GENERATE ORERFLOW FROM TRADES AND QUOTES IN CRYPTOCURRENCY MARKET

Tinglu Xu
Henry Boakye Antwi
Williams Botchway

Abstract

This paper describes a method to re-create order flow from aggregated trade and tick data (best bid and ask) that is commonly available for crypto-currency exchanges. The trade and best quote data only represent partial information of the whole orderflow, which would including insertion, trade or cancellation of limit order book (LOB). Finally, we apply the algorithm on Tick and Trade data on BTC perpetual future contract from Binance Futures.

Literature Review

Limit order book (LOB) is essential component in financial exchanges to understand the market micro-structure. (Luckock 2001, Gu et al. 2008b). It represent the state of book on seller and buyer at point-in-time. However, the LOB data in crypto-market are normally in snapshot at 100ms or evenly lower frequency. Trade and Quotes data are more frequently provided also incomplete. In this paper we will develop a method to inferring orderflow from partial trade and quote data from Binance Futures, the most liquid derivative exchange on trading crypto-market.

Note some exchanges do provide L3 Orderflow data like Coinbase on spot market, generally L3 data are not available in most of the exchanges.

Order flow data contains all orders contains from market participants, mainly are:

- Limit orders (Bid and Ask)
  These are the orders that potentially market participants are willing to sell and buy at a pre-defined prices, which are typically price cannot be match instantly but willing to wait for a better price.
- Market orders.
  These are order willing to buy and sell at market prevalent price and it can match with other limit order instantly. Market orders are time-sensitive, meaning they are be executed instantly but cannot guarantee the prices.

For back testing, purpose, any new coming orders would be either match with current order in the LOB book, or queuing in the LOB until it is matched with other order or cancelled.

This paper argues that for High Frequency Trading event using order flow data is of real advantage. However even with event base back tester latency still remains a crucial factor to consider when developing a back tester for High Frequency trading. A change in price between when an order is routed and when an order is executed known as slippage needs to be taken into account in determining the amount of latency to simulate in a back tester. It is important to indicate that no matter what consideration is put in designing a back tester there will always be some amount of slippage as a result of communication between various computing nodes.

The unusually rich, detailed, and high-quality historic data from Limit Order Book provides a suitable testing ground for theories about well-established statistical regularities common to crypto currency and a wide range of markets (Farmer and Lillo 2004, Bouchaud et al. 2009) With the help of back-testing engine along with good quality of LOB order flow data, we should be able to evaluate different market making strategies. For example, (Avellaneda & Stoikov, 2008) , (Spooner et al., 2018) and (Abernethy & Kale, 2013)

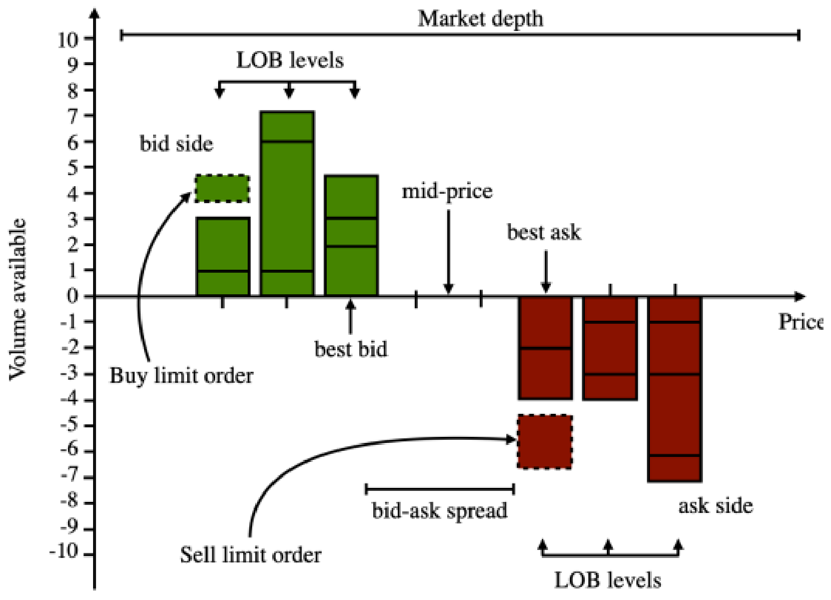
Limit Order Book

The following diagram from (Briola et al., 2020) show what a typical LOB looks like, this is point-in-time snapshot and at least we can extra the following information:

- Best bid price and bid size
  The highest price and amount that buyer are willing to buy for instant transactions.
- Best ask price and ask size
  The lower price and amount that seller are willing to sell for instant transaction.
- Depth of book
  The depth of either side of book is the cumulative quantity on bid (right) and ask (left) queues
- Bid-ask spread
  The gap between best bid and best ask is the spread that no limit orders are in. More liquid market would have tighter spread.

