In [292…
```
import pandas as pd
import requests
import json
```

In [293…
```
pd.set_option('display.max_colwidth', None)
```

In [294…
```
COW_GRAPH_URL = "https://api.thegraph.com/subgraphs/name/cowprotocol/cow"
COW_AUCTION_API = "https://api.cow.fi/mainnet/api/v1/solver_competition/b

def make_graph_query_request(_query):
    return requests.post(COW_GRAPH_URL, json={'query': _query}).json()

def get_cow_auction(tx_hash):
    return requests.get(COW_AUCTION_API + tx_hash).json()
```

In [295…
```
opps = pd.json_normalize(json.load(open('data/opportunities.json')))
len(opps["cowOrder.id"].unique())
```

Out[295]: 46

In [296…
```
query = """
    {
        orders(where: {
            id_in: [%s]
        }) {
            id
            trades(first: 10) {
                timestamp,
                txHash,
                settlement {
                    solver { address }
                }
                sellToken {
                    decimals
                }
                buyToken {
                    decimals
                }
                sellAmount
                buyAmount
                feeAmount
            }
        }
    }""" % ",".join([f'"{i}"' for i in opps["cowOrder.id"].unique()])

graph_response = make_graph_query_request(query)
```

In [297…
```
settlements = []

for order in graph_response["data"]["orders"]:
    for trade in order["trades"]:
        auction = get_cow_auction(trade["txHash"])
        winning_surplus = 0 if auction.get("solutions") is None else max(

        settlements.append({
            "order_id": order["id"],
            "solver": trade["settlement"]["solver"]["address"],
```

```
            "sell_amount_fix": (int(trade["sellAmount"])-int(trade["feeAm
            "buy_amount_fix": int(trade["buyAmount"]) / 10**trade["buyTok
            "winning_surplus": winning_surplus,
            "tx_hash": trade["txHash"],
        })

settlements_df = pd.DataFrame(settlements)
settlements_df
```

Out[297]:

0	0x0235fdf6659eaa3d6273f32972f866bfba297baf7b67d24851566568e8499ad466e
1	0x05f7ddbdf798e6baa07342983f1e315a498e46a0953ff3c6e5a9c8e1c7bdef550b5c
2	0x0b77e604f4791a37605d2e3b807e3dd0dfbe7d8ac27cdedb88797c4c0480ac2989ec
3	0x0de23176a4cb67a88bbe54674648ee03e1a853e8ca80eef1b1b416284df163a589ec
4	0x10af43fbda2b36a55ea8a96741aa96929e1db7af7a3d012126244069774a269dc52
5	0x116b2a4aac4e82ac7673243a8e1df1d4f922714f8f855c58b99001b6bf0c56d7d361
6	0x172b177104265feb9343668ce76454458c7e1cb09c74fb928f4bf881aebce9d137c
7	0x1799016a31f055527fd67ebf8740bf5bbe2a0d759636c3edb5a5c98961ac61c74
8	0x191732e117d6278a12c249d08868fb6b40bca28d42696d4a42d40b760bc37db9a334
9	0x1b017dd9fe89fb94e1cb9f305c96c2bf3778b802297db6f14404de4aa2836d43f074
10	0x1c7daadce9aeead417835b10b721cdc9b01681d76fc414ec7c93f975c7ec9e2209d44
11	0x1e634981092fe93594f6b9ff852c621449ec73f9153ee418cedee1b99f83f2404e82
12	0x34441958948f206e0e5e2780c4ddd3cdd8aa0f82147dd3d944edbf65457b381471df
13	0x3a73ea64a4cef94ba2d997da2c2b45678fe4546d1a20093e3fc379cb0b74d5fb0b5c
14	0x3f4cb176bd83748591bf8513fdbff95809e59065d71c6e5b914aa94070f2845c4
15	0x410324d3e2babe5990ea3e18f5be78738dfdf8f12f0f38584d2714bab6ed9bdb727
16	0x42b42423bbdf8d76613e22fd85e505b2138e9e4a3d0a5075b11b129110ae15e5e71
17	0x458b177ef2ba7c496e35042461592393919ce69d1e4aefdf20ea5f70ad1764d3d361
18	0x463ae9bb0c8a6643e901fdd136fccc310cc618fd8cf67d94fd7096527ae6c4970d78
19	0x5c1b33929dbd24d765cf4355e56cd021180541089cb0a3843ebaefa8065e1fe5f6f
20	0x5c596ccd7faee963cb17dcaa277aec0a66ba4d0faa9246fdba4f7311435642e34c568
21	0x5f14eb32aa5c1fa247664b94fcea25cef05b22bca0933013ee13c2ada135b1bb1ba6
22	0x63e0ed7abacfeda16f98c257120547ebd168ce13d49401d8575cf1a98a5ad009f8b7d
23	0x68ceec239284c1512cbcfd6e00e8ab18f61599ed65d6c5d12f1434c60b52f8ae4dced
24	0x7d13d5c41459b5ab8928bea1835a777cf095c9449f740f6ec80417a2e0e963ea6e26
25	0x7f66e899d3f64baea03d64ae72c312237f069301cbadd841ab89dab205fcc62276bb
26	0x881cfb08f43b542afe6e9402a3ccf12ea23d36bd63796fcf96ed9935dcc9abdb4
27	0x8e7c26170fae40f6d507434e3cceb683fd8b3c0a5d7a1b50dfd3208ba747b4d022f54
28	0x8fdb8e41c7edaeca6ef3ecc7f6fd1ef24e3f4ee909d97cd07148fd39e6ddcfbdf5e0
29	0x90f03a1e61f7bba104005af0d7a8035804acc3f2959f6b660d7f249a3cb787d64
30	0x92d97c619a7cde3ac0b8a00b76d4b45ef93fd7fec7e232a21ce530f9d41660feaa
31	0x970dcfbcc40b4e347792a9a8e9a71cf0a6c0ea129992a3a0921494975786dfae33da
32	0xa4047c8eb429f92206696b970cde792b04acea099ac44ce57dfbd7991915cf404
33	0xac4550f342e13a88d6dbff240e0447dc6ad3ffb442ae306cb277528e30f203c522c

