

# RFAB v2 기획서

*RFAB = Risk Free Arbitrage Bot*

## 1. RFAB v1 한계점

- 제1호가의 제한적 잔량 경향으로 인한 수익한계
- Unit spread 이용 시 최대 수익 기회 제한
- 다양한 호가 및 그에 상응한 잔량 이용 기회 전무
- 한 조합으로 거래 시 자본효과 감소

## 2. RFAB v2 전략

- Optimized Traded Spread (OTS)
- Individual Combination Optimization (ICO)
- 3D Portfolio Optimization (3D)

## 3. 백테스팅 알고리즘

- 기존 벡터테스팅 문제점 + 해결안

**Team Gazeua**

# 1. RFAB v1 한계점

## ○ 제1호가의 제한적 잔량 경향으로 인한 수익한계

	6.265	1,655,000	
	1.3	1,654,500	
	1.4835	1,651,500	
제결가	거래량	1,650,000	0.551
1,652,500	5.3000	1,645,500	0.4345
1,652,500	9.0900	1,645,000	6.2021
1,650,000	2.5508		

- ma[0] 혹은 mb[0]에 걸려있는 잔량이 trading unit보다 작은 경우 스프레드 > 0 임에도 불구하고 trading 못할 수도
  - 그림처럼 제1호가에는 잔량이 적은 경향이 있음
    - 아무래도 빠르게 소진되고 가능한 1호가 위에 maker주문 거는 경향일수도
    - 어찌됐든 이유는 중요하지 않음
  - 스프레드가 0보다 큰 (수익이 나는 경우) 구간도 많지 않은데 그 수익 구간에서 매수, 매도 1호가 잔량 중 minimum값이 굉장히 작은 경우가 많다는 사실을 백테스팅으로 발견
    - mm1과 mm2 간 코인 거래 수는 같아야 하므로 매도, 매수 1호가 중 최소치를 택해야 함
- ➔ 스프레드 > 0 인데 먹을 수 있는 잔량이 별로 없으면 그만큼 수익기회는 줄어듦**

# 1. RFAB v1 한계점

## ○ Unit Spread 이용시 최대 수익기회 제한

```
if new_spread_in_unit > self.NEW_SPREAD_THRESHOLD and new_trading_amount >= self.MIN_COIN_TRADING_UNIT:
```

- RFAB v1에서는 이런식으로 수익기회 판단 로직에 1코인 거래 가정시 스프레드 > 0 을 사용
- 그러나 이는 1코인 거래할 시 얻을 수 있는 수익금이고, 실제로는 이보다 훨씬 적은 잔량으로 거래할 수 밖에 없음

```
Spread in unit = 1079.83, BUY_index = 0, SELL_index = 0, Traded Spread = 0.13, Traded QTY = 0.00012
Spread in unit = 29558.46, BUY_index = 0, SELL_index = 0, Traded Spread = 32.51, Traded QTY = 0.00110
Spread in unit = 1079.83, BUY_index = 0, SELL_index = 0, Traded Spread = 0.13, Traded QTY = 0.00012
Spread in unit = 1582.96, BUY_index = 0, SELL_index = 1, Traded Spread = 15.83, Traded QTY = 0.01000
Spread in unit = 1083.33, BUY_index = 0, SELL_index = 1, Traded Spread = 10.83, Traded QTY = 0.01000
Spread in unit = 1083.33, BUY_index = 0, SELL_index = 0, Traded Spread = 10.83, Traded QTY = 0.01000
Spread in unit = 1083.33, BUY_index = 0, SELL_index = 0, Traded Spread = 0.47, Traded QTY = 0.00013
Spread in unit = 8576.83, BUY_index = 0, SELL_index = 0, Traded Spread = 85.77, Traded QTY = 0.01000
Spread in unit = 8576.83, BUY_index = 0, SELL_index = 0, Traded Spread = 85.77, Traded QTY = 0.01000
Spread in unit = 8576.83, BUY_index = 0, SELL_index = 0, Traded Spread = 85.77, Traded QTY = 0.01000
Spread in unit = 3071.33, BUY_index = 0, SELL_index = 0, Traded Spread = 30.71, Traded QTY = 0.01000
Spread in unit = 4072.33, BUY_index = 0, SELL_index = 0, Traded Spread = 40.72, Traded QTY = 0.01000
Spread in unit = 4572.83, BUY_index = 0, SELL_index = 0, Traded Spread = 45.73, Traded QTY = 0.01000
Spread in unit = 5073.33, BUY_index = 0, SELL_index = 0, Traded Spread = 50.73, Traded QTY = 0.01000
Spread in unit = 9577.83, BUY_index = 0, SELL_index = 0, Traded Spread = 95.78, Traded QTY = 0.01000
```

- **빨간색** : 기존 RFAB v1에서는 스프레드가 29558.46으로 굉장히 크지만, 로직에 의해 거래된 QTY는 0.00110으로 작음
- **파란색** : Unit spread는 작지만, 잔량이 많아 Traded spread는 빨간색보다 큼.

