Index Weighting Practice

Naixian Carucci

March 10, 2022

Abstract

Weighting rule is not an insignificant component in index creation. Some indexes have complex weighting schemes. Practitioners often use embedded Excel function or programming to implement. This paper intends to summarize what we practice in applying index weighting rules and discuss additional mathematical optimization approaches not only to fulfill the task efficiently but also to guide the designing of weighting rules.

1 Introduction

In the index-backed passive investment world, there are four main weighting schemes in constructing indexes: price weighted, (float adjusted) market capitalization weighted, equal weighted and fundamental weighted.

Price weighted indexes are rare even the earliest and famous Dow Jones Industrial Average (DJIA) belongs to this kind. It is composed of 30 prominent companies listed on stock exchanges in the United States. An obvious flaw is that when stock undergoes price change such as split or reverse split, the weighting is greatly affected without legitimate foundation.

A capitalization-weighted (or cap-weighted) index, also called a market-value-weighted index is a stock market index whose components are weighted according to the total market value of their outstanding shares. Since oftentimes public company stock shares are held by the government, royalty, or company insiders (privately held), not available for the public to trade on, their corresponding stake percentage should be removed from true-market-value calculation, thus this "free float adjusted market capitalization weighting" is largely adopted in indexing industry now. Because this market capitalization based represents the "efficient frontier" in the classical market efficiency theory and CAPM. This approach is the most prevalent in both research and practice.

However, this approach has its intrinsic flaws too. Robert Arnott, Jason Hsu and Philip Moore stated their concern [1] that "market capitalization is a particularly volatile way to measure a company's size or true fair value." To put it in a plainer logic, this weighting method gives additional weight to those overpriced or simply while reduce weight to those undervalued small companies, exacerbate the mis-pricing and hence digresses from an optimal portfolio.

To address this deficiency, equal weighting is an alternative which is largely applied in real market too. It gives each component security same weight regardless of any metrics. The dilemma however is that good stocks are continuously punished by reducing its weight while under-performers are irrationally rewarded by being given more weight.

Other than equal-weighting, Robert Arnott, Jason Hsu and Philip Moore proposed a new weighting scheme in their paper "Fundamental Indexation". They examined a series of equity market indexes weighted by fundamental metrics of size, rather than market capitalization and conclude that annual

return of new weighting schema are on average 213 basis points higher than equivalent capitalization-weighted indexes over the 42 years of the study. By their definition, metric size is computed by book value, income, gross dividends, revenues, sales, and total company employment.

Since passive investment gaining more and more traction in the market, financial practitioners have been evolving based on above academic research and developing to form the following rules in index weighting: Capping, flooring, 45 percent rule and other supplemental rules.

This paper aims to provide the concrete computational methods to perform various kinds of weighting and compare the pros and cons.

2 Commonly practiced weighting in indexing industry

2.1 Capping and flooring

Capping is the most popular rule in indexing. It acknowledges the prominent merits of float-adjusted market capitalization and complemented with a ceiling for large equities so their weight are limited at that value. In creating thematic portfolio on Electric Vehicles, Metaverse, Artificial Intelligence etc. some behemoth kind of companies like Apple, Tesla, or Google constantly show up as the top leader, dominating the whole portfolio. Usually a cap of 12% or 10% is applied so they still keep the top positions but at a moderate level.

The mechanism of Capping usually is carried out in this way, for instance, if the cap is set at 2%, as in the methodology for NYSE FactSet Global Virtual Work and Life Index™ (NYFSVWL), the index weight is first determined by dividing their individual float-adjusted market capitalization by the total float-adjusted market capitalization of all constituents. Individual security weights are capped at 2%, with excess weight redistributed proportionally among remaining securities whose weights are less than 2%. If this redistribution leads to additional security weights exceeding 2%, the aforementioned redistribution process is repeated iteratively until no security weight exceeds 2%.

Even Microsoft Excel's embedded function can realize this capping computation, a "recursive" algorithm is ideal. In computer science, recursion is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem. Such problems can generally be solved by iteration, but this needs to identify and index the smaller instances at programming time hence I wrote a Capping algorithm in Python with the following logic: while there is any security whose weight is greater than the threshold/ceiling value, rank all securities by their current weights, chop off the top large weight to be equal to the threshold, and then redistribute the delta to the rest proportionally, check again so on and so forth, recursively until all securities met the criteria.

