belongs to passive trading category.

on active trading, the trader's view on the market is expressed by a position he assumes, in the side (buying side or selling side), asset and size of his choice, provided there is market liquidity for that.

arbitrage, on the other hand, is much less of trading than observing. the trader (or rather, the observer) relentlessly scans different markets for price inefficiencies. they come in the form of price differences between supply and demand that could afford an intermediary party between them. the most common expression of such is a buyer bidding for an asset in one market more than a seller on another is asking for it. that's an arbitrage opportunity if the asset carry cost between these markets is worth the price difference at the traded volume.

note that in this example the trader has not chosen the asset, the quantity nor the price difference on the trade. he only picks up (or declines) an opportunity that the market offers him. that's why we call it passive trading.

1.2 arbitrage

such price difference can occur in the same market but in different timing, for instance, the corn price might differ between the spot and a future contract in the same or different locations, the so called cash-and-carry arbitrage [3]. a price difference might occur simultaneously in different exchanges, for instance, the same cryptocurrency pair is traded at different prices in different exchanges. that could be exploited in the so called pair trading [4].

lastly, in the same exchange, three pairs composed by three different coins might be simultaneously being traded at prices that are discrepant enough to be exploited with profit, in the so called triangular arbitrage. we'll dive on its specifics through out this paper.

as a last word before we move on, arbitrage opportunities are short lived. thus, it's all about timing. and to prevent you from future losses, in a metaphorical sense, if an arbitrage opportunity lives long, many who trade it won't.

in the next section we detail triangular arbitrage on the context of cryptocurrencies. we then derive its profit conditions.

2 triangular arbitrage in cryptocurrencies

perhaps a good starting question is *why is it called triangular?* the spoiler is it's more about he number three than of that geometric shape.

in short, triangular arbitrage consists of the following; from an initial balance on a given currency, to sell it for another, to sell the latter for yet another one, and finally to exchange it back to the first one. when all's said and done, you end up with more. let's try it with an example.

OKEX [5] is a cryptocurrency exchange, with its own blockchain. it has issued it's own coin and a dollar-pegged coin, one of the so called stable coins. let us assume you have an account on this exchange and that you trade on the spot market. in this context, say

1. you have an initial balance of ETH. you sell it for USDK[1]
2. with your USDK balance, you buy OKB[2]
3. you sell your OKB back for ETH.

there are specific market circumstances, which we'll cover now, when this trade sequence grants you an end balance of ETH higher than the initial one.

first and foremost, we address *how is this even possible.*

[1] stable coin issued by OKEX
[2] token issued by OK blockchain foundation

2.1 the origin of price inefficiencies

the first thing to be noted is that balances are accounted in coins, while coins are traded in pairs. so, in the aforementioned example, you start with a single balance of ETH. any change in this balance that is not a deposit or a withdrawal, will necessarily be a trade[3].

each pair trades at a price, and in a free market assumption, prices float freely to accommodate supply and demand. moves in the price of one pair might not necessarily, and not simultaneously propagate to other related pairs. for example, a surge in OKB/USDK price might not immediately reflect in a surge on the price of OKB/ETH pair. in case there's a demand for OKB, one can expect a steadily surge in all OKB pairs, possibly occurring in decreasing order of liquidity. in case it is caused by a move out of stable coins, it is likely to have no impact on OKB/ETH price.

this pair specific, yet market wide, price discover mechanism is the origin of temporarily price inefficiencies that can give rise to arbitrage opportunities.