When new order arrives, it can either match current queued order (if buy order equal or higher than best ask, or sell order equal or lower than best bid), or insert into the queue with other order. In order-driven markets, the priority of orders to be matched at each price level depends upon the arrival time, according to a FIFO (First In, First Out) rule.

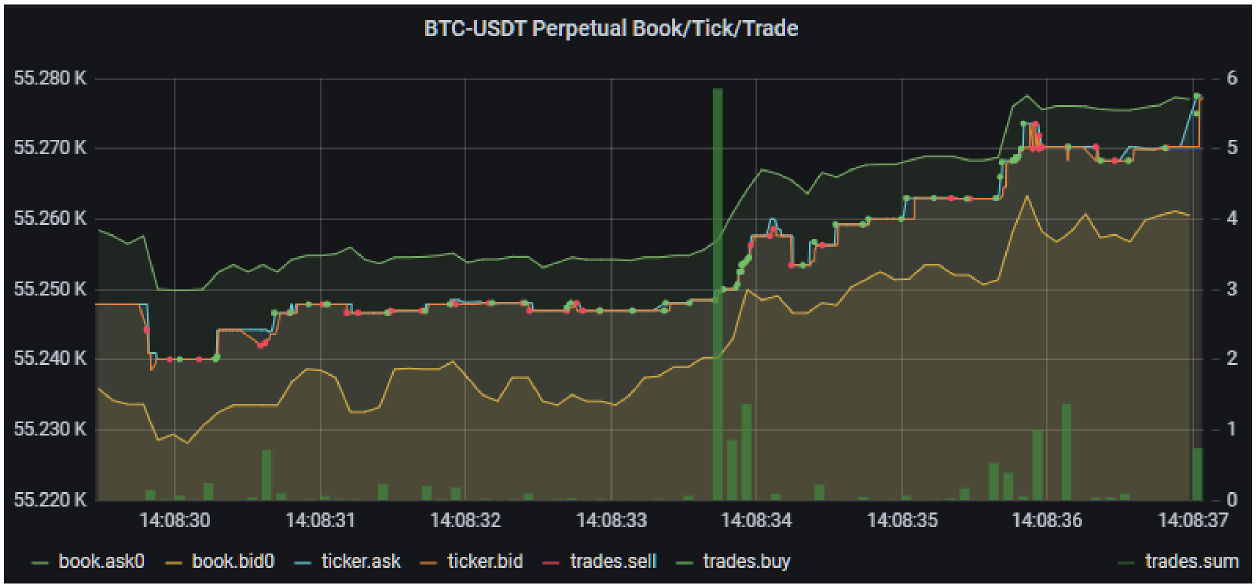
If we start LOB as a blank canvas, order flow will insert or remove (by either matching or cancel) orders in the LOB. After any new order, LOB would be updated to display the current data of the order book.



Data source

Data overview:

- Exchange: Binance Futures
- BTC-USDT Perpetual contract
- Tick and Trade data fetched from public websocket.
    - Trades is aggregated data at maximum 100ms
    - Tick is real-time best bid and ask price and size.



Data collection:

Data are collected in official API from Binance futures, and stored in Influx data base.