To facilitate a user-friendly interface to apply this capping computation, I created web app where one can upload an index or portfolio with uncapped weights and receive a capped version.

This Capping gadget with simple arithmetic steps works fine. However, at its core, I deem it as a minimization/least square optimization problem too. Because given an original portfolio of weights and a constraint of ceiling value, the resulting portfolio weights should be bounded by the capping rule yet still stay in the most approximate vicinity of their original values. In math, it means we need to find new points (Figure 1) by minimizing the sum of the squares of the residuals.

This is same to find the minimum value of residual $r = b - Ax$, $r^T r = (b - Ax)^T (b - Ax)$, $x = (AA^T)^{-1} A^T b$, depending how complicated the constraints are, this could be either linear or non-linear least square problem. Nevertheless, given our particular indexing instance, in Python scipy package,

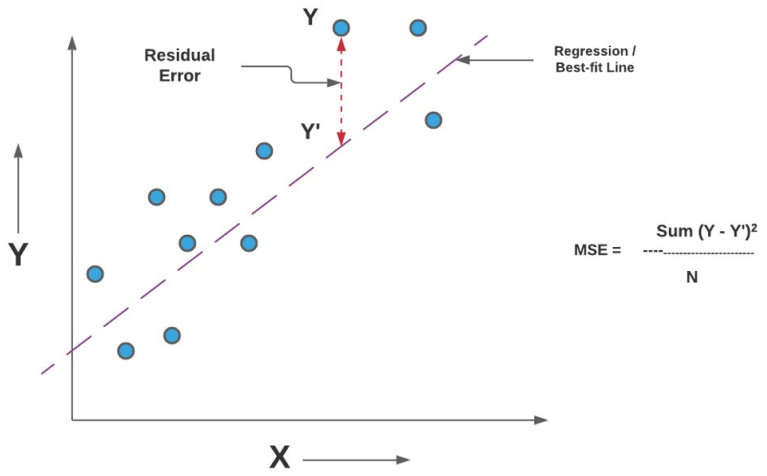

Figure 1: This picture was referenced and uploaded from vitalflux to illustrate residual concept

the method 'SLSQP' suites well. The details are outlined in D Kraft's paper - A software package for sequential quadratic programming published in 1988 [2], where the gist of this algorithm is as below.

$$(NLP) : \min_{x \epsilon R^n} f(x) \tag{1}$$

subject to

$$g_j(x) = 0, \qquad j = 1, ..., m_e \tag{2}$$
$$g_j(x) \geq 0, \qquad j = m_e + 1, ..., m, \tag{3}$$
$$x_1 \leq x \leq x_n \tag{4}$$
$$\tag{5}$$

(NLP)

$$L(x, \lambda) = f(x) - \sum_{j=1}^{m} \lambda_j g_j(x) \tag{6}$$

and in the standard form of quadratic programming:

$$(QP) : \min_{d \epsilon R^n} \frac{1}{2} d^T B^k d + f(x^k) d \tag{7}$$

subject to

$$g_j(x^k) d + g_j(x^k) = 0, \qquad j = 1, ..., m^e \tag{8}$$
$$g_j(x^k) d + g_j(x^k) \geq 0, \qquad j = m_e + 1, ..., m. \tag{9}$$