2.2 the reason for a three pairs setup

it might sound obvious to some why to arbitrate between three pairs, but there's no practical difference between a forgotten knowledge and a never learnt one.

a simpler setup might be two coins only, where from the initial balance of ETH we sell it for USDK, and then buy back our ETH. the reason it won't work is that two coins constitute a single pair, that you are either buying or selling. if you buy and sell simultaneously the same pair, at best you'll end up with the same amount of ETH. yes, you can never profit by buying and selling simultaneously the same pair in the same exchange.

the only way you can profit from a single pair trading is by opening a temporarily position (for much longer than an arbitrage would require) in one side (long or short), wait for prices to move favourably, and then closing it. that is position trading, which might range from scalping [6] to trend trading [7].

thus, you need a third coin, and that gives rise to a three pairs setup. trading through them allows price inefficiencies to be exploited profitably. one might argue that it could, then, be more than three pairs. yes, it could. however, each trade adds to the cost and to the risk. so, the shortest the path, the quickest the trip, the best.

we revisit our example, computing it with exchange data.

3 an algorithm for triangular arbitrage

from an initial ETH balance, we're interested in increasing it. we, thus, resort to triangular arbitrage which will be traded in three rounds. each round occurs in a specific market. in this example, these markets are

[3] distributions and airdrops are outruled for simplicity.

1. OKB/ETH, where we're buyers, i.e. we buy OKB with ETH.
2. OKB/USDK, where we're sellers, i.e. we sell our OKB for USDK.
3. ETH/USDK, where we're buyers, i.e. we buy ETH with USDK.

in other words, we'll run the triple ETH/OKB/USDK clockwise, and we start by defining some variables

```
balance.ETH[0], initial balance of ETH
balance.ETH[1], balance of ETH at end of round #1
position.ETH[1], ETH position committed in round #1
position.OKB, intermediary OKB position in a given round
position.OKB.market, best bid/ask size for OKB
ask.OKB_ETH[0], best ask for OKB/ETH pair
ask.OKB_ETH[0][0], price of best ask for OKB/ETH pair, in ETH
ask.OKB_ETH[0][1], amount of best ask for OKB/ETH pair, in OKB
fee_taker, worst case fee paid on a spot trade
coef_profit, minimal profitability percentage in decimals.
```

the first round of trades is placed in the OKB/ETH market, where we're interested in buying OKB with ETH. the following data is retrieved from the books

```
position.ETH[1].market = ask.OKB_ETH[0][0] * ask.OKB_ETH[0][1]
position.ETH[1] = min(balance.ETH[0], position.ETH[1].market)
position.OKB = ask.OKB_ETH[0][1]
             * position.ETH[0] / position.ETH[0].market
```

the OKB position assumed in this trade is capped by the amount of the best ask. it is as well dependent on the available balance of ETH. this is why we compare the amount of ETH demanded in this trade (`position.ETH[1].market`) with the current balance to determine how much (percentually) we'll take from the offered OKB amount.

based on this data, an order is issued with the following parameters

```
type = limited
side = buy
market = OKB/ETH
price = ask.OKB_ETH[0][0]
amount = position.OKB
```

once the order is completely executed, the balances become

```
balance.ETH[1] = balance.ETH[0] - position.ETH[1]
balance.OKB[1] = balance.OKB[0]
                + position.OKB * (1 - fee_taker)
balance.USDK[1] = balance.USDK[0]
```

note that the resulting OKB balance is smaller than the OKB position assumed in the trade. the difference is the fees due to the exchange for trade completion.

the second round of trades is placed in the OKB/USDK market, where we're interested in selling OKB for USDK. based on the data we've already gathered and processed, only complementary data is retrieved now

```
position.OKB = min(balance.OKB[1], bid.OKB_USDK[0][1])
```

note that in the previous round, we've compared `position.ETH[1].market` with `balance.ETH[0]` to determine `position.ETH[0]`, while in this round, we compare `balance.OKB[1]` directly with `bid.OKB_USDK[0][1]`. in other words, in OKB/ETH market, the ETH position depends on the product between the best price paid by OKB and its corresponding amount. in the OKB/USDK market, the OKB position depends on the best bid amount, solely. this difference occurs because in the first case we're buyers so we want to know how much money we need to afford the purchased amount, while in the second we're sellers, so the position is straight forward.

