

# GA와 Solver를 활용한 시장 리스크 헤지 방안 모색

## 샤프지수 최대화를 중심으로

---

2019136008 김민석

2021121015 박상현

2022123268 박서연

2020125090 진영웅

# 차례

---

01 주제 선정 이유

---

02 이론적 배경

---

03 문제 정의

---

04 데이터 수집 및 처리

---

05 최적화

---

06 결론

---

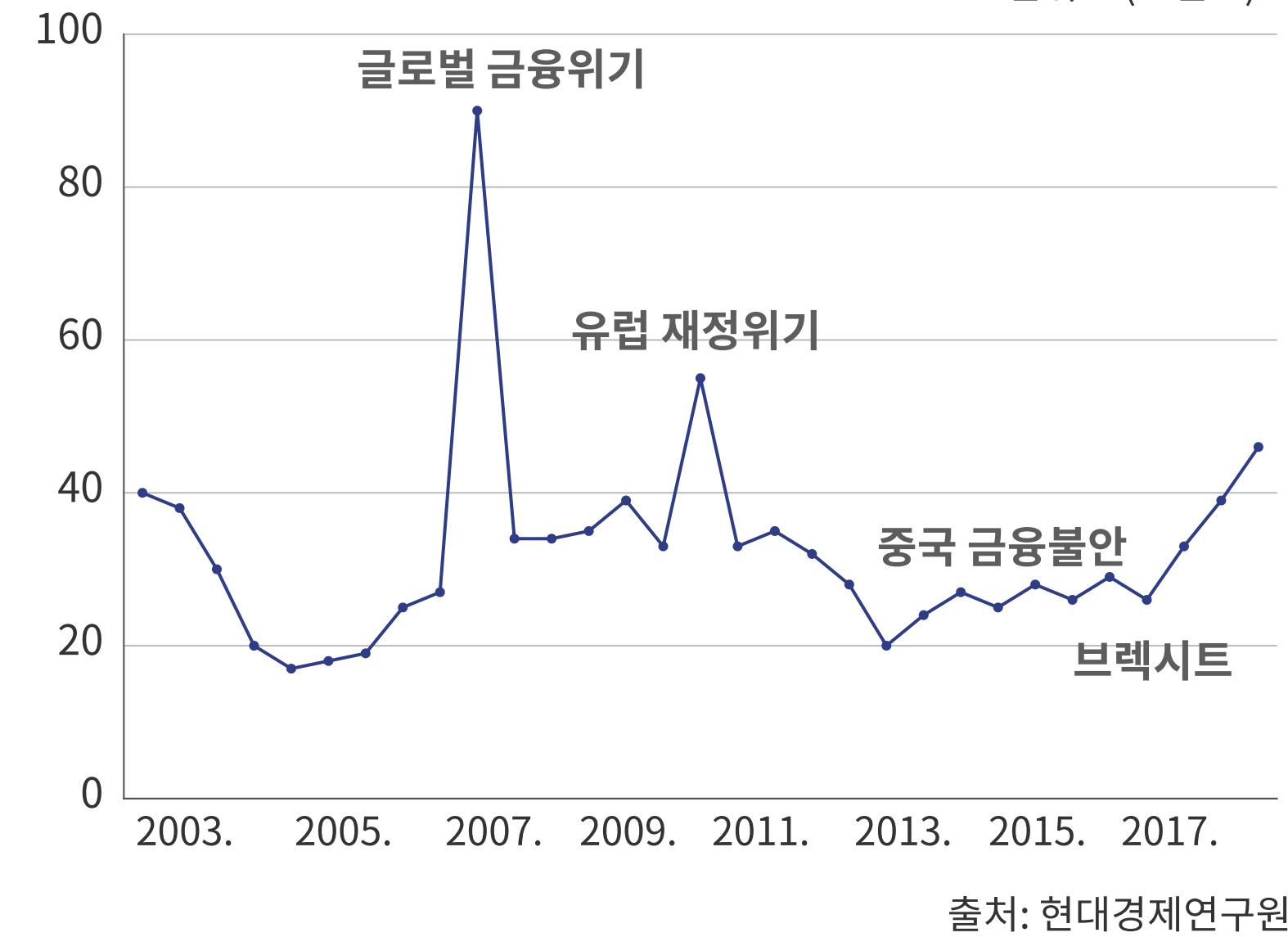
# 01 주제 선정 이유

## 배당금 최적화 포트폴리오를 통한 시장 리스크 해지

- 갈수록 가속화되는 증시 변동성
  - 이는 코로나19와 같은 글로벌 이슈에서 더욱 두드러짐
- 
- 시장 상황에서의 변동성은 곧 리스크
  - 예측 불가능한 시장 리스크 관리에 대한 수요가 높아짐
- 
- 글로벌 경제 타격 이벤트에도 크게 흔들리지 않는 안정적인 수익 창출 방안으로서 배당금 개념 도입
- 
- 단순 자산 포트폴리오가 아닌, 특정한 지표를 바탕으로 배당금 최적화. 즉, 변동성을 완화하는 리스크 해지 방안 강구

## 한국경제의 대내외 불확실성 지수 추이

단위: P(포인트)



출처: 현대경제연구원

\* 특정한 지표: 고든성장모형을 차용한 샤프지수 최대화

## 02 이론적 배경(1)

### 배당금(dividends) 및 배당 포트폴리오

#### [ 배당금 ]

기업이 이익의 일부를 주주에게 현금이나 주식 형태로 분배하는 금액. 주주들에게 투자에 대한 보상을 제공하는 방법.

#### [ 배당 포트폴리오 ]

주로 배당금을 지급하는 주식으로 구성된 투자 포트폴리오. 장기적인 배당 수익을 통해 안정적인 현금 흐름을 창출하고, 장기적으로 자산 가치를 증가시키는 것을 목표로 함.



이러한 안정성을 극대화하기 위해 '귀족 배당주'만을 활용한 포트폴리오 구성