I am going to use a sample Petcare weight portfolio to illustrate using this numerical optimization approach to find the optimal portfolio weights satisfying the 10% Capping rule.

```
## Capping Optimization.py
from scipy import optimize
from scipy.optimize import minimize

ow =
    [0.84,4.07,2.20,2.37,0.14,1.08,10.41,0.62,0.67,2.38,9.10,4.25,4.94,0.03,4.89,2.72,11.57,2.71,
1.34,4.22,3.84,1.18,0.45,1.19,5.54,1.60,0.82,0.75,2.40,6.25,0.44,3.80,1.19]
```

```
np.sum(ow)
ow = [x/100.0 for x in ow]

def objective_fcn(weights):
    listo = [abs(x-y) for (x, y) in zip(ow, weights)]
    return np.sum(listo)

def equality_constrain(weights):
    return 1 - np.sum(weights)

def inequality_constrain(weights):
    return 0.1 - max(weights)

n = len(ow)
# init_guess = np.repeat(1/n, n)
init_guess = ow

bounds = ((0.0, 1.0),) * n
constraint1 = {'type': 'eq', 'fun': equality_constrain}
constraint2 = {'type': 'ineq', 'fun': inequality_constrain}
constraint = [constraint1, constraint2]

result = minimize(objective_fcn, init_guess, method='SLSQP', options={'disp':
    True,'maxiter':1000, 'ftol': 1e-9,}, bounds=bounds, constraints = constraint)
```

The result is as the following
Optimization terminated successfully (Exit Mode 0)
Current function value: 0.03959999997337148
Iteration: 46
Function evaluations: 1653
Gradient evaluations: 45

Now let's compare the straightforward Capping algorithm versus optimization approach. The two are close but different. In the spirit of strictly following the weighting rule, which usually is described as "Weight of individual securities can not exceed 10%, if it is greater than 10%, then it is reduced to be 10% and the excess amounts are redistributed on a pro-rata basis among the remaining securities". Hence, we conclude when the weighting requirements can be executed by direct capping algorithm, we favor this approach instead of optimization tool.

2.2 Additional Constraints

In common practice of indexing nowadays, the regulation as well as asset managers who issue passive funds endorsed by indexes mandates additional rules such as constrain certain industries or countries total weights to a limit value. The purpose is to mitigate concentrated risk exposure. Juggling multiple weighting rules by brute force algorithms becomes more difficult than applying the numerical optimization. In this paper I am going to use The NYSE FactSet Global Blockchain Technologies Index weighting rule as an example to compare these two approaches.

According to the index methodology, the weighting rule is "At the semi-annual Index reconstitutions and rebalances, constituent weights are determined by dividing their individual security-level float-adjusted market capitalization by the total security-level float-adjusted market capitalization of all constituents as of the reference date. These weights are then capped in the following order: (i) If the initial cumulative weight of Tier 1 securities is less than 75%, then it is scaled up to 75%, with individual securities in Tier 1 scaled up and Tier 2 scaled down, on a pro-rata basis. This rule is

Table 1: Compare Capping and Optimization

Original Weight	**Capping Algo**	**Optimization**
0.0084	0.0086	0.0091
0.0407	0.0417	0.0413
0.0220	0.0226	0.0226
0.0237	0.0243	0.0243
0.0014	0.0014	0.0021
0.0108	0.0111	0.0114
0.1041	0.1000	0.1000
0.0062	0.0064	0.0069
0.0067	0.0069	0.0074
0.0238	0.0244	0.0244
0.0910	0.0933	0.0916
0.0425	0.0436	0.0431
0.0494	0.0507	0.0500
0.0003	0.0003	0.0010
0.0489	0.0501	0.0495
0.0272	0.0279	0.0278
0.1157	0.1000	0.1000
0.0271	0.0278	0.0277
0.0134	0.0137	0.0140
0.0422	0.0433	0.0428
0.0384	0.0394	0.0390
0.0118	0.0121	0.0124
0.0045	0.0046	0.0052
0.0119	0.0122	0.0125
0.0554	0.0568	0.0560
0.0160	0.0164	0.0166
0.0082	0.0084	0.0089
0.0075	0.0077	0.0082
0.0240	0.0246	0.0246
0.0625	0.0641	0.0631
0.0044	0.0045	0.0051
0.0380	0.0390	0.0386
0.0119	0.0122	0.0125