in this round, an order is issued with the following parameters

```
type = limited
side = sell
market = OKB/USDK
price = bid.OKB_USDK[0][0]
amount = position.OKB
```

once the order is completely executed, the balances become

```
balance.ETH[2] = balance.ETH[1]
balance.OKB[2] = balance.OKB[1] - position.OKB
balance.USDK[2] = balance.USDK[1]
                  + bid.OKB_USDK[0][0] * position.OKB
                  * (1 - fee_taker)
```

the third round of trades is placed in the ETH/USDK market, where we're interested in buying ETH with USDK. the following data is retrieved

```
position.USDK.market = ask.ETH_USDK[0][0] * ask_ETH_USDK[0][1]
position.USDK = min(balance.USDK[2], position.USDK.market)
position.ETH[3] = ask.ETH_USDK[0][1]
                  * position.USDK / position.USDK.market
```

and a final order is issued with the following parameters

```
type = limited
side = buy
market = ETH/USDK
price = ask.ETH_USDK[0][0]
amount = position.ETH[3]
```

once the order is completely executed, the final balances become

```
balance.ETH[3] = balance.ETH[2]
                 + position.ETH[3] * (1 - fee_taker)
balance.OKB[3] = balance.OKB[2]
balance.USDK[3] = balance.USDK[2] - position.USDK
```

and the trade is concluded. this is the first result of this paper.

3.1 clock, anticlock senses

it is worth noting that the trade sequence occurred in the so called clockwise, ETH/OKB/USDK order. the reverse sense ETH/USDK/OKB could have been chosen. the differences would be regarding the order we visit each market and our side in them. while trading clockwise, we start as buyers on OKB/ETH market, in anticlockwise, we start as sellers on ETH/USDK market. thus, in the former we look for asks, and in the latter we look for bids.

we believe this is enough to straighten forward this algorithm in the anti-clockwise sense. we now proceed to present the profit conditions in either sense for this same example.

3.2 profit conditions

if the trade was profitable, we get

```
balance.ETH[3] > balance.ETH[0]
```

which can be expanded as

```
balance.ETH[2] + position.ETH[3]
                 * (1 - fee_taker) > balance.ETH[0]
balance.ETH[1] + position.ETH[3]
                 * (1 - fee_taker) > balance.ETH[0]
balance.ETH[0] - position.ETH[1]
+ position.ETH[3] * (1 - fee_taker) > balance.ETH[0]
```

which can be represented in terms of ETH positions as

```
position.ETH[3] * (1 - fee_taker) > position.ETH[1]
```

it is our interest, however, to know whether the trade is likely to be profitable, upfront. we thus assume we're trading without limitations from coin balances, or bids and asks amounts. that allows us to estimate the maximum obtainable profit. we then are lead to the following condition[4]

```
bid.OKB_USDK[0][0] * (1 - fee_taker)^3
>= ask.OKB_ETH[0][0] * ask.ETH_USDK[0][0] * (1 + coef_profit)
```

[4] the complete development of this condition is not included in this paper.

this is a sufficient condition for profit clockwise. an equivalent anticlockwise condition would be

```
bid.ETH_USDK[0][0] * bid.OKB_ETH[0][0] * (1 - fee_taker)^3
>= ask.OKB_USDK[0][0] * (1 + coef_profit)
```

some takeaways

- **these are sufficient conditions**, so most of the time none is likely to be true.
- **they are mutually exclusive**. never, ever both will be simultaneously true. if so happens your code is wrong.
- **inherent imperfection**. as there's nothing bounding the triple pairs together, one can always expect an inequality, even in a no profit, no fee scenario. the imbalance might not be sufficiently high to cover for the fees, or to cross the threshold profit for risk taking.
- **zero sum game**. the imbalance clockwise is opposite to the imbalance otherwise at the same instant or market conditions. *and in the end, the love you take is equal to the love you made.*

this is the second result of this paper.

4 generalized profit conditions

in order to exploit market opportunities, trading cannot be confined to a single or few triples. however, scaling up the process shouldn't take along the costs. understanding the *rationale* behind the profit conditions might allow for generalizations, which in turn might grant us scale economy.