시장 리스크를 헤지하는 전략으로서 활용하는 방법 제언

대표 배당주 리스트 예시

지급 주기	종목명	티커	시가총액 (십억달러)	배당수익률 (%, 12M)	배당성향* (%)	배당 증액 기간(년)
1/3/6/9	펩시코	PEP	231.8	3.0	65	52
	디지털 리얼티	DLR	40.9	3.6	76	-
1/4/7/10	TSMC	TSM	565.5	1.3	35	*배당지급 19년
	필립모리스	PM	139.1	5.8	82	15
2/5/8/11	Morgan Stanley	MS	140.0	4.0	55	10
3/6/9/12	BROADCOM	AVGO	574.1	1.7	45	13
	화이자	PFE	155.7	6.1	77	15
4/7/10/12	코카콜라	KO	262.4	3.2	69	62

참고: 2024.2.20 종가. \*리츠는 FFO Ratio  
자료: Bloomberg, 삼성증권

#### [ 귀족 배당주 ]

지속적으로 배당금을 증가시켜 온 기업의 주식. 일반적으로 최소 25년 이상 연속적으로 배당금을 증가시킨 기업들을 지칭함. 안정적인 재무 상태와 꾸준한 수익성을 자랑하는 기업들이 대부분에 해당함.

## 02 이론적 배경(2)

### 샤프지수 (Sharpe Ratio)

"위험 대비 투자 수익에 대한 지표"

$$Sharpe = (R_p - R_f) / \sigma_p$$

= 위험만 반영하였을 때의 포트폴리오가 내는 수익의 정도

$R_p$  : 포트폴리오의 기대수익  
(포트폴리오를 통해 실제 취득될 것으로 기대되는 수익의 정도)

$R_f$  : 무위험자산 수익  
(시장리스크에 상관없이 안정적으로 취득되는 수익의 정도)

$\sigma_p$  : 포트폴리오 초과수익의 표준편차

#### [ 샤프지수의 의미 ]

'포트폴리오의 기대수익으로부터 무위험자산 수익을 뺀 값', 즉 샤프지수의 분자는 포트폴리오의 기대수익 중 시장리스크에 영향을 받는 자산(위험 자산)이 내는 수익의 정도를 나타냄.