- 결론적으로 Unit spread로 New Threshold나 기타 여러 initial setting하는게 아닌 Traded Spread를 가지고 판단해야함
- Traded Spread는 Mkt 잔량이 변수로 들어간다.  $\text{Spread} = (P1 - P2) * Qty$  에서 S를 최대화하는 적정 P와 Q를 찾아야함.

# 1. RFAB v1 한계점

## ○ 다양한 호가 및 그에 상응한 잔량 이용 기회 전무

	0.6618	<b>1,683,500</b>	현재가	1,650,000
	0.2988	<b>1,683,000</b>	전일가	1,780,000
	0.1	<b>1,673,500</b>	전일대비	-130,000 / -7.30%
	0.1	<b>1,672,500</b>	고가	1,780,000
	62	<b>1,668,000</b>	저가	1,649,000
	0.074	<b>1,664,000</b>	거래량	5,850
	0.6564	<b>1,656,000</b>		
	6	<b>1,655,500</b>		
	10.39	<b>1,653,000</b>		
체결가	거래량	<b>1,650,000</b>	0.551	
1,650,000	0.6810	<b>1,645,000</b>	9.508	
1,649,000	10.0000	<b>1,641,000</b>	1	
1,650,000	4.4772	<b>1,640,000</b>	0.5303	
1,649,000	0.1580	<b>1,638,000</b>	7.9505	
1,650,000	6.8893	<b>1,637,500</b>	1.1044	
1,649,500	0.1580	<b>1,635,500</b>	7	
1,650,000	1.0940	<b>1,635,000</b>	0.0303	
1,649,500	0.6486	<b>1,634,000</b>	0.0096	
1,650,000	40.2978			
1,651,000	0.1160			
80.25	표시잔량합계	41.21		

- 마찬가지로 제1호가만 이용해 Spread를 만들면 다른 호가들 \* 해당 잔량으로 계산되는 스프레드까지 비교하는 것보다 그 기회 수준이 적을 것.
- 지금까지 살펴본 한계점들을 고려해 나머지 여러 호가들로 Traded Spread를 구성해 그 중 Initial Setting에서 설정한 조건들과 비교해 부합하는 Spread 중 가장 큰 것을 이용하면 수익 조금 더 개선 가능.

# 1. RFAB v1 한계점

## ○ 한 조합으로 거래 시 자본효과 감소

- 현재까지 코인원-코빗-이더리움의 조합 (Individual Combination 라 명명하겠음) 으로 봇을 개발하였는데, 이 조합 내 거래소들이 제공하는 유동성은 한계가 있음.
- 물론 전체 유동성이 아니라, 스프레드 > 0 인 수익구간에서 두 거래소가 제공하는 미체결잔량은 제한적이다.
- 이로써, 이 IC (Individual Combination) 에서 천만원 투자시 하루 2%의 괄목할 만한 수익을 냈다하더라도, 같은 IC에 1억을 투자했을 때 같은 2% 수익이 날 수 없음
- 그 이유는 다음 페이지에서

$$\text{하루 수익률} = \frac{\text{하루 현금 창출액}}{\text{투자금} \& \text{initial setting}}$$

# 1. RFAB v1 한계점

- 한 조합으로 거래 시 자본효과 감소

$$\text{하루 수익률} = \frac{\text{하루 현금 창출액}}{\text{투자금} \& \text{initial setting}}$$

- 하루 현금 창출액을 maximize하는 방법은 스프레드가 아무리 작더라도 거래를 하고, 스프레드  $> 0$  인 지점에서 내가 거래할 수 있는 Qty가 아무리 크더라도 무조건 다 거래하는 방법일 것이다.
- 그러나 그러면 투자금액이 너무 많이 들어간다. Initial Setting에서 mm1 현금, 코인수 / mm2 현금, 코인수 / Maximum Trading Unit 무한정 늘리면 당연히 모든 스프레드  $> 0$  인 것과 그 잔량을 다 먹어서 하루 현금 창출액은 최대치를 찍겠지만, 그만큼 분모의 투자금이 커지므로 수익률은 최대가 아닐 수 있음
- 따라서, 하루 수익률을 Maximize하는 적정 투자금 및 Initial Setting, 그리고 이로부터 파생되는 하루 현금 창출액이 있을 것이다.
  - 최소의 투자금으로 가능한 많은 현금을 창출해 수익률 극대화 시켜야함.