relaxed in order to meet subsequent constraints/caps (ii) - (iv) below. (ii) Individual security weights are capped in Tier 1 at 12% and in Tier 2 at 4%, with any excess amounts redistributed among the remaining securities in their respective tier on a pro-rata basis, subject to the 12% and 4% caps. The redistribution of excess amounts can secondarily occur to the other tier to meet the 12% and 4% caps. (iii) Next, for any Tier 1 security whose resulting Index weight is greater than ten times their three-month ADTV liquidity weight when measured against other Tier 1 securities ("Liquidity Cap Value"), its weight is instead capped at the Liquidity Cap Value, with any excess amounts redistributed among the remaining securities in Tier 1 on a pro-rata basis, subject to the 12% cap in (ii). The redistribution of excess amounts can secondarily occur to securities in Tier 2 to meet this cap, subject to the 4% cap in (ii). (iv) Finally, if the cumulative weight of individual securities in Tier 1 whose weights are greater than 4.5% is greater than 45%, then the lowest weighted securities greater than 4.5% are reduced to 4.5% in ascending order until the cumulative weight no longer exceeds the 45% cap. Excess amounts are redistributed on a pro-rata basis among the remaining securities in Tier 1 with weights less than 4.5%, subject to the caps in (ii) and (iii). Due to the low number of qualifying Tier 1 securities, this step was not executed prior to the March 2021 reconstitution".

To sum up, above narratives conveys 4 rules: Tier1 capped at 12%, Tier2 capped at 4%, Tier1 and Tier2 in aggregate limited to be 75 to 25 respectively, 45% rule, and on top of all those, there is a

liquidity based maximum weight rule.

It's not that easy to write a simple capping algorithm, but feasible to nest various logic together to accomplish. For the latest composition file as following, this nested algo works fine.

Table 2: Latest Composition of the Blockchain Index

Symbol	Name	Float Market Cap	3-Month ADTV)	Tier
COIN-US	Coinbase Global, Inc.	39397.02	1499.95	1
MARA-US	Marathon Digital Holdings Inc	3857.29	620.14	1
RIOT-US	Riot Blockchain Inc	3198.37	457.23	1
HUT-US	Hut 8 Mining Corp.	1426.33	148.66	1
CAN-US	Canaan Inc.	912.09	53.8	1
BITF-US	Bitfarms Ltd.	952.89	52.31	1
HVBT-US	HIVE Blockchain Technologies Ltd	1084.63	36.56	1
GLXY-CA	Galaxy Digital Holdings Ltd.	1838.49	17.16	1
ARB-GB	Argo Blockchain Plc	738.5	9.81	1
VYGR-CA	Voyager Digital Ltd.	1855.78	8.35	1
BTBT-US	Bit Digital, Inc.	224.94	104	1
EBON-US	Ebang International Holdings, Inc.	168.14	6.92	1
NB2-DE	Northern Data AG	1376.84	4.47	1
EQOS-US	Diginex Limited	96.9	12.43	1
ADE-DE	Bitcoin Group SE	185.91	2.69	1
DMGI-CA	DMG Blockchain Solutions, Inc.	122.08	1.53	1
BIGG-CA	BIGG Digital Assets Inc.	217.36	1.5	1
NVDA-US	NVIDIA Corporation	736176.21	9283.65	2
PYPL-US	PayPal Holdings Inc	215760	3014.89	2
AMD-US	Advanced Micro Devices, Inc.	172943.68	6502.07	2
IBM-US	International Business Machines	106428.41	714.27	2
SQ-US	Square, Inc.	71206.05	1740.72	2
NPN-ZA	Naspers Limited	66014.8	118.36	2
WKL-NL	Wolters Kluwer NV	29243.67	53.84	2
4689-JP	Z Holdings Corporation	17907.26	123.63	2
HOOD-US	Robinhood Markets, Inc.	14446.03	282.77	2
9613-JP	NTT DATA Corporation	13167.13	60.06	2
SCB-TH	Siam Commercial Bank Public	8619.8	73.77	2
DXC-US	DXC Technology Co.	7551.99	55.29	2
AMBA-US	Ambarella, Inc.	6723.67	139.03	2
8473-JP	SBI Holdings, Inc.	6325.43	37.52	2
ALLFG-NL	Allfunds Group plc	4928.93	6.76	2
MXL-US	MaxLinear inc	4756.09	29.49	2
SI-US	Silvergate Capital Corp.	4220.67	172.01	2
KC-US	Kingsoft Cloud Holdings Ltd	3419.93	31.6	2