이러한 지표를 '포트폴리오 초과수익의 표준편차'로 나눈 샤프지수는, 시장리스크가 존재할 때 위험 대비 포트폴리오가 내는 기대수익의 정도를 나타냄. 다시 말해, 아래의 등식이 성립함.

샤프지수가 높다  더 적은 위험으로 더 높은 수익률을 낸다

 "샤프지수를 최대화하는 포트폴리오를 구성하자"

## 02 이론적 배경(2)

### 샤프지수 (Sharpe Ratio)의 한계점

앞선 페이지에서 살펴본 샤프지수의 도출 과정에는 아래와 같은 한계가 있다.

$$Sharpe = \frac{(R_p - R_f)}{\sigma_p}$$

$$R_p = \frac{1}{T} \sum_{t=1}^T R_{p,t}$$

$T$  : 과거 수익률의 기간       $R_{p,t}$  : 시점 t에서의 포트폴리오 수익률

기대수익률( $R_p$ )을 단순히 과거 수익률의 평균을 통해 도출하기 때문에 이 과정에서 현실과의 괴리 발생

좌측의 공식을 그대로 차용해서 단순히 과거 수익률의 평균을 통해 미래의 기대수익률을 계산하면, 실제 시장 메커니즘과의 괴리가 발생하여 미래에 대한 예측력이 떨어진다는 문제가 발생함

문제 1. 실제 포트폴리오의 기대수익률은 단순히 과거 수익률에 의존하지 않는다.  
: 하지만 미래를 예측하기 위해서는 과거의 데이터를 활용할 수밖에 없기에, 좀 더 정밀하게 책정된 미래 예측 지표가 필요함

2. 그렇다면 과거 데이터를 어떻게 활용할 것인가?  
: 과거 수익률의 흐름을 최대한 반영하여 미래의 기대수익률을 예측하는 것은 합리적이지만, 단순 산술평균보다는 정밀한 지표가 필요함

다시 말해, 기대수익률 도출 과정을 정교화함으로써 현실과의 괴리를 완화하고 미래에 대한 예측력을 제고할 필요성이 있음

기대수익률의 도출에 또 다른 수리모델(고든성장모형)을 적용

## 02 이론적 배경(3)

---

### 고든성장모형(Gordon Growth Model)

주식의 내재가치를 추정하는 방법론

$$P = D(1 + g) / (k_e - g)$$

배당성장률과 배당수익률을 활용하여 주가를 도출하는 방법

$k_e$  : 주식투자의 요구수익률(기대수익률과 유사한 개념)