4.1 a prologue on generalization

is it even possible? with some contour conditions, yes. we'll address them as we evolve in the generalization.

good sense restrictions on coins and exchange.

- **the triple should be tradable**. pick for instance, the triple BTC/LTC/XRP. you cannot trade them as we did in previous section. why? there's no way you can sell LTC directly in XRP or to buy XRP directly in LTC[5]. you need first to sell your LTC for USDK, BTC, ETH or OKB. then you get to buy your XRP. that just added another coin to the triple, making them a quadruple. no way.
- **hold on to what is worth**. *buy into a company because you want to own it, not because you want the stock to go up*, warren buffet's quote. although its execution is quite personal, it wipes out most of the existing coins, nowadays.

[5] as of the writing of this paper, there is no XRP or LTC market in OKEX.

– **there's no good deal in a bad marketplace**. choose carefully your exchange.

based on the aforementioned guidelines, we reach the following coins set for the current time being[6].

– `BTC, ETH, XRP, LTC, EOS` as hold
– `USDK, OKB, BCH, XMR` as additional coins for the trading set
– OKEX as the exchange for performing the trades

it remains to address how to make this coin set tradable in the context of triangular arbitrage.

4.2 on pairing the coins

naturally, the larger the set of coins considered, the greater the trading possibilities, and hopefully the larger the profits. at the same time, not all the combinations exist. we consider these aspects in the sequel.

triple tradability conditions. as just posed, BTC/LTC/XRP cannot be traded. the reason is the current impossibility to trade LTC directly for XRP within the exchange spot markets. in other words, there's no LTC/XRP or XRP/LTC pairs.

this impossibility highlights directly the solution. there should be no adjacent coins comprising an untradable pair. conversely, adjacent coins should comprise tradable pairs.

to further enlight the tradability of a pair, we segregate the coins against which, other coins are priced. in other words, coins that are base currencies in markets. to implement this segregation, we suggest a boolean valued property, `marketness`. to ease for the reader, we classify coins as **marketness** or **marketless** (for those whose **marketness** property is false).

two coins are not directly tradable if both are marketless. as a consequence, *a triple can have at most one marketless coin*, regardless the order.

we propose the notation $\mathtt{M}$, $\bar{\mathtt{M}}$ for marketness, marketless, respectively. tradable triples are thus $(\mathtt{M},\mathtt{M},\mathtt{M})$, or $(\bar{\mathtt{M}},\mathtt{M},\mathtt{M})$, or $(\mathtt{M},\bar{\mathtt{M}},\mathtt{M})$, or $(\mathtt{M},\mathtt{M},\bar{\mathtt{M}})$. these are the stereotypes of tradable triples. as a byproduct, we're now able to pair $\bar{\mathtt{M}},\mathtt{M}$ coins, but what about $\mathtt{M}/\mathtt{M}$ pairs?

pairing marketness coins. marketness coins require a score to differentiate them, so pairing can be done through the same concept of $\bar{\mathtt{M}},\mathtt{M}$ pairs. for instance, both BTC and ETH are $\mathtt{M}$ coins, but there's no BTC/ETH pair, only ETH/BTC. same for USDK and BTC. there's no USDK/BTC pair, only BTC/USDK. this restriction must be captured in a score. we then build the pairs according to it.

the score we propose is headed by stable coins, followed by BTC, ETH, then exchange coin. one could expect that stable coins should be ordered by liquidity,

[6] today is 2019/09/21

but exchanges may have their own criteria for listing their stable coin pairs. for two `M` coins in a pair, the lower indexed coin[7] is the base currency[8], and the other is the quote currency[9].

notice that there's no relation between adjacency in a triple, and the position of coins in the pair traded. for instance, a triple (A,B,C) can imply in a pair A/B or B/A, and that is determined by their marketness property; not by the A, B position in the triple. A preceding B in the triple means only that a balance in A coin shall be converted to a balance in B coin; that might be achieved by either selling A/B or buying B/A.

according to the proposed score we define our array of `M` coins as

```
arrayMarketness = [USDK, BTC, ETC, OKB]
```

the pertinence of a coin to the above array is the value of its marketness property, and its array position set its score.

we'll represent as `M`, `m`, `n`, three marketness coins with increasing index value, i.e. decreasing marketness score.

we now revisit the tradable triples of previous subsection, expanding their combinations according to the marketness property and score values of the coins.