## 2. RFAB v2 전략

### ○ Optimized Traded Spread (OTS)

```
# init virtual mm when backtesting
v_mm1 = VirtualMarketManager(Market.VIRTUAL_CO, 0.001, 3000000, 0.2, target_currency)
v_mm2 = VirtualMarketManager(Market.VIRTUAL_GP, 0.00075, 300000, 2, target_currency)

super().__init__(v_mm1, v_mm2, target_currency, target_interval_in_sec, should_db_logging,
is_backtesting, start_time, end_time)

self.MAX_COIN_TRADING_UNIT = 0.01
self.MIN_COIN_TRADING_UNIT = 0.0001
self.MAX_OB_INDEX_NUM = 3
self.NEW_SPREAD_THRESHOLD = 60
self.REV_SPREAD_THRESHOLD = 0
self.REV_FACTOR = 3
```

```
@staticmethod
def get_optimized_spread_infos(buy_dict: dict, buy_fee: float,
                                sell_dict: dict, sell_fee: float,
                                max_trading_unit: float, ob_index_num: int):

    spread_set = list()
    for i in range(0, ob_index_num):
        buy_price = int(buy_dict["price"][i].to_decimal())
        for k in range(0, ob_index_num):
            sell_price = int(sell_dict["price"][k].to_decimal())

            # since we have to trade the same amount at buy and sell side, as well as complying with each market
            # amounts and max_trading_unit
            buy_dict_amount = float(buy_dict["amount"][i].to_decimal())
            sell_dict_amount = float(sell_dict["amount"][i][k].to_decimal())
            possible_trading_qty = float(min(buy_dict_amount * (1 - buy_fee), sell_dict_amount, max_trading_unit))

            # actual spread and append to spread list
            spread_in_unit = (-1) * buy_price / (1 - buy_fee) + (+1) * sell_price * (1 - sell_fee)
            spread_for_trade = spread_in_unit * possible_trading_qty
            spread_set.append((spread_for_trade, i, k, possible_trading_qty, spread_in_unit))

    # get the maximized pair for trading
    (opt_spread, opt_buy_index,
     opt_sell_index, opt_trading_qty, spread_in_unit) = Analyzer.get_max_pair_infos(spread_set)

    opt_buy_price = int(buy_dict["price"][opt_buy_index].to_decimal())
    opt_sell_price = int(sell_dict["price"][opt_sell_index].to_decimal())

    return (opt_spread, opt_buy_price, opt_buy_index,
            opt_sell_price, opt_sell_index, opt_trading_qty, spread_in_unit)
```

### 1. Initial Setting에서

- **Max\_Coin\_Trading\_Unit** : 1회 거래시 거래 가능 최대 Coin수
- **Min\_Coin\_Trading\_Unit** : 1회 거래시 거래 가능 최소 Coin수
- **Max\_OB\_INDEX\_Num** : 분석할 호가 index num 설정
- **Spread\_Threshold** : 스프레드가 얼마나 커야 trading execution할 지 결정

### 2. Get\_Optimized\_Spread\_Info

- ma[0] ~ ma[n] 의 price와 qty
- mb[0] ~ mb[n] 의 price와 qty 의 조합 중 optimization
- 즉,  $P_{ma[n1]} \cdot Q_{ma[n1]}$  과  $P_{mb[n2]} \cdot Q_{mb[n2]}$ 의 스프레드 중 maximum
- n1 and n2  $\leq$  Max\_OB\_index\_num
- 거래 Q =  $\min(Q_{ma[n1]}, Q_{mb[n2]}, min\_coin\_trading\_unit)$

➤ 엄밀한 로직은 코드 보시길.

➤ 미리 설정한 호가 index의 가격과 잔량의 조합 중 optimized된 놈 찾는것

## 2. RFAB v2 전략

### ○ Individual Combination Optimization (ICO)

➤ IC는 mm1-mm2-coin와 같이 1대1 아비트리지 조합을 뜻함.

앞서 보았듯이

$$\text{하루 수익률} = \frac{\text{하루 현금 창출액}}{\text{투자금} \& \text{initial setting}}$$

이며, 한 IC에 대해서 자본효과는

감소할 수 밖에 없는 구조이다.

Case1) 현금 33만원, 코인 수 2.2

```
.VIRTUAL_CO, 0.001, 3000000, 0.2, target_currency)  
.VIRTUAL_GP, 0.00075, 300000, 2, target_currency)
```

```
, balance - 2.2000  
.0000, balance - 3329684.3387
```

수익률 = 8.9%

Case2) 현금 330만원, 코인 수 22

```
.VIRTUAL_CO, 0.001, 30000000, 2, target_currency)  
.VIRTUAL_GP, 0.00075, 3000000, 20, target_currency)
```