$P$  : 주가

$D$  : 배당금

$g$  : 배당성장률

샤프지수의 기대수익률( $R_p$ ) 도출 과정에 고든성장모형 차용



샤프지수의 한계점(단순히 과거 수익률의 평균을 계산하여 미래의 기대수익률을 도출하는 과정) 보완

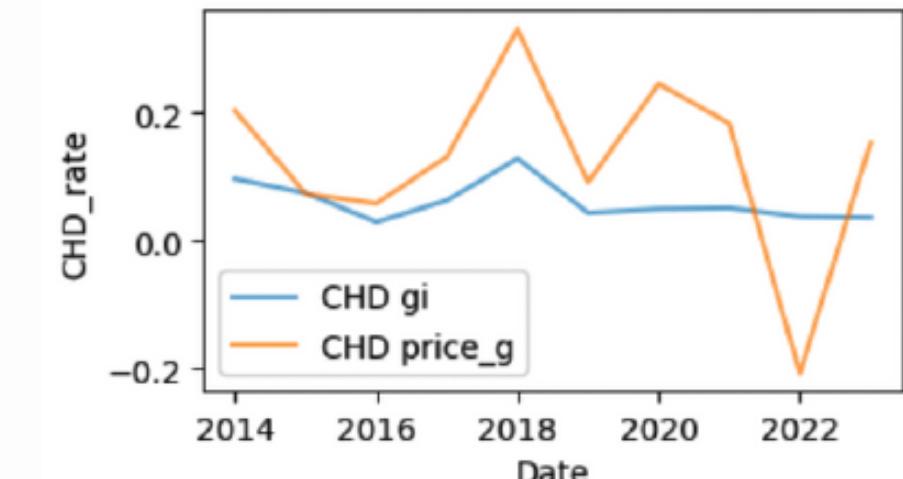
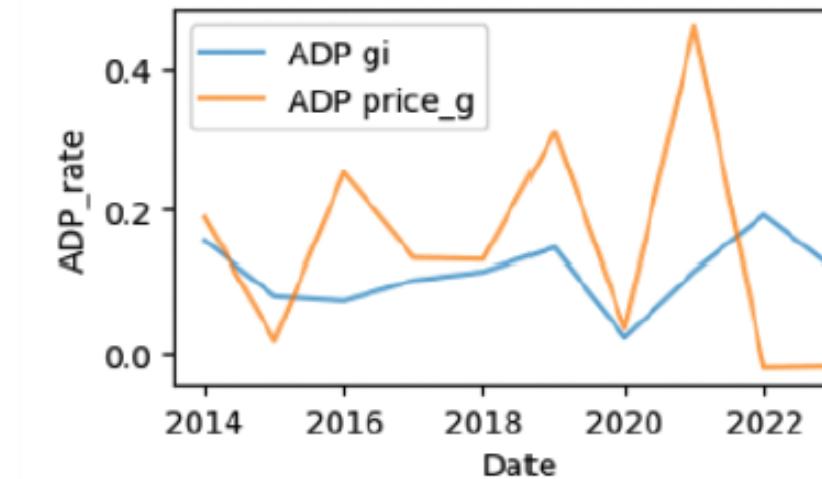


배당금을 적용한 포트폴리오 수익을 기준 샤프지수보다 정밀하게 책정하는 개선된 지표로 활용

## 02 이론적 배경(3)

### 고든성장모형(Gordon Growth Model)

고든성장모형은 기업의 이익과 배당이 매년  $g\%$  만큼 동일하고 일정하게 성장한다고 가정함. 하지만 현실적으로 기업의 이익과 배당의 증가분이 상수인 경우는 매우 드물게 존재함. 따라서 고든성장 모형의 활용에 대한 검증이 필요함.



이를 위해 배당성장률의 연간 변화율(파란색)과 주가 연간 변화율(주황색) 시각화를 통해 두 변화율의 흐름을 비교함. (우측 그래프 참고)



데이터의 주가 연간 변화율과 배당 성장률의 연간 변화율을 비교했을 때 두 변화율이 완전히 일치하지는 않았지만,

그럼에도 불구하고 상승 및 하락의 경향성은 비슷한 기조를 보임을 확인함.

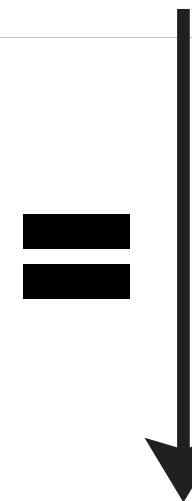
이를 바탕으로, 기업의 주가 성장률을 배당성장률로 보는 근거를 확보함. 즉,  $R_p$ 의 도출 과정에 고든성장모형을 적용하기로 결정함.