- Binance API
  The base endpoint is: https://fapi.binance.com
  Usage information can be found in https://binance-docs.github.io/apidocs/futures/en

- **Trades** data (Aggregate Trade Streams):
  <symbol>@aggTrade
  The Aggregate Trade Streams push trade information that is aggregated for a single taker order every 100 milliseconds. The trades will include the data with same direction, same price and within 100ms horizon.
  The data structure of trades data show as below. The following highlighted fields are used for the purpose of this paper.

```
{
  "e": "aggTrade",  // Event type
  "E": 123456789,   // Event time
  "s": "BTCUSDT",    // Symbol
  "a": 5933014,     // Aggregate trade ID
  "p": "0.001",     // Price
  "q": "100",       // Quantity
  "f": 100,         // First trade ID
  "l": 105,         // Last trade ID
  "T": 123456785,   // Trade time
  "m": true,        // Is the buyer the market maker?
}
```

- **Ticks** data (Individual Symbol Book Ticker Streams):
  <symbol>@bookTicker
  Pushes any update to the best bid or ask's price or quantity in real-time for a specified symbol. The data structure of trades data show as below. The following highlighted fields are used for the purpose of this paper.

```
{
  "e":"bookTicker",        // event type
  "u":400900217,           // order book updateId
  "E": 1568014460893,      // event time
  "T": 1568014460891,      // transaction time
  "s":"BNBUSDT",           // symbol
  "b":"25.35190000",       // best bid price
  "B":"31.21000000",       // best bid qty
  "a":"25.36520000",       // best ask price
  "A":"40.66000000"        // best ask qty
}
```

Methodology

Market making strategy will add limit order to the order book (one side or both sides), which will match active order (taker) and make a trade. Marking making strategy show the price that trade will be place and will provide liquidity to the market. As a result, the fees for market maker pays a lower transaction fee than market take, who drain liquidity of the market. The maker fee can be 0 and sometimes negative as incentive to offer market liquidity. There are more complex order type like Iceberg order which doesn't reveal the full Size in initial order book.

Order flow can arrive LOB in 3 types of actions:

1. Insertion
   Add new queue in LOB.
2. Cancellation
   Remove quantity from LOB
3. Trade
   Match with existing LOB and reduce quantity in LOB.

Type 1 will increase the quantity of LOB, while Type 2 and 3 will reduce the quantity. The main difference between 2 and 3 is that Cancellation can reduce quantity in LOB queue at any position, while Trade only reduce the best bid/ask quantities.

Direction for each action was categorized as below. The rational for each direction is that Direction=1 indicate the action will increase mid price=0.5*(best bid + best ask), while direction=0 is likely cause mid price to drop.

For order insertion, if new bid is added at best bid, the mid price will move up, corresponding to direction=1, vice versa, cancellation of best bid will cancel mid price so direction=0

- If there is Trade=buy, it will reduce best ask price and push up mid price, so direction=0, and vice versa sell order on direction=0

action	**direction=1**	**direction=0**
insertion	Bid	ask
trade	Buy	sell
cancel	Ask	bid

It is also noticed that Trade timestamp will be earlier than Tick data updates, possibly because that Timestamp was from the first match timestamp, and tick data was updated after order has been matched.

Implementation

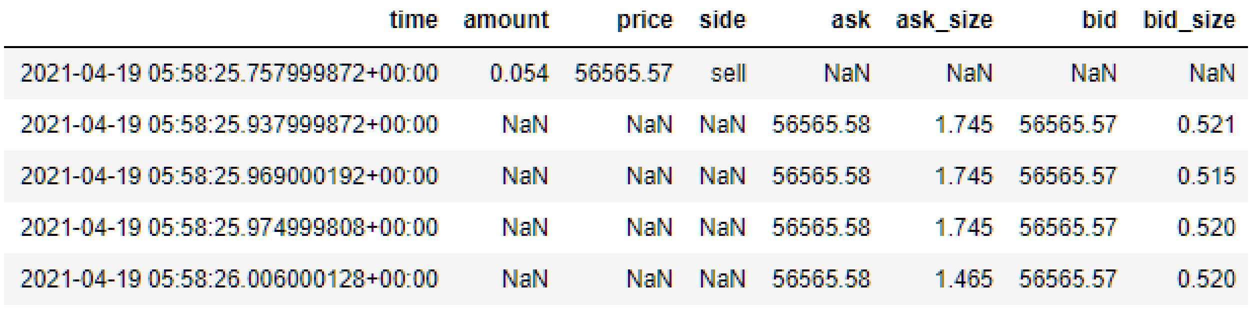
The implementation source code is uploaded to https://github.com/txu2014/binance_orderflow.git

Required data format:

Padas dataframe in python with following header, it is concatenated from Trade and Tick data and sorted by time. We can notice the following:

- Tick LOB will update due to either bid or ask updated. We assume no update for bid or ask if its price and size are the same as previous timestamps.
- Trade time stamp is earlier than Tick LOB update.
- Trade data are aggregated by 100ms and the same price, meaning:
    - The market order time will be later than reported time

- It may be several order of the same direction and price been aggregated into the same reporting. We will try our best to decompose this.



	time	amount	price	side	ask	ask_size	bid	bid_size
	2021-04-19 05:58:25.757999872+00:00	0.054	56565.57	sell	NaN	NaN	NaN	NaN
	2021-04-19 05:58:25.937999872+00:00	NaN	NaN	NaN	56565.58	1.745	56565.57	0.521
	2021-04-19 05:58:25.969000192+00:00	NaN	NaN	NaN	56565.58	1.745	56565.57	0.515
	2021-04-19 05:58:25.974999808+00:00	NaN	NaN	NaN	56565.58	1.745	56565.57	0.520
	2021-04-19 05:58:26.006000128+00:00	NaN	NaN	NaN	56565.58	1.465	56565.57	0.520

In practice, the algorithm was implemented in python, which take following steps:

1. Identify LOB insertions
    a. For ask side, it will be:
        i. Best ask increased size
        ii. New ask price has lower price

```
#ask insertion
df_ask['ask_price_delta']=df_ask['ask'].diff()
df_ask['ask_size_delta']=df_ask['ask_size'].diff()
df_ask_insertion=df_ask[((df_ask['ask_price_delta'].abs()<EPS) &(df_ask['ask_size_delta']>0))|(df_ask['ask_price_delta']<0)].copy()
df_ask_insertion['size'] = df_ask_insertion['ask_size_delta']
df_ask_insertion['direction']=0
df_ask_insertion['action']='insertion'
df_ask_insertion['price']=df_ask_insertion['ask']
```

    b. For bid side, it will be:
        i. Best bid increase size
        ii. New bid price is higher price

```
#bid insertion
df_bid['bid_price_delta']=df_bid['bid'].diff()
df_bid['bid_size_delta']=df_bid['bid_size'].diff()
df_bid_insertion=df_bid[((df_bid['bid_price_delta']==0) &(df_bid['bid_size_delta']>0))|(df_bid['bid_price_delta']>0)].copy()
df_bid_insertion['size'] = df_bid_insertion['bid_size_delta']
df_bid_insertion['direction']=1
df_bid_insertion['action']='insertion'
df_bid_insertion['price']=df_bid_insertion['bid']
```

2. Identify LOB reduction, It can be caused by:
    a. LOB Cancellation
       Trade data has no match on reduction, meaning

```
#ask reduce (can be buy, or cancel ask)
```

```
df_ask_reduce=df_ask[((df_ask['ask_price_delta']==0) &(df_ask['ask_size_delta']<0))|(df_ask['ask_price_delta']>0)]
df_ask_reduce_merge = pd.concat([df_trade[df_trade['side']=='buy'], df_ask_reduce], axis=1)
df_ask_reduce_merge[['amount', 'price']] = df_ask_reduce_merge[['amount', 'price']].fillna(method='ffill')
df_ask_reduce_merge['direction'] = 1
# ask cancellation.  cancel amount=ask_size_delta
df_ask_cancel1 = df_ask_reduce_merge[(df_ask_reduce_merge['price']!=df_ask_reduce_merge['ask'])&(df_ask_reduce_merge['ask'].notnull())]
```

i. Ask size decrease but no buy order can match the price, or
   Ask price decrease more than Buy order amount
ii. Bid size decrease but no Sell order can match that price, or
   Bid size decrease more than Sell order amount.

b. Trade that match against existing order, meaning
   i. Buy order match the price of Ask
      It will be split into multiple trades if price is different,
      and it may cause to LOB updated several time if several market orders are matched at the same price.