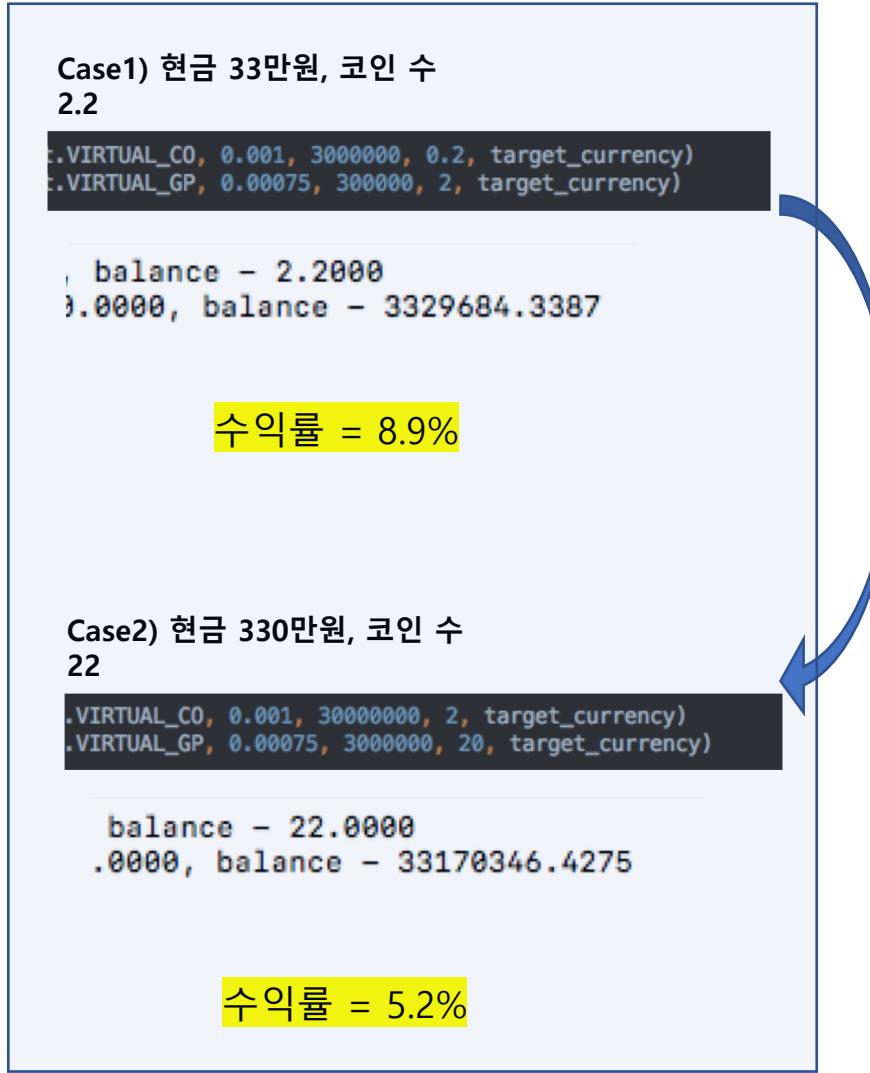
```
, balance - 22.0000  
.0000, balance - 33170346.4275
```

수익률 = 5.2%

- 수익률 계산 과정에서 코인 가치는 들어가지 않음 (그러나 현금 비율 만큼 코인 비율도 동일 적용 – 10배)
- 해당 알고리즘은 앞에 설명한 OTS를 기준으로 설정.
- 추후 설명하겠지만, 현재 보유하고 있는 백테스팅 로직은 수익률 과대 평가함 → 수익률은 둘다 더 낮을 것임.