34	0xb23b4446d66c6f191664bdf997a127d6cd9dbedd5e8565cd6ace3a51ea8e26608626
35	0xb5593cc937a15d6f8f34f0324a8d4f4eefa37917595e8a5aaaa09c181513f7429aef
36	0xc4053191dbcbc6d6d91441e5dc2e312db15a67db23af9308a823a36596d866a9fbe8
37	0xc91a37f9557c691e59a564f640b872bf494b70d267c328eabfa65b021700ff084
38	0xd1cd5b7d8097a846e165c65379ea6eb7c2793f4a939b5b95b1389c124c4f2da189ecb
39	0xd6abfa4b62ee1145dabcfb3752c595d98a1797a67cace9af84d9ec63d341b685090
40	0xee74215969daee5a04c50ef24b15c843e23bf92c32764e44e7a2a66f7b874bc771930
41	0xf5db176140ae2bdcc02277e8365f6e5078ded4d1cbfbc07f1c58970dcd0ae6577604
42	0xf9a48c7d293928963710f2c4d3f430a96f9c57807090b36132a7595cdedd5c904

Positive spread

In [298…

```
def get_duration_sec(series):
    return int((series.max()-series.min()) / 1e3)



opps_with_settlements = opps.merge(settlements_df, left_on="cowOrder.id",
prices = opps_with_settlements['binanceOrderSpreadEth']/(opps_with_settle
opps_with_settlements["spread_binance_settlement"] = ((opps_with_settleme
opps_with_settlements["spread_binance_order"] = opps_with_settlements["bi
opps_with_settlements["spread_binance_oneinch"] = opps_with_settlements["
opps_with_settlements["winning_surplus"] /= 1e18

# groupby order-id and agg: max spread, min spread, count,
agg_opps = opps_with_settlements \
    .groupby(by="cowOrder.id") \
    .agg(
        spread_binance_settlement_max=("spread_binance_settlement", max),
        spread_binance_settlement_min=("spread_binance_settlement", min),
        spread_binance_order_max=("spread_binance_order", max),
        spread_binance_order_min=("spread_binance_order", min),
        spread_binance_oneinch_max=("spread_binance_oneinch", max),
        spread_binance_oneinch_min=("spread_binance_oneinch", min),
        duration_sec=("timestamp_ms", get_duration_sec),
        winning_surplus=("winning_surplus", max),
    ) \
    .reset_index() \
    .sort_values(by="spread_binance_settlement_max", ascending=False)
agg_opps["have_winning_surplus"] = agg_opps["spread_binance_order_max"] >

agg_opps
```