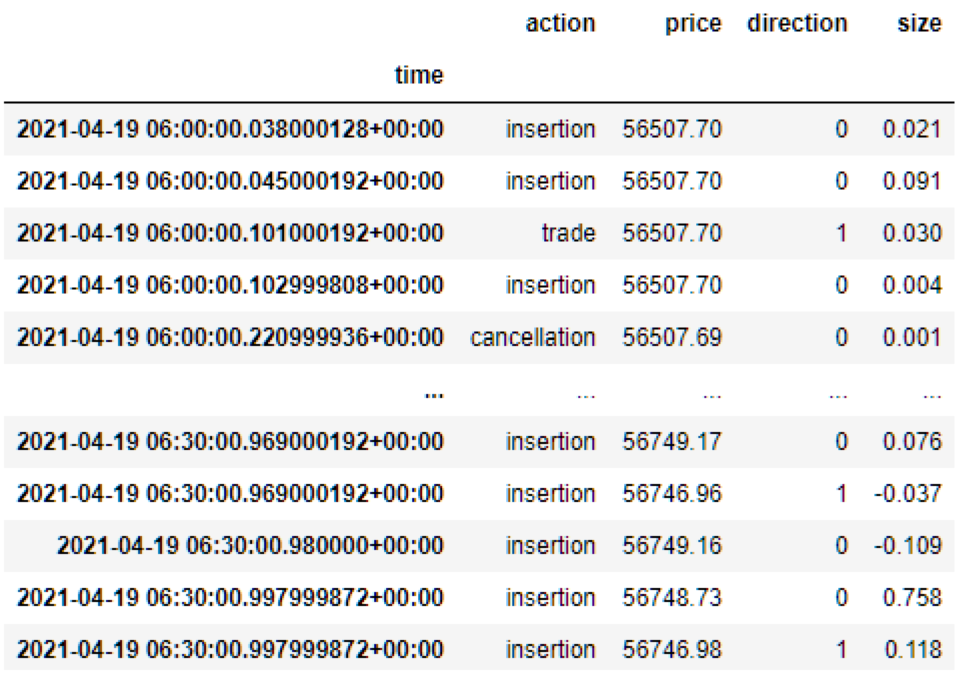
```
prev_trade_cum = df_ask_reduce_temp['trade_cum'].iloc[0]
for ix,row in df_ask_reduce_temp.iterrows():
    x = row['trade_cum']
    if np.isnan(x):
        x = prev_trade_cum + row['ask_size_delta']
    else:
        x = x + row['ask_size_delta']
    x = round(x,8)
    df_ask_reduce_temp.loc[ix, 'trade_cum'] = x
    prev_trade_cum = x
df_ask_reduce_temp['is_trade'] = df_ask_reduce_temp['trade_cum']>-EPS
# ask reduce caused by trade
df_ask_trade = df_ask_reduce_temp[df_ask_reduce_temp['is_trade']]
df_ask_cancel2 = df_ask_reduce_temp[~df_ask_reduce_temp['is_trade']]
df_ask_trade['size'] = df_ask_trade['ask_size_delta'].abs()
# ask redue due to cancellation
df_ask_cancel = pd.concat([df_ask_cancel1, df_ask_cancel2], sort=False).sort_index()
df_ask_cancel['size'] = df_ask_cancel['ask_size_delta'].abs()
df_ask_cancel['action'] = 'cancellation'
```

   ii. Sell order match the price of Bid
      Vice versa as buy order.

Note on step 1.b.i., it uses a looping when handling path dependent conditions, it took 80% of the running time (7.8 seconds in the example run). What it does essentially is:

For each row, compare the order amount with size deduction, if order amount > size deduction, then it is fully trade, and executed amount if the same as size_durection, then the remaining order amount will de deduced pending next match.

Output example as below (use 30 min period of sample data.)



	action	price	direction	size
time				
2021-04-19 06:00:00.038000128+00:00	insertion	56507.70	0	0.021
2021-04-19 06:00:00.045000192+00:00	insertion	56507.70	0	0.091
2021-04-19 06:00:00.101000192+00:00	trade	56507.70	1	0.030
2021-04-19 06:00:00.102999808+00:00	insertion	56507.70	0	0.004
2021-04-19 06:00:00.220999936+00:00	cancellation	56507.69	0	0.001
...	...	...	...	...
2021-04-19 06:30:00.969000192+00:00	insertion	56749.17	0	0.076
2021-04-19 06:30:00.969000192+00:00	insertion	56746.96	1	-0.037
2021-04-19 06:30:00.980000+00:00	insertion	56749.16	0	-0.109
2021-04-19 06:30:00.997999872+00:00	insertion	56748.73	0	0.758
2021-04-19 06:30:00.997999872+00:00	insertion	56746.98	1	0.118