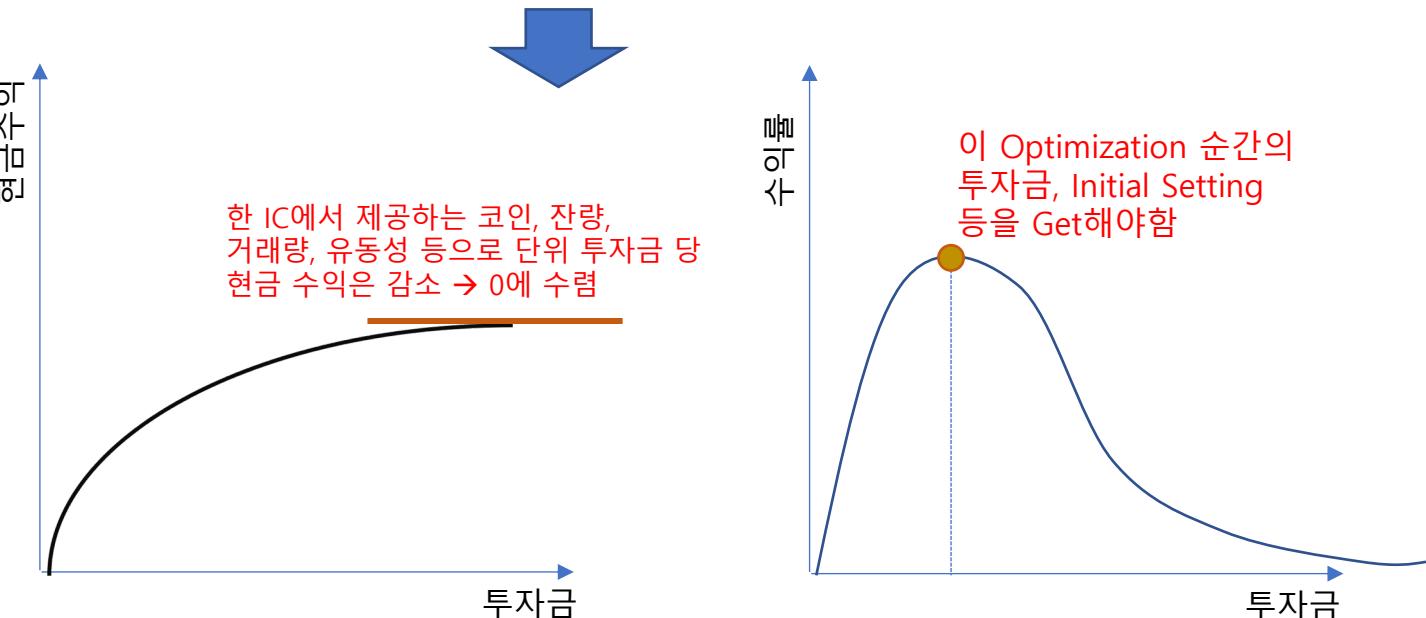
## 2. RFAB v2 전략

### ○ Individual Combination Optimization (ICO)



→ 투자금은 10배 → 수익률은 감소  
→ 자본효과 증가하다 일정수준 투자금 증가하면 0

Investment Decision Behavior는 연속함수 → Optimization 존재함



## 2. RFAB v2 전략

### ○ Individual Combination Optimization (ICO)

#### Optimization을 하기 위해 고려되어야 할 변수들

##### □ Threshold 수치

- ✓ 알고리즘에서 new\_spread > threshold 로 하는 것보다 traded\_new\_spread > threshold 으로 설정하는것이 수익률 개선에 큰 도움 줌
- ✓ 백테스팅 결과 30 ~ 70% 개선 가능
- ✓ 엄밀히 생각하면 기존 알고리즘은 실제로 거래되는 스프레드를 고려하는게 아닌 단순 unit spread를 threshold하는거라.. 수익기회 많이 놓침

##### □ Max\_trading\_unit, Min\_trading\_unit 수치

- ✓ Initial Setting시 해당 변수 설정을 바꾸다 보면 수익률 극대화 지점 나옴
- ✓ 또한, 봇 run시작할때 고정변수로 들어가기 보단, 시장의 변화에 적절히 대응하며 유동적으로 변하면 좋을듯. → 모니터링 알고리즘의 필요성 도래

##### □ Mm1, mm2 balance 설정 및 현재 balance 수치 + 정산까지 남은 시간

- ✓ 현재 밸런스가 얼마나 많냐, 적냐에 따라 / 정산시간까지 남은 시간이 많냐 적냐에 따라 현재 스프레드에 크게 베팅하냐 마느냐를 적절히 조절해야만 함. 이를 확률 함수로 계산하여 opt하는게 논리적이라봄.
- ✓ 실제로 정산에 가까워질수록 Not Enough Balance로 장 초반보다 스프레드 커졌음에도 불구하고 못먹는경우가 있음. 스프레드 분포와 발생확률 및 밸런스, 정산까지 남은 시간등을 변수로..