## 03 문제 정의: Objective Function

### 목적식: 샤프지수 최대화

$$Max : Sharpe = \frac{R_p - R_f}{\sigma_p}$$

$$E(p) = \sum_i g_i w_i + \sum_i w_i d_i (1 + g_i)$$



이 과정에서,

1. Rp 대신 고든성장모형을 적용한 E(p)를 기대수익률 지표로 활용
2. Rf는 앞서 언급한 0.01(일반적인 국채 수익률 값) 활용
3. sigma(p)에는 우측 하단의 유도 공식 대입

$$Max \frac{\sum_i g_i w_i + \sum_i w_i d_i (1 + g_i) - 0.01}{\sqrt{\sum_i \sum_j w_i w_j Cov(t_i, t_j)}}$$

$$\sigma_p = \sqrt{\sum_i \sum_j w_i w_j Cov(t_i, t_j)}$$

## 03 문제 정의: Decision Variable/Constraints/Parameter

---

### Decision Variable

---

$w_i$

포트폴리오에서  $i$ 번째 종목의 비중

프로젝트 초기에는  $i$ 번째 종목의 구매 개수( $X_{-i}$ )를 Decision Variable로 설정하여 문제를 풀이했지만, 이 경우 종목가격이 각각 다르기 때문에 GA에서 예산 제약을 만족하는 자손을 생성하기 어렵다는 문제를 발견했다. 이에 의사결정변수를  $w_i$ 로 새롭게 정의하여 문제를 해결하고자 한다.

### Parameter

---

$i$ : 종목 ( $= 1, 2, \dots, 40$ )

$g_i$ : 배당성장률

$d_i$ : 배당수익률

$s_i$ : 주식 가격 수익률

$t_i$ : 총 수익률 ( $= d_i + s_i$ )

### Constraints

---

$$\sum_i w_i = 1, \forall i$$

$$0 \leq w_i \leq 1, \forall i$$

프로젝트 초기에는 종목의 구매 개수와 각 종목의 가격을 곱한 총 투자비용이 현재 가진 자본금을 넘지 못하는 제약식을 추가로 설정하였다. 그러나 의사결정변수를  $i$ 번째 종목의 비중으로 변경함으로써 자본금 제약의 필요성이 없어졌으며 이에 따라 두 개의 제약식으로 문제를 해결하고자 한다.

1. 각 의사결정변수의 값은 주식에 투자할 금액의 비율을 의미하므로 0과 1 사이의 연속형 변수이다.
2. 보유하고 있는 자본금을 전부 투자할 계획이므로, 투자금액의 비율을 의미하는 의사결정변수의 합은 1이 되어야한다.

# 04 데이터 수집 및 처리

## 포트폴리오 기대수익 ( $R_p$ )

### [ $R_p$ 유도 과정]

1. 기업별 배당금 수익률 계산 (= 배당금 / 종가)
2. 기업별 총 수익률 계산
  - a. 주식 가격 수익률 = (현시점 종가 - 이전시점 종가) / 이전시점 종가
  - b. 총 수익률 = 배당금 수익률 + 주식 가격 수익률
3. 기업별 평균 수익률 계산 (= sum(총 수익률 / 시점 수))
4.  $R_p$  계산 (= sum(기업별 평균 수익률 \* 각 기업의 가중치))



필요 데이터(데이터 수집 기간: 2019 - 2022년(4개년))

기간별 배당금 데이터, 기간별 종가 데이터, 거래량 데이터

배당금 수익률(데이터 일부)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Date	AFL	ALB	AOS	APD	ATO	BF-B	BRO	CHRW	CINF	CL	CLX	CVX	DOV	ESS
2	2019-04-30	0	0	0.005697	0	0	0	0	0	0	0.01988	0.00395	0	0	0
3	2019-05-31	0.007432	0	0	0	0.007254	0	0.003545	0	0	0	0	0.01409	0.007147	0
4	2019-06-30	0	0.007934	0	0.007897	0	0.004505	0	0.0091230	0.008151	0	0	0	0	0.009871
5	2019-07-31	0	0	0.006876	0	0	0	0	0	0	0.020376	0.003792	0	0	0
6	2019-08-31	0.007436	0	0	0	0.006767	0	0.003129	0	0	0	0	0.0141520	0.007594	0
7	2019-09-30	0	0.008241	0	0.007868	0	0.003941	0	0.008835	0.007367	0	0	0	0	0.008985
8	2019-10-31	0	0	0.006672	0	0	0	0	0	0	0.020624	0.003870	0	0	0
9	2019-11-30	0.007487	0	0	0	0.007963	0	0.003374	0	0	0	0	0.014935	0.006695	0