Next, apply the optimization conditions as in the following codes:

```
## weight_optimization.py
from scipy import optimize
from scipy.optimize import minimize

tfm = dmark['Float Market Cap ($Mil, USD)'].sum()
dmark['weight'] = dmark['Float Market Cap ($Mil, USD)']/tfm
tier1 = dmark[dmark.Tier == 1]
tier1tl = tier1['3-Month ADTV ($Mil, USD)'].sum()
tier1['lweight'] = tier1['3-Month ADTV ($Mil, USD)']/tier1tl
tier1['weight'] = tier1.weight/tier1.weight.sum()*0.75
```

```
tier2 = dmark[dmark.Tier == 2]
tier2['weight'] = tier2.weight/tier2.weight.sum()*0.25

df = tier1.append(tier2)
df = df[['Tier','weight']].reset_index(drop=True)
df.head(30)
weight = df.weight.tolist()
lweight = tier1.lweight.tolist()
# objective is to minimize the variation in adjusting per constraints
def objective_fcn(weights):
    # listo = [abs(x-y) for (x, y) in zip(weight, weights)]
    listo = [ (x-y)**2 for (x, y) in zip(weight, weights)]
    return np.sum(listo)

# constraint 0, final weights add up to be 100%
def equality_constrain(weights):
    return 1 - np.sum(weights)
def equality_constrain1(weights):
    return 0.75 - np.sum(weights[0:17])
# constraint 1, tier 1 capped at 12%
def inequality_constrain1(weights):
    return 0.12 - max(weights[0:17])
# constraint 2, tier 2 capped at 4%
def inequality_constrain2(weights):
    return 0.04 - max(weights[17:])
# constraint 3, tier 1 weight can not be greater than 10 times of liquid based weight
def inequality_constrain3(weights):
    listol = [(10*v - u) for (u, v) in zip(weights[0:17], lweight)]
    return np.min(listol)
# adding 45% rule
def inequality_constrain4(weights):
    list45 = [x for x in weights if x > 0.045]
    return 0.45 - np.sum(list45)

n = len(weight)
# init_guess = np.repeat(1/n, n)
init_guess = weight
bounds = ((0.0, 1.0),) * n
constraint0 = {'type': 'eq', 'fun': equality_constrain}
constrainte1 = {'type': 'eq', 'fun': equality_constrain1}
constraint1 = {'type': 'ineq', 'fun': inequality_constrain1}
constraint2 = {'type': 'ineq', 'fun': inequality_constrain2}
constraint3 = {'type': 'ineq', 'fun': inequality_constrain3}
constraint4 = {'type': 'ineq', 'fun': inequality_constrain4}
constraint = [constraint0, constrainte1, constraint1, constraint2, constraint3,constraint4]

result = minimize(objective_fcn, init_guess, method='SLSQP', options={'disp':
    True,'maxiter':1000, 'ftol': 1e-9,}, bounds=bounds, constraints = constraint)
print(result)
```

Compare the results from capping algo and optimization:

The output function value 0.17534448183723447, not close to 0. The values are not as ideal as the rigid algo's in terms of strictly complying the constraining rules mandated.

Table 3: Compare Capping and Optimization

Original Weight	Capping Algo	Optimization
0.0255	0.1200	0.1200
0.0025	0.1200	0.1070
0.0021	0.1200	0.0500
0.0009	0.0704	0.0742
0.0006	0.0450	0.0450
0.0006	0.0450	0.0450
0.0007	0.0450	0.0603
0.0012	0.0450	0.0552
0.0005	0.0323	0.0299
0.0012	0.0275	0.0270
0.0001	0.0212	0.0450
0.0001	0.0159	0.0220
0.0009	0.0147	0.0123
0.0001	0.0091	0.0405
0.0001	0.0089	0.0085
0.0001	0.0050	0.0038
0.0001	0.0049	0.0044
0.4757	0.0400	0.0401
0.1394	0.0400	0.0398
0.1118	0.0400	0.0330
0.0688	0.0379	0.0204
0.0460	0.0254	0.0162
0.0427	0.0235	0.0157
0.0189	0.0104	0.0070
0.0116	0.0064	0.0112
0.0093	0.0051	0.0077
0.0085	0.0047	0.0107
0.0056	0.0031	0.0100
0.0049	0.0027	0.0049
0.0043	0.0024	0.0066
0.0041	0.0023	0.0050
0.0032	0.0018	0.0045
0.0031	0.0017	0.0033
0.0027	0.0015	0.0079
0.0022	0.0012	0.0059