Out[298]:

36	0xc4053191dbcbc6d6d91441e5dc2e312db15a67db23af9308a823a36596d866a9fbe8
19	0x5c1b33929dbd24d765cf4355e56cd021180541089cb0a3843ebaefa8065e1fe5f6
24	0x7d13d5c41459b5ab8928bea1835a777cf095c9449f740f6ec80417a2e0e963ea6e26
26	0x881cfb08f43b542afe6e9402a3ccf12ea23d36bd63796fcf96ed9935dcc9abdb4
0	0x0235fdf6659eaa3d6273f32972f866bfba297baf7b67d24851566568e8499ad466e
39	0xd6abfa4b62ee1145dabcfb3752c595d98a1797a67cace9af84d9ec63d341b685090
10	0x1c7daadce9aeead417835b10b721cdc9b01681d76fc414ec7c93f975c7ec9e2209d44
27	0x8e7c26170fae40f6d507434e3cceb683fd8b3c0a5d7a1b50dfd3208ba747b4d022f54
34	0xb23b4446d66c6f191664bdf997a127d6cd9dbedd5e8565cd6ace3a51ea8e2660862
13	0x3a73ea64a4cef94ba2d997da2c2b45678fe4546d1a20093e3fc379cb0b74d5fb0b5c
30	0x92d97c619a7cde3ac0b8a00b76d4b45ef93fd7fec7e232a21ce530f9d41660feaa
28	0x8fdb8e41c7edaeca6ef3ecc7f6fd1ef24e3f4ee909d97cd07148fd39e6ddcfbdf5e0
35	0xb5593cc937a15d6f8f34f0324a8d4f4eefa37917595e8a5aaaa09c181513f7429aef
41	0xf5db176140ae2bdcc02277e8365f6e5078ded4d1cbfbc07f1c58970dcd0ae6577604
37	0xc91a37f9557c691e59a564f640b872bf494b70d267c328eabfa65b021700ff084
23	0x68ceec239284c1512cbcfd6e00e8ab18f61599ed65d6c5d12f1434c60b52f8ae4dced
40	0xee74215969daee5a04c50ef24b15c843e23bf92c32764e44e7a2a66f7b874bc771930
21	0x5f14eb32aa5c1fa247664b94fcea25cef05b22bca0933013ee13c2ada135b1bb1ba6
3	0x0de23176a4cb67a88bbe54674648ee03e1a853e8ca80eef1b1b416284df163a589ec
38	0xd1cd5b7d8097a846e165c65379ea6eb7c2793f4a939b5b95b1389c124c4f2da189ecb
14	0x3f4cb176bd83748591bf8513fdbff95809e59065d71c6e5b914aa94070f2845c4
4	0x10af43fbda2b36a55ea8a96741aa96929e1db7af7a3d012126244069774a269dc52
18	0x463ae9bb0c8a6643e901fdd136fccc310cc618fd8cf67d94fd7096527ae6c4970d78
6	0x172b177104265feb9343668ce76454458c7e1cb09c74fb928f4bf881aebce9d137c
16	0x42b42423bbdf8d76613e22fd85e505b2138e9e4a3d0a5075b11b129110ae15e5e71
2	0x0b77e604f4791a37605d2e3b807e3dd0dfbe7d8ac27cdedb88797c4c0480ac2989ec
11	0x1e634981092fe93594f6b9ff852c621449ec73f9153ee418cedee1b99f83f2404e82
20	0x5c596ccd7faee963cb17dcaa277aec0a66ba4d0faa9246fdba4f7311435642e34c568
22	0x63e0ed7abacfeda16f98c257120547ebd168ce13d49401d8575cf1a98a5ad009f8b7d
5	0x116b2a4aac4e82ac7673243a8e1df1d4f922714f8f855c58b99001b6bf0c56d7d361
33	0xac4550f342e13a88d6dbff240e0447dc6ad3ffb442ae306cb277528e30f203c522c
7	0x1799016a31f055527fd67ebf8740bf5bbe2a0d759636c3edb5a5c98961ac61c74
1	0x05f7ddbdf798e6baa07342983f1e315a498e46a0953ff3c6e5a9c8e1c7bdef550b5c
25	0x7f66e899d3f64baea03d64ae72c312237f069301cbadd841ab89dab205fcc62276bb