주식 가격 수익률(데이터 일부)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date	AFL	ALB	AOS	APD	ATO	BF-B	BRO	CHRW	CINF	CL	CLX	CVX
2	2019-04-30	0	0	0	0	0	0	0	0	0	0	0	0
3	2019-05-31	0.054320	-0.129861	-0.137943	0.062695	0.027355	-0.007661	0.054709	-0.068677	0.119293	0.047187	-0.031076	-0.012211
4	2019-06-30	-0.001308	-0.068183	-0.099958	0.007507	-0.035360	-0.000301	-0.030851	-0.052176	-0.003505	-0.039793	-0.019455	-0.038925
5	2019-07-31	0.083411	(0.109020)	0.067895	0.101780	0.091961	0.075822	0.141503	0.087906	0.105721	0.071502	0.097766	0.089590
6	2019-08-31	-0.076190	-0.106192	0.007844	(-0.011407)	0.017685	0.015787	0.023905	-0.008072	0.019908	-0.018100	-0.010007	-0.049188
7	2019-09-30	-0.049705	-0.028696	0.000233	(-0.078616)	-0.042285	0.045958	-0.056597	-0.043093	-0.018899	-0.055706	-0.072203	-0.038859
8	2019-10-31	0.126556	0.097234	0.115185	(0.073142)	0.117201	0.118357	0.114506	0.114568	0.117229	0.065573	0.066865	0.061560
9	2019-11-30	-0.072248	-0.098006	-0.069785	-0.020080	-0.130095	-0.066857	-0.062853	-0.194850	-0.150065	-0.133197	-0.118841	-0.071333
10	2019-12-31	-0.006925	0.042216	-0.051299	0.017195	0.018758	0.010421	0.043834	0.027160	-0.017035	0.038159	0.049387	0.007505
11	2020-01-31	-0.0062240	0.139008	-0.009756	0.007884	0.050691	0.054820	0.068136	0.022782	0.007161	0.024753	0.027901	-0.028741
12	2020-02-29	-0.066303	0.083991	-0.107935	0.018145	-0.011470	-0.039568	0.091326	-0.109485	0.001525	0.035155	0.022827	-0.103324

이때 2 - a 단계의 계산을 위해서는 종가의 시점 구분 필요함.

따라서 시점 구분을 위해 일별로 산출된 종가 데이터를 월별로 환산

(환산기준으로 VWAP(가중평균) 사용)

# 04 데이터 수집 및 처리

## 무위험자산 수익 ( $R_f$ ) 및 포트폴리오 초과수익의 표준편차 ( $\sigma_p$ )

### [ 무위험자산 수익 ]

일반적으로 미국 국채 수익률을 사용하며, 통상 0.01을 가정하고 계산함.  
국채 데이터는 일반인이 접근하기 어려운 편에 해당하여, 이번 프로젝트  
에서도 0.01을 가정하고 계산

### [ 포트폴리오 초과수익의 표준편차 ]

총수익률  $t_i$ 를 이용하여 만든 공분산 행렬과 각 종목별 가중치인  $w_i$ 로  
생성한 벡터  $w$ 와의 이차형식의 곱으로 나타낸다.

$$\begin{aligned}\sigma_p &= \sqrt{\sum_i \sum_j w_i w_j Cov(t_i, t_j)} \\ &= \sqrt{W^\top Cov(t_i, t_j) W}\end{aligned}$$