2.3 Recommend Weighting based on Return Optimization and Volatility Minimization

Indexers and ETF issuers can come up with all sorts of these complex and arbitrary weighting rules, algorithm approach does a better job in implementing based on above analysis. However, optimization approach is set for achieving two significant goals in financial asset management - maximizing the total return and minimize the volatility. The mathematical framework of Modern Portfolio Theory authored by Markowitz [4] provides the foundation for practicing.

I will continue to use the Blockchain index as example by compiling daily return in the past one year for each component to compute return and variance matrix. The function to compute the portfolio return and portfolio volatility are as follows:

```
def portfolio_return(weights, returns):
    return weights.T @ returns
```

```
def portfolio_vol(weights, cov):
    return (weights.T @ cov @ weights)**0.5
```

In this section I will try various scenarios to optimizing return and/or volatility, aiming to provide guidance to design the weighting rules for indexes.

First, I set objectives simply to minimize volatility or maximize return respectively without imposing any constraints; second, set the same objectives but applying the Blockchain weighting constraints as detailed above; third, purely apply Markowitz's efficient frontier theory and find the optimal weights for the Maximum Sharpe Ratio (MSR) portfolio.

Simply setting the objective to minimize the volatility and maximize the portfolio return respectively inevitably leads to extreme weighted portfolio. So what's more interesting is to impose the constraints too. The following table 4 shows these two weights with the same constraints applied for Blockchain index.

Table 4: Weight Optimization Aimed to Minimize Volatility and Maximize Portfolio

Symbol	**Minimize Volatility**	**Maximize Return**
COIN-US	0.0000	0.1284
MARA-US	0.0000	0.0267
RIOT-US	0.0000	0.0315
HUT-US	0.0000	0.0508
CAN-US	0.0000	0.0264
BITF-US	0.0000	0.0149
HVBT-US	0.0000	0.0390
GLXY-CA	0.0000	0.1266
ARB-GB	0.0000	0.0390
VYGR-CA	0.0000	0.0000
BTBT-US	0.0178	0.0390
EBON-US	0.0000	0.0390
NB2-DE	0.0285	0.0839
EQOS-US	0.0000	0.0000
ADE-DE	0.0000	0.0398
DMGI-CA	0.0000	0.0240
BIGG-CA	0.0000	0.0411
NVDA-US	0.0000	0.0000
PYPL-US	0.0000	0.0114
AMD-US	0.0837	0.0000
IBM-US	0.0579	0.0259
SQ-US	0.0000	0.0000
NPN-ZA	0.0000	0.0447
WKL-NL	0.2942	0.0000
4689-JP	0.0000	0.0447
HOOD-US	0.0000	0.0000
9613-JP	0.0335	0.0447
SCB-TH	0.0000	0.0226
DXC-US	0.1836	0.0000
AMBA-US	0.1681	0.0000
8473-JP	0.0143	0.0447
ALLFG-NL	0.0000	0.0000
MXL-US	0.0079	0.0112
SI-US	0.0000	0.0000
KC-US	0.1105	0.0000

Lastly, I prepared the daily return of the latest Blockchain composition file for the past one year (252 trading days) and plot it's efficient frontier by simulating 5000 various weighting schemes.If I select small number of assets/stocks the output displays a classical efficient frontier graph 2:

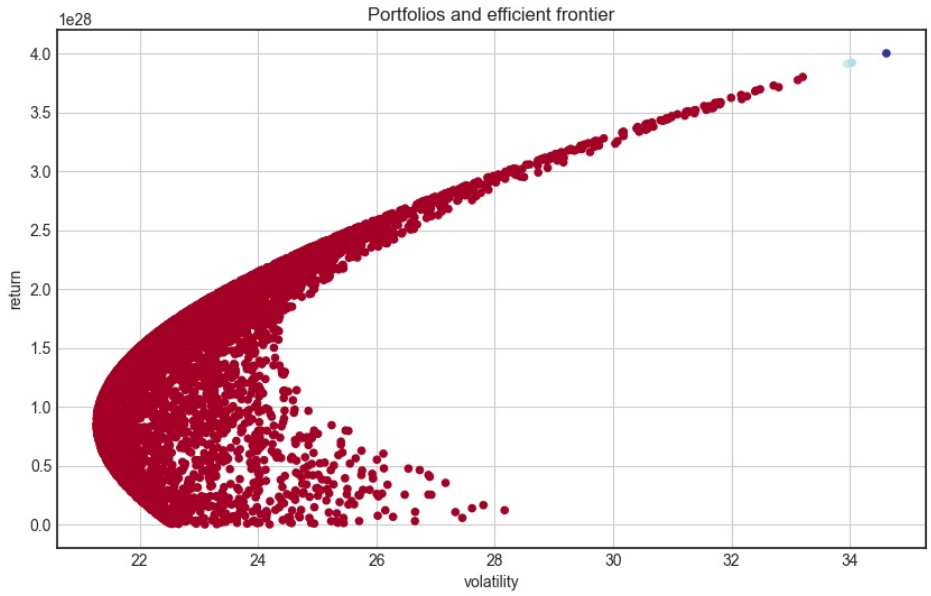

Figure 2: 4689-JP, 8473-JP and 9613-JP from 20210108 to 20220106; note return in percentage

However, when the number of assets/stocks are set to be the exact number in this portfolio, the frontier graph 3 looks convoluted:

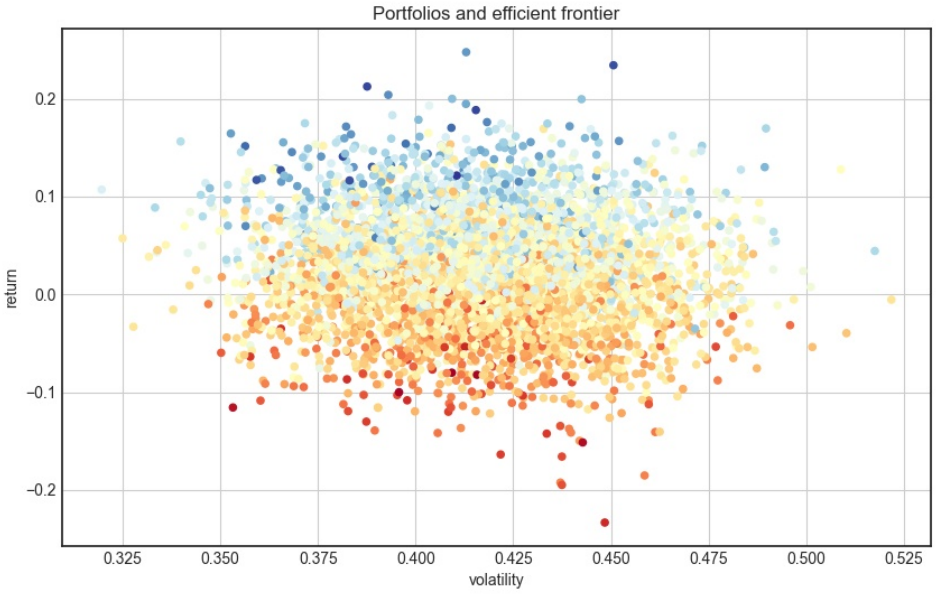

Figure 3: Entire Portfolio from 20210108 to 20220106

The MSR(Maximum Sharpe Ratio) portfolio's return 9.1%, volatility is 43.76% and Sharpe Ratio is 0.23. The corresponding weight is listed below to compare side by side with the Algo based weight:

Table 5: Compare Capping and Optimization

Symbol	Original Weight	Capping Algo	Maximum Sharpe Ratio
COIN-US	0.0255	0.1200	0.0076
MARA-US	0.0025	0.1200	0.0305
RIOT-US	0.0021	0.1200	0.0107
HUT-US	0.0009	0.0704	0.0710
CAN-US	0.0006	0.0450	0.0451
BITF-US	0.0006	0.0450	0.0158
HVBT-US	0.0007	0.0450	0.0513
GLXY-CA	0.0012	0.0450	0.0532
ARB-GB	0.0005	0.0323	0.0087
VYGR-CA	0.0012	0.0275	0.0073
BTBT-US	0.0001	0.0212	0.0285
EBON-US	0.0001	0.0159	0.0120
NB2-DE	0.0009	0.0147	0.0554
EQOS-US	0.0001	0.0091	0.0265
ADE-DE	0.0001	0.0089	0.0473
DMGI-CA	0.0001	0.0050	0.0232
BIGG-CA	0.0001	0.0049	0.0067
NVDA-US	0.4757	0.0400	0.0440
PYPL-US	0.1394	0.0400	0.0124
AMD-US	0.1118	0.0400	0.0579
IBM-US	0.0688	0.0379	0.0238
SQ-US	0.0460	0.0254	0.0131
NPN-ZA	0.0427	0.0235	0.0160
WKL-NL	0.0189	0.0104	0.0140
4689-JP	0.0116	0.0064	0.0042
HOOD-US	0.0093	0.0051	0.0182
9613-JP	0.0085	0.0047	0.0435
SCB-TH	0.0056	0.0031	0.0650
DXC-US	0.0049	0.0027	0.0006
AMBA-US	0.0043	0.0024	0.0669
8473-JP	0.0041	0.0023	0.0004
ALLFG-NL	0.0032	0.0018	0.0465
MXL-US	0.0031	0.0017	0.0292
SI-US	0.0027	0.0015	0.0187
KC-US	0.0022	0.0012	0.0251

This theoretical portfolio can be achieved via Optimization other than the above simulation. Additionally, we can also impose the constraints and generate the most ideal weight scheme for an index product.

3 Conclusion

As is well stated in Stock Index Weighting Matters, two index portfolios - Invesco SP 500 Equal Weighted ETF (RSP) and SPDR SP 500 ETF Trust (SPY) can yield significantly different returns while tracking the same SP 500 index composition. Weighting has been a dedicated research in academia as well as in real investment practice.

In indexing industry, emphasis is not only to chase return but also for risk diversification, which also includes tradability/liquidity risk diversification. What's more especially for thematic index construction, highlighting the theme by assigning weight based on theme relevancy rather than market capitalization. What it leads to often time is a float-adjusted market capitalization based portfolio with additional constraints on capping, flooring, country limit, industry limit etc.

This paper compared the recursive algorithm approach and Least-Square optimization approach in implementing constraints. We conclude the former approach is more conforming and flexible but the optimization approach is easier to customize and hence more efficient. Even we've been practicing the former approach in daily work, We would recommend to employ the latter approach more often if it meets the requirement. Only when optimization fails we shall recourse to algorithm.

This paper also explored maximization returns within constraints based on Markowitz's efficient frontier theory, which theoretically satisfies the risk diversification purpose as well as achieving maximum return. It can provide referencing framework for index construction. Edward O. Thorp [3] in his paper -Portfolio Choice and the Kelly Criterion- back in 1975 discussed the use of the Kelly criterion for portfolio management. He particularly mentions that "On November 3, 1969, a private institutional investor decided to . . . use the Kelly criterion to allocate its assets". This was actually a private limited partnership, specializing in convertible hedging, which I managed. A notable competitor at the time (see Institutional Investor (1998)) was future Nobel prize winner Harry Markowitz. After 20 months, our record as cited was a gain of 39.9% versus a gain for the Dow Jones Industrial Average of +4.2%." His great article as well as actual investment accomplishment apparently is very compelling in enticing further exploration on applying Kelly Criterion. However it is beyond the scope of this paper.

References

[1] Robert D Arnott, Jason Hsu, and Philip Moore. Fundamental indexation. *Financial Analysts Journal*, 61(2):83–99, 2005.

[2] Dieter Kraft et al. A software package for sequential quadratic programming. 1988.

[3] Edward O Thorp. Portfolio choice and the kelly criterion. In *Stochastic optimization models in finance*, pages 599–619. Elsevier, 1975.

[4] Graeme West. An introduction to modern portfolio theory: Markowitz, cap-m, apt and black-litterman. *Parktown North: Financial Modelling Agency*, 2006.