- $(\mathtt{M},\mathtt{M},\mathtt{M})$; leads to the following combinations $(\mathtt{M},\mathtt{m},\mathtt{n})$, $(\mathtt{m},\mathtt{M},\mathtt{n})$, $(\mathtt{n},\mathtt{M},\mathtt{m})$.
- $(\mathtt{M},\mathtt{M},\bar{\mathtt{M}})$; leads to $(\mathtt{M},\mathtt{m},\bar{\mathtt{M}})$, $(\mathtt{m},\mathtt{M},\bar{\mathtt{M}})$.
- $(\mathtt{M},\bar{\mathtt{M}},\mathtt{M})$; is covered by the above, anticlockwise.
- $(\bar{\mathtt{M}},\mathtt{M},\mathtt{M})$; has only $(\bar{\mathtt{M}},\mathtt{M},\mathtt{m})$ as unaddressed combination.

in the matter of coin pairing, `n` and $\bar{\mathtt{M}}$ are always quote currencies in their respective pairs. moreover, in their respective triples they hold the lower marketness. in this context, we combine the triples where `n` and $\bar{\mathtt{M}}$ have the same position, reducing the combinations to

$$(\mathtt{M},\mathtt{m},\bar{\mathtt{M}}+\mathtt{n}),(\mathtt{m},\mathtt{M},\bar{\mathtt{M}}+\mathtt{n}),(\bar{\mathtt{M}}+\mathtt{n},\mathtt{M},\mathtt{m}).$$

it simplifies the ruling, for as for each triple with an $\bar{\mathtt{M}}$ coin, there will be an exact one where in its place there's an `n` coin. both are ruled by the same profit conditions and trading sequence. we thus get the following pairs, clockwise.

- $(\mathtt{M},\ \mathtt{m},\ \bar{\mathtt{M}}+\mathtt{n})$; $\mathtt{m/M}$, $\{\bar{\mathtt{M}}+\mathtt{n}\}/\mathtt{m}$, $\{\bar{\mathtt{M}}+\mathtt{n}\}/\mathtt{M}$
- $(\mathtt{m},\ \mathtt{M},\ \bar{\mathtt{M}}+\mathtt{n})$; $\mathtt{m/M}$, $\{\bar{\mathtt{M}}+\mathtt{n}\}/\mathtt{M}$, $\{\bar{\mathtt{M}}+\mathtt{n}\}/\mathtt{m}$
- $(\bar{\mathtt{M}}+\mathtt{n},\ \mathtt{M},\ \mathtt{m})$; $\{\bar{\mathtt{M}}+\mathtt{n}\}/\mathtt{M}$, $\mathtt{m/M}$, $\{\bar{\mathtt{M}}+\mathtt{n}\}/\mathtt{m}$

this pairs will be traded clockwise and anticlockwise. note that the first pair in the first and second items is `m/M`. however, in the first item we're in the buy side because we're exchanging `M` for `m`. the other is vice versa.

[7] the lower the index, the higher the marketness of the coin.
[8] the subsequent coin in a pair, i.e. Y in X/Y.
[9] the preceding coin in the pair, i.e. X in X/Y.

4.3 on trading the pairs

in section 4.2 we've paired the coins in triples. we'll now get their buy/sell side by abiding to: adjacent coins in decreasing order of marketness lead to a buy order, and vice versa.