➤ 최종 목표 : 해당 변수 input으로 넣으면 output으로 setting값 및 밸런스, 투자금액 등등 return

## 2. RFAB v2 전략

### ○ 3D Portfolio Optimization (3D)

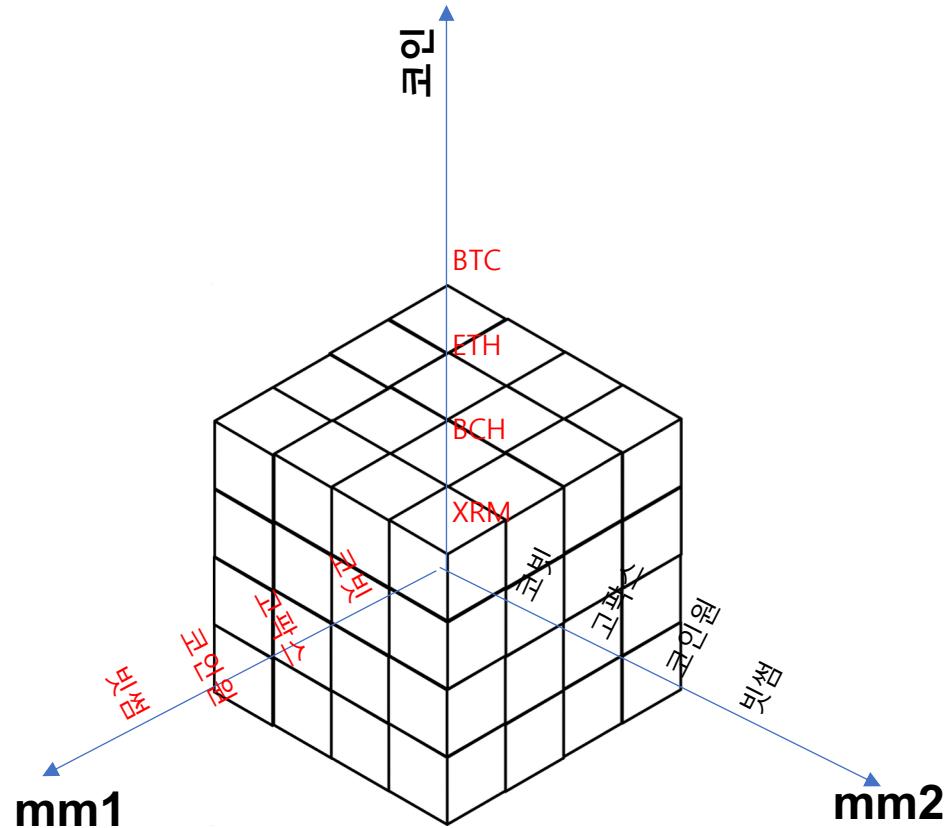
- 앞서 본 ICO 전략으로 return된 값들로 봇은 개별 IC를 optimization 할 수 있는 최적 setting 값을 판단함.
- 따라서, 이에 적용되는 IC를 엄청 늘려서 실시간 모니터링하여 opt된 setting값으로 매번 바꿔주고, 정산시간에는 그 다음날 거래를 위해 자산을 분배하고 배치하는 역할을 하는 알고리즘이 필요함.

- 현재 총자본금을 RFAB v2로 굴렸을때 여러 조합가능한 IC에 어떻게 분배해야하는가에 대한 포트폴리오 분배 문제임.
- 이를 3D 행렬 계산 알고리즘으로 실시간 모니터링하고, input으로 총투자금을 넣었을때, 어느 IC에 얼마만큼의 coin과 현금을 배분할지 계산

- ➔ 투자금이 얼마 없다면 모르겠지만, 큰 돈을 굴리게 되면, 이는 필수적인 요소
- ➔ 다행히 국내에 Big 거래소는 4개 이상, 코인도 10개 이상
  - ➔  $4C2 * 10 = 120$ 개의 IC 조합가능
  - ➔ 해외 계좌도 뚫어서 아비트리지 가능하다면 더 많은 조합가능