31	0x970dcfbcc40b4e347792a9a8e9a71cf0a6c0ea129992a3a0921494975786dfae33da
17	0x458b177ef2ba7c496e35042461592393919ce69d1e4aefdf20ea5f70ad1764d3d361
29	0x90f03a1e61f7bba104005af0d7a8035804acc3f2959f6b660d7f249a3cb787d64
9	0x1b017dd9fe89fb94e1cb9f305c96c2bf3778b802297db6f14404de4aa2836d43f074
15	0x410324d3e2babe5990ea3e18f5be78738dfdf8f12f0f38584d2714bab6ed9bdb727
12	0x34441958948f206e0e5e2780c4ddd3cdd8aa0f82147dd3d944edbf65457b381471df
8	0x191732e117d6278a12c249d08868fb6b40bca28d42696d4a42d40b760bc37db9a334
32	0xa4047c8eb429f92206696b970cde792b04acea099ac44ce57dfbd7991915cf404
42	0xf9a48c7d293928963710f2c4d3f430a96f9c57807090b36132a7595cdedd5c904

In [299…
```
agg_opps.loc[(agg_opps.spread_binance_settlement_max > 0) & (agg_opps.hav
```

Out[299]: 0.12842942044799585

Simulation

In [300…
```
from collections import defaultdict


best_opps_for_order = opps_with_settlements.loc[opps_with_settlements.gro
best_opps_for_order.loc[best_opps_for_order.tokenIn == "0xeeeeeeeeeeeeeee
best_opps_for_order.loc[best_opps_for_order.tokenOut == "0xeeeeeeeeeeeeeee


starting_balance = 1e6
binance_balance = defaultdict(lambda: starting_balance)
onchain_balance = defaultdict(lambda: starting_balance)

for (i, order) in best_opps_for_order.sort_values(by="timestamp_ms").iter
    binance_amount_in = order["binanceMatch.amountIn"]
    binance_amount_out = order["binanceMatch.amountOut"]
    token_in = order["tokenIn"]
    token_out = order["tokenOut"]

    if binance_balance[token_in] < binance_amount_in:
        print(f"Insufficient balance for {token_in} on binance")
        continue

    binance_balance[token_in] -= binance_amount_in
    binance_balance[token_out] += binance_amount_out

    if order.winning_surplus > order.spread_binance_order:
        # print(f"Order {order['cowOrder.id']} would lose auction")
        continue

    if onchain_balance[token_out] < binance_amount_out:
        print(f"Insufficient balance for {token_out} on chain")
        continue

    onchain_balance[token_out] -= binance_amount_out  # ! suppose we give
```

```
    onchain_balance[token_in] += binance_amount_in


print("Binance balance deltas")
for token, balance in binance_balance.items():
    print(f"\t{token}: {balance-starting_balance}")
print("Onchain balance deltas")
for token, balance in onchain_balance.items():
    print(f"\t{token}: {balance-starting_balance}")
```

```
Binance balance deltas
        0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48: 151905.96355938143
        0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2: -55.84863445512019
        0x6b175474e89094c44da98b954eedeac495271d0f: -30793.245285494835
        0xdac17f958d2ee523a2206206994597c13d831ec7: -936.3882890000241
        0x2260fac5e5542a773aa44fbcfedf7c193bc2c599: -0.11844384472351521
Onchain balance deltas
        0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2: 100.89603361603804
        0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48: -235995.90388179326
        0x6b175474e89094c44da98b954eedeac495271d0f: 25223.333819224034
```