– $(\mathtt{M}, \mathtt{m}, \bar{\mathtt{M}} + \mathtt{n})$; **clockwise**; buy, buy, sell. **anticlockwise**; buy, sell, sell.
– $(\mathtt{m}, \mathtt{M}, \bar{\mathtt{M}} + \mathtt{n})$; **clockwise**; sell, buy, sell. **anticlockwise**; buy, sell, buy.
– $(\bar{\mathtt{M}} + \mathtt{n}, \mathtt{M}, \mathtt{m})$; **clockwise**; sell, buy, buy. **anticlockwise**; sell, sell, buy.

combining the previous subsection pairs with the above orders we get the following sequence.

– $(\mathtt{M}, \mathtt{m}, \bar{\mathtt{M}} + \mathtt{n})$;
  - clockwise; **buy** $\mathtt{m}/\mathtt{M}$, **buy** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{m}$, **sell** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{M}$
  - anticlockwise; **buy** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{M}$, **sell** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{m}$, **sell** $\mathtt{m}/\mathtt{M}$
– $(\mathtt{m}, \mathtt{M}, \bar{\mathtt{M}} + \mathtt{n})$;
  - clockwise; **sell** $\mathtt{m}/\mathtt{M}$, **buy** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{M}$, **sell** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{m}$
  - anticlockwise; **buy** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{m}$, **sell** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{M}$, **buy** $\mathtt{m}/\mathtt{M}$
– $(\bar{\mathtt{M}} + \mathtt{n}, \mathtt{M}, \mathtt{m})$;
  - clockwise; **sell** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{M}$, **buy** $\mathtt{m}/\mathtt{M}$, **buy** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{m}$
  - anticlockwise; **sell** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{m}$, **sell** $\mathtt{m}/\mathtt{M}$, **buy** $\{\bar{\mathtt{M}} + \mathtt{n}\}/\mathtt{M}$

4.4 generalized profit conditions

based on the developments here presented so far, it is straightforward (although worksome) to derive generalized profit conditions for our resulting set of triples.

– $(\mathtt{M}, \mathtt{m}, \bar{\mathtt{M}} + \mathtt{n})$;
  - clockwise;
```
bid.{M̄+n}_M[0][0] * (1 - fee_taker)^3
>= ask.m_M[0][0] * ask.{M̄+n}_m[0][0] * (1 + coef_profit)
```
  - anticlockwise;
```
bid.{M̄+n}_m[0][0] * bid.m_M[0][0] * (1 - fee_taker)^3
>= ask.{M̄+n}_M[0][0] * (1 + coef_profit)
```
– $(\mathtt{m}, \mathtt{M}, \bar{\mathtt{M}} + \mathtt{n})$;
  - clockwise;
```
bid.m_M[0][0] * bid.{M̄+n}_m[0][0] * (1 - fee_taker)^3
>= ask.{M̄+n}_M[0][0] * (1 + coef_profit)
```
  - anticlockwise;
```
bid.{M̄+n}_M[0][0] * (1 - fee_taker)^3
>= ask.{M̄+n}_m[0][0] * ask.m_M[0][0] * (1 + coef_profit)
```
– $(\bar{\mathtt{M}} + \mathtt{n}, \mathtt{M}, \mathtt{m})$;
  - clockwise;
```
bid.{M̄+n}_M[0][0] * (1 - fee_taker)^3
>= ask.m_M[0][0] * ask.{M̄+n}_m[0][0] * (1 + coef_profit)
```
  - anticlockwise;
```
bid.{M̄+n}_m[0][0] * bid.m_M[0][0] * (1 - fee_taker)^3
>= ask.{M̄+n}_M[0][0] * (1 + coef_profit)
```

this is the third result of this paper. in the next section we'll rank them by profit.

5 triples profit rank

naturally, from all the triples he's interested, a trader is likely to prefer the most profitable he's able to trade right now.

the conditions obtained in section 4.4 are boolean and sufficient to ensure desired profit is achieved, whether clockwise or reversed. they're not suited for ranking. we, thus, rearrange those conditions to allow for ranking.