## 2. RFAB v2 전략

### ○ 3D Portfolio Optimization (3D)



이런식으로 mm1, mm2, 코인으로 x,y,z 축으로 삼고 큐브처럼 만들어서

→ 큐브 안 작은 큐브는 ICO 알고리즘 및 Portfolio Optimization (PO) 가 내장

#### 1. 정산시간

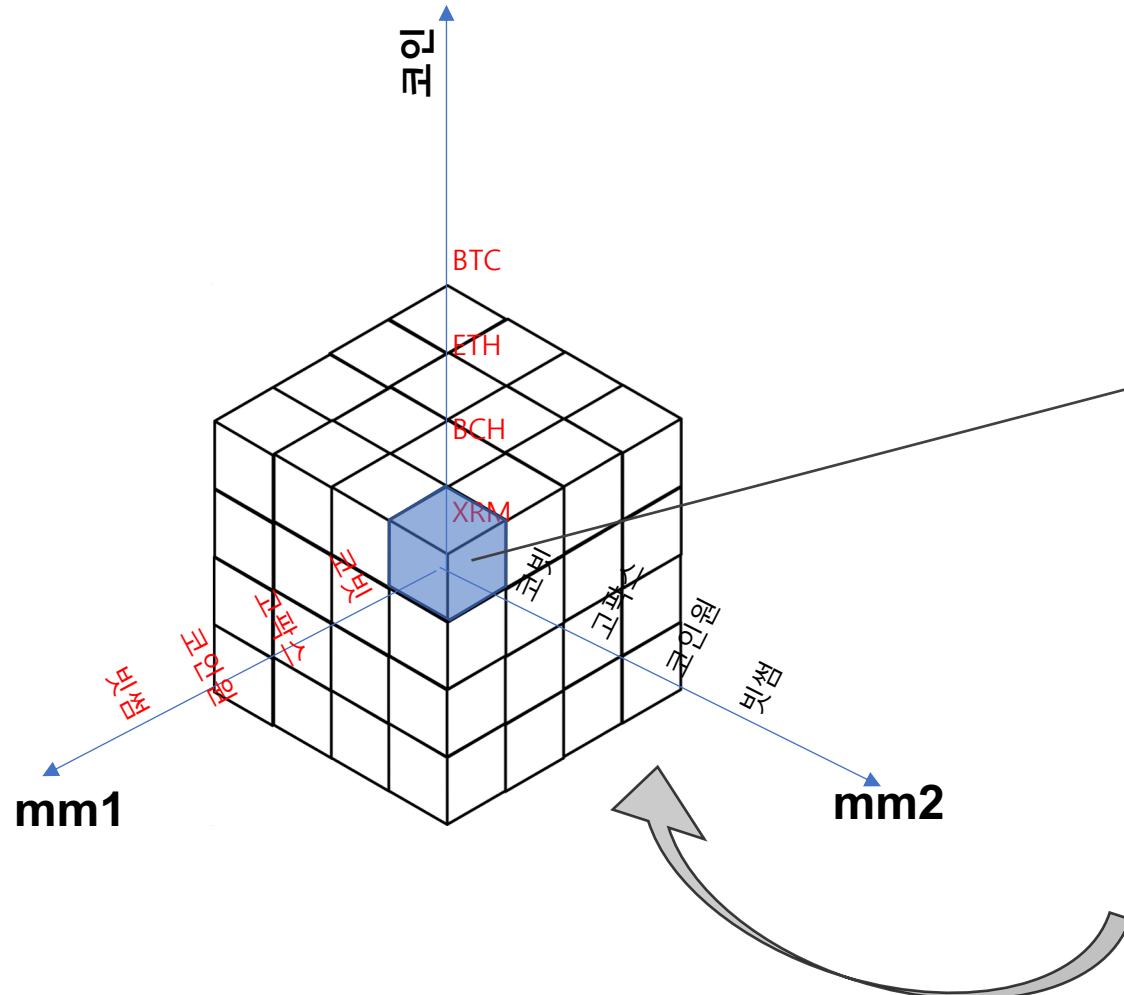
- 정산시간에는 PO 가 입력된 총 투자금으로 어느 작은큐브에 얼마만큼의 자본을 투자, initial setting은 어떻게 할지 결정.  
(ICO 전략 활용해야함)
  - 결국 return 값으로 투자할 IC량 각각의 투자금, initial setting 설정

#### 2. 장 중

- 장 중에는 각각의 작은 큐브는 실시간 ICO 알고리즘으로 현재 스프레드, 과거 평균, 통계적 수치, balance 등등을 분석하여 최적의 trading unit 및 투자 결정을 내림

## 2. RFAB v2 전략

### ○ Summary



If 정산:

스프레드 평균, oppy 개수, qty평균 및 확률분포, 표편 등등

elif 장중:

과거 스프레드 확률분포, qty, 현재 balance, 정산까지 남은 시간, 누적 oppy 수, 현재 스프레드, 표편 등

Input

ICO 알고리즘

If 정산:

최적 투자금액, 예상 수익률, initial setting

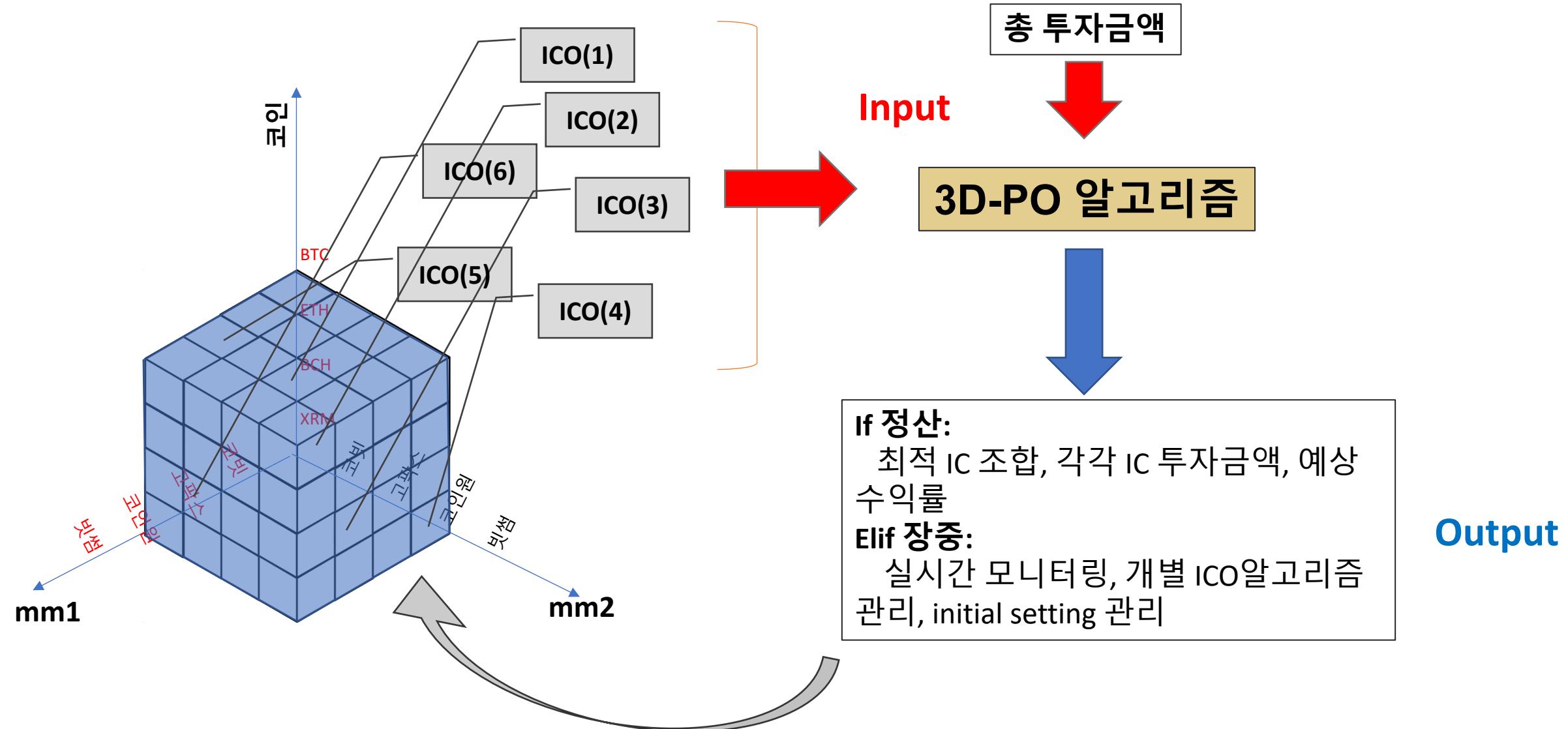
elif 장중:

실시간 최적 initial setting 등

Output

## 2. RFAB v2 전략

### ○ Summary



### 3. 백테스팅 알고리즘 제안

#### ○ 기존 백테스팅 문제점 + 해결안

```
INFOS: Spread in unit = 12301.55, BUY_index = 0, SELL_index = 0, Traded Spread = 123.02, Traded QTY = 0.01000
INFOS: Spread in unit = 11801.93, BUY_index = 0, SELL_index = 0, Traded Spread = 118.02, Traded QTY = 0.01000
INFOS: Spread in unit = 11801.93, BUY_index = 0, SELL_index = 0, Traded Spread = 118.02, Traded QTY = 0.01000
INFOS: Spread in unit = 11801.93, BUY_index = 0, SELL_index = 0, Traded Spread = 118.02, Traded QTY = 0.01000
INFOS: Spread in unit = 11801.93, BUY_index = 0, SELL_index = 0, Traded Spread = 118.02, Traded QTY = 0.01000
INFOS: Spread in unit = 11801.93, BUY_index = 0, SELL_index = 0, Traded Spread = 118.02, Traded QTY = 0.01000
INFOS: Spread in unit = 11801.93, BUY_index = 0, SELL_index = 0, Traded Spread = 118.02, Traded QTY = 0.01000
INFOS: Spread in unit = 11801.93, BUY_index = 0, SELL_index = 0, Traded Spread = 118.02, Traded QTY = 0.01000
INFOS: Spread in unit = 12301.55, BUY_index = 0, SELL_index = 0, Traded Spread = 123.02, Traded QTY = 0.01000
```

- ▶ 현재 백테스팅 알고리즘은 같은 스프레드가 일정 시간 변동없이 유지될때 해당 스프레드 기회 모두를 개별 수익기회로 인식하고 execute하여 누적시킴.

#### A. 만약 위의 예처럼 11801.93 스프레드에 거래가능 QTY가 0.034... 이라 가정하면

- ▶ 처음 3번까지만 거래가 가능할것임. 시장이 그보다 많은 잔량을 제공하지 않기에...
- ▶ 따라서, 수익은  $118.02 * 6$ 번 이 아닌  $118.02 * 3$ 번임

#### B. 또한, 이렇게 수정했다해도 엄밀히 따져봤을땐 현실과 동떨어진 요소가 있긴함.

- ▶ 다시 위의 예처럼 수익기회가 나서 거래를 하면 시장이 변동하여 잔량, 가격, 제1호가 등이 모두 바뀔수있음.
- ▶ BUT, 이런것까지 고려할 이유는 없어보이고 백테스팅이므로 어느정도 현실성 있는 예상 수익 및 거래 시점 + 정보만 보여주면 충분함.

'A 안' 으로 수정하면  
좀더 수익률 신뢰도  
높일 수 있을것임.