Conclusion

This article describes a algorithm to derive order flow information from Tick and Trade data, using real world imperfect data from Binance Futures. The algorithm can also be implemented on other markets (those missing L3 order flow data) to improve the information completeness. This algorithm fill the gap between theoretical market microstructure theory that derived from order flow data in traditional financial market, and the crypto-currency market, which enable future development on applying existing techniques to crypto-markets.

References:

Abernethy, J., & Kale, S. (2013). Adaptive Market Making via Online Learning. In *Advances in Neural Information Processing Systems* (Vol. 26).

Avellaneda, M., & Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, *8*(3), 217–224. https://doi.org/10.1080/14697680701381228

Briola, A., Turiel, J., & Aste, T. (2020). *Deep learning modelling of the limit order book: a comparative perspective*.

Bouchaud, J.P., Farmer, J.D. and Lillo, F., How markets slowly digest changes in supply and demand. In Handbook of Financial Markets: Dynamics and Evolution, edited by T. Hens and K.R. SchenkHoppé, pp. 57–160, 2009 (Elsevier: Amsterdam)

Chen Yuanda, Backtesting Performance with a Simple Trading Strategy using Market Orders (Dec,2016)

Farmer, J.D. and Lillo, F., On the origin of power-law tails in price fluctuations. Quant. Finance, 2004, 4, 7–11.

Friedman, D., The double auction market institution: A survey. In The Double Auction Market: Institutions, Theories, and Evidence, edited by D. Friedman and J. Rust, 2005 (Addison-Wesley: Reading, MA)Abernethy, J., & Kale, S. (2013). Adaptive Market Making via Online Learning. In *Advances in Neural Information Processing Systems* (Vol. 26).

Avellaneda, M., & Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, *8*(3), 217–224. https://doi.org/10.1080/14697680701381228

Briola, A., Turiel, J., & Aste, T. (2020). *DEEP LEARNING MODELLING OF THE LIMIT ORDER BOOK: A COMPARATIVE PERSPECTIVE*.

Spooner, T., Fearnley, J., Savani, R., & Koukorinis, A. (2018). Market Making via Reinforcement Learning. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, *1*, 434–442. http://arxiv.org/abs/1804.04216

Furlaneto D. C, Oliveira S. Luiz, Menotti D (Dec,2016) Bias effect on prediction market trends with EMD

Hilpisch, Yves. 2018. Python for Finance: Mastering Data-Driven Finance. 2nd ed. Sebastopol: O'Reilly.

Mahfound S., & Mani, G. (1996). Financial Forecasting using genetic algorithms. Applied Artificial Intelligence, 10(6), 543-566

Parlour, C. and Seppi, D.J., Limit order markets: A survey. In Handbook of Financial Intermediation and Banking, edited by A. Thakor and A. Boot, 2008 (Elsevier: Amsterdam).

Gu, G.F., Chen, W. and Zhou, W.X., Empirical regularities of order placement in the Chinese stock market. Phys. A, 2008b, 387, 3173– 3182

Ro¸su, I., A dynamic model of the limit order book. Rev. Financ. Stud., 2009, 22, 4601–4641

Luckock, H., A Statistical Model of a Limit Order Market, 2001 (Sydney University). Available online at: http://www.maths.usyd. edu.au/res/AppMaths/Luc/2001-9.pdf

Abernethy, J., & Kale, S. (2013). Adaptive Market Making via Online Learning. In *Advances in Neural Information Processing Systems* (Vol. 26).

Avellaneda, M., & Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, *8*(3), 217–224. https://doi.org/10.1080/14697680701381228

Briola, A., Turiel, J., & Aste, T. (2020). *DEEP LEARNING MODELLING OF THE LIMIT ORDER BOOK: A COMPARATIVE PERSPECTIVE*.

Spooner, T., Fearnley, J., Savani, R., & Koukorinis, A. (2018). Market Making via Reinforcement Learning. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, *1*, 434–442. http://arxiv.org/abs/1804.04216