– ($\mathtt{M}$, $\mathtt{m}$, $\bar{\mathtt{M}} + \mathtt{n}$);
  - clockwise;
    `bid.{M̄+n}_M[0][0] * (1 - fee_taker)^3`
    `- ask.m_M[0][0] * ask.{M̄+n}_m[0][0] =` ε_M
  - anticlockwise;
    `bid.{M̄+n}_m[0][0] * bid.m_M[0][0] * (1 - fee_taker)^3`
    `- ask.{M̄+n}_M[0][0] =` ε_M

– ($\mathtt{m}$, $\mathtt{M}$, $\bar{\mathtt{M}} + \mathtt{n}$);
  - clockwise;
    `bid.m_M[0][0] * bid.{M̄+n}_m[0][0] * (1 - fee_taker)^3`
    `- ask.{M̄+n}_M[0][0] =` ε_m
  - anticlockwise;
    `bid.{M̄+n}_M[0][0] * (1 - fee_taker)^3`
    `- ask.{M̄+n}_m[0][0] * ask.m_M[0][0] =` ε_m

– ($\bar{\mathtt{M}} + \mathtt{n}$, $\mathtt{M}$, $\mathtt{m}$);
  - clockwise;
    `bid.{M̄+n}_M[0][0] * (1 - fee_taker)^3`
    `- ask.m_M[0][0] * ask.{M̄+n}_m[0][0] =` $\varepsilon_{\bar{M}+n}$
  - anticlockwise;
    `bid.{M̄+n}_m[0][0] * bid.m_M[0][0] * (1 - fee_taker)^3`
    `- ask.{M̄+n}_M[0][0] =` $\varepsilon_{\bar{M}+n}$

previous conditions were rearranged above in equations, where the profit measured in the initial coin (`M, m` or `M̄+n`) is represented by the respective ε.

this equations allow for ranking the profit within triples with the same initial coin, for instance, ETH. the reason is straightforward. you cannot compare a profit in ETH with a profit in XRP, for example. it is enough, though, for single coin holders. in this case they'll test the conditions whose triple starts with the same marketness of their coin.

traders with a broader portfolio, usually want more. we, thus, need to rearrange these equations to account for percentual profit,

– ($\mathtt{M}$, $\mathtt{m}$, $\bar{\mathtt{M}} + \mathtt{n}$);
  - clockwise;
    `bid.{M̄+n}_M[0][0] * (1 - fee_taker)^3`
    `/ ask.m_M[0][0] / ask.{M̄+n}_m[0][0] =` θ
  - anticlockwise;
    `bid.{M̄+n}_m[0][0] * bid.m_M[0][0]`
    `* (1 - fee_taker)^3 / ask.{M̄+n}_M[0][0] =` θ

- (m, M, $\bar{\texttt{M}}$ + n);
  - clockwise;
    `bid.m_M[0][0] * bid.{M̄+n}_m[0][0]`
    `* (1 - fee_taker)^3 / ask.{M̄+n}_M[0][0] =` θ
  - anticlockwise;
    `bid.{M̄+n}_M[0][0] * (1 - fee_taker)^3`
    `/ ask.{M̄+n}_m[0][0] / ask.m_M[0][0] =` θ
- ($\bar{\texttt{M}}$ + n, M, m);
  - clockwise;
    `bid.{M̄+n}_M[0][0] * (1 - fee_taker)^3`
    `/ ask.m_M[0][0] / ask.{M̄+n}_m[0][0] =` θ
  - anticlockwise;
    `bid.{M̄+n}_m[0][0] * bid.m_M[0][0]`
    `* (1 - fee_taker)^3 / ask.{M̄+n}_M[0][0] =` θ

now you can compare any tradable triple from your coin set by ranking the obtained θ. this is our fourth, and perhaps more relevant, result of this paper.

5.1 an epilogue on generalization

we now present an algorithm for trading a set of coins according to our latest generalized profit conditions. we illustrate it with our coin set of section 4.1.

define a set of coins. segregate those you'd hold.

```
arrayCoins.hold = [BTC, ETH, XRP, LTC, EOS]
arrayCoins.additional = [USDK, OKB, BCH, XMR]
```

segregate them in marketness. those with these property, index them by it.

```
arrayCoins.marketness = [USDK, BTC, ETH, OKB]
arrayCoins.marketless = [XRP, LTC, EOS, BCH, XMR]
```

build the triples. all triples as `(coin1, coin2, coin3)`

1. `coin1` is in `arrayCoins.hold`
2. if `coin1` is in `arrayCoins.marketless`, then `coin2` is from `arrayCoins.marketness`. `coin2` is different of `coin1`.
3. if `coin1` or `coin2` is in `arrayCoins.marketless`, then `coin3` is from `arrayCoins.marketness`. `coin3` is different of `coin1` and `coin2`.

group the triples per stereotype. stereotypes are as follows

- $(\mathtt{M}, \mathtt{m}, \bar{\mathtt{M}} + \mathtt{n})$;
- $(\mathtt{m}, \mathtt{M}, \bar{\mathtt{M}} + \mathtt{n})$;
- $(\bar{\mathtt{M}} + \mathtt{n}, \mathtt{M}, \mathtt{m})$;

for simplicity we'll represent $\bar{\mathtt{M}} + \mathtt{n}$ solely by `n`, without any loss of generality.

```
arrayTriples.Mmn = [[USDK, BTC, ETH], [BTC, ETH, XRP], ...]
arrayTriples.mMn = [[BTC, USDK, ETH], [ETH, BTC, XRP], ...]
arrayTriples.nMm = [[ETH, USDK, BTC], [XRP, BTC, ETH], ...]
```

this is a one time procedure for any new set of coins. on it, all triples matching each stereotype are generated and cast in their respective array. the arrays remain valid as far as the set remains unchanged. they can be stored in a file, an a trading process reads them at its start up.

apply the profit conditions on the triples. within each array.

1. for triples within `arrayTriples.Mmn` replace $(\mathtt{M}, \mathtt{m}, \bar{\mathtt{M}} + \mathtt{n})$ by (`coin1, coin2, coin3`) so that condition
   `bid.{M̄+n}_M[0][0] * (1 - fee_taker)^3`
   `/ ask.m_M[0][0] / ask.{M̄+n}_m[0][0] =` θ
   becomes
   `bid.coin3_coin1[0][0] * (1 - fee_taker)^3`
   `/ ask.coin2_coin1[0][0] / ask.coin3_coin2[0][0] =` θ
2. run these conditions (clockwise and anticlockwise) against all the triples within this array.
3. store the θ obtained in each triple on an array

   ```
   arrayTriples.Mmn.theta = [theta0, theta1, ...]
   ```

   where `theta0` corresponds to θ of the clockwise profit condition of the first triple, and `theta1` is its anticlockwise θ. thus, the length of such array must be the double of the number of the triples. the best profit can be readily mapped to its corresponding triple.
4. repeat the process on triple arrays `arrayTriples.mMn`, `arrayTriples.nMm`.

this procedure will be executed periodically. it can be executed in parallel, each by a different vCPU. the largest θ of each array are then compared with each other to determine the largest amongst them.

trade the winners. for each triple you decide to trade, follow the sequence described in section 3. on it

1. replace the coins symbols by `coin1`, `coin2`, `coin3`, according to their order of trading
2. replace `coin1`, `coin2`, `coin3` by the coins of your specific triple, according to their respective position in the triple.

the first item is a one time procedure. the second is ran at each trade, and can be factored in a function with `coin1`, `coin2`, `coin3` as parameters. it is out of the scope of this work to dive in such details.

this concludes our work.

References

1. https://doi.org/10.1007/978-1-4757-0602-418
2. https://doi.org/10.1016/j.bushor.2018.03.006
3. https://www.investopedia.com/terms/c/cash-and-carry-arbitrage.asp
4. https://www.investopedia.com/terms/p/pairstrade.asp
5. https://www.okex.com
6. https://www.investopedia.com/articles/trading/05/scalping.asp
7. https://www.investopedia.com/terms/t/trendtrading.asp