공분산 행렬의 일부 (40 by 40 Matrix)

	AFL	ALB	AOS	APD	ATO	BF-B	BRO
AFL	0.009296	0.005435	0.002060	0.004686	0.004696	0.004967	0.005317
ALB	0.005435	0.010477	0.003504	0.002963	0.004368	0.004341	0.005565
AOS	0.002060	0.003504	0.005697	0.000769	0.002785	0.003026	0.001819
APD	0.004686	0.002963	0.000769	0.003072	0.002836	0.002101	0.003624
ATO	0.004696	0.004368	0.002785	0.002836	0.004475	0.003391	0.004406
BF-B	0.004967	0.004341	0.003026	0.002101	0.003391	0.003984	0.003072
BRO	0.005317	0.005565	0.001819	0.003624	0.004406	0.003072	0.005558
CHRW	0.003627	0.004599	0.004824	0.001997	0.004612	0.003852	0.003344
CINF	0.006984	0.003879	0.001821	0.003865	0.005102	0.004204	0.005165
CL	0.003654	0.004271	0.001892	0.002577	0.004005	0.002402	0.004156
CLX	-0.000755	0.002261	0.002469	0.000704	0.002366	0.000344	0.001867
CVX	0.007163	0.004468	0.002377	0.003591	0.003711	0.004142	0.004154
DOV	0.006921	0.005843	0.002684	0.003246	0.003406	0.004181	0.004034

코드 구현 (2019년도 초과수익의 표준편차 예시)

```
def sigma(*args):
    weights = []
    for i in args:
        weights.append(i)
    weights = np.array(weights)
    risk = np.sqrt(np.dot(weights.T, np.dot(cov_2019, weights)))
    return(risk)
```

# 04 데이터 수집 및 처리

## 데이터 수집 종목 예시: BF-B

Yahoo Finance

Search for news, symbols or companies

News Finance Spec

My Portfolio News Markets Sectors Screeners Personal Finance Videos

Summary News Chart Conversations Statistics Historical Data

NYSE - Nasdaq Real Time Price - USD

**Brown-Forman Corporation (BF-B)** Follow

**42.62 -1.27 (-2.89%)**

As of 1:02 PM EDT. Market Open.

Jun 10, 2023 - Jun 10, 2024 Historical Prices Daily

Date	High	Low	Close	Adj Close	Volume	
Jun 10, 2024	3.65	42.51	42.63	42.63	669,292	
Jun 7, 2024		<b>0.22 Dividend</b>				
Jun 7, 2024	4.49	43.46	43.89	43.89	2,553,300	
Jun 6, 2024	43.15	44.48	43.13	43.92	3,345,000	
Jun 5, 2024	46.02	46.44	42.72	43.04	42.83	5,664,300
Jun 4, 2024	45.91	46.22	45.63	45.74	45.51	2,258,900
Jun 3, 2024	45.07	46.50	45.05	46.09	45.86	2,180,300
May 31, 2024	44.25	45.94	43.55	45.86	45.63	9,237,500
May 30, 2024	44.18	44.66	44.02	44.27	44.05	2,803,800
May 29, 2024	44.34	44.80	44.20	44.22	44.00	1,975,700
May 28, 2024	45.63	45.63	44.62	44.88	44.66	2,703,600
May 24, 2024	46.49	46.56	45.67	45.76	45.53	1,409,200
May 23, 2024	46.78	46.78	46.04	46.32	46.09	1,757,600
May 22, 2024	47.09	47.54	46.61	47.06	46.83	1,485,300
May 21, 2024	47.13	47.43	46.85	47.31	47.08	1,577,600

1. 25년 동안 배당금을 지속적으로 상승시킨 귀족배당주 40개 선정

2. [야후 파이낸스] - [Historical Data] - 종가 데이터, 배당금 데이터 수집

3. 선정한 종목 40개의 기간 내 데이터를 csv파일로 추출 후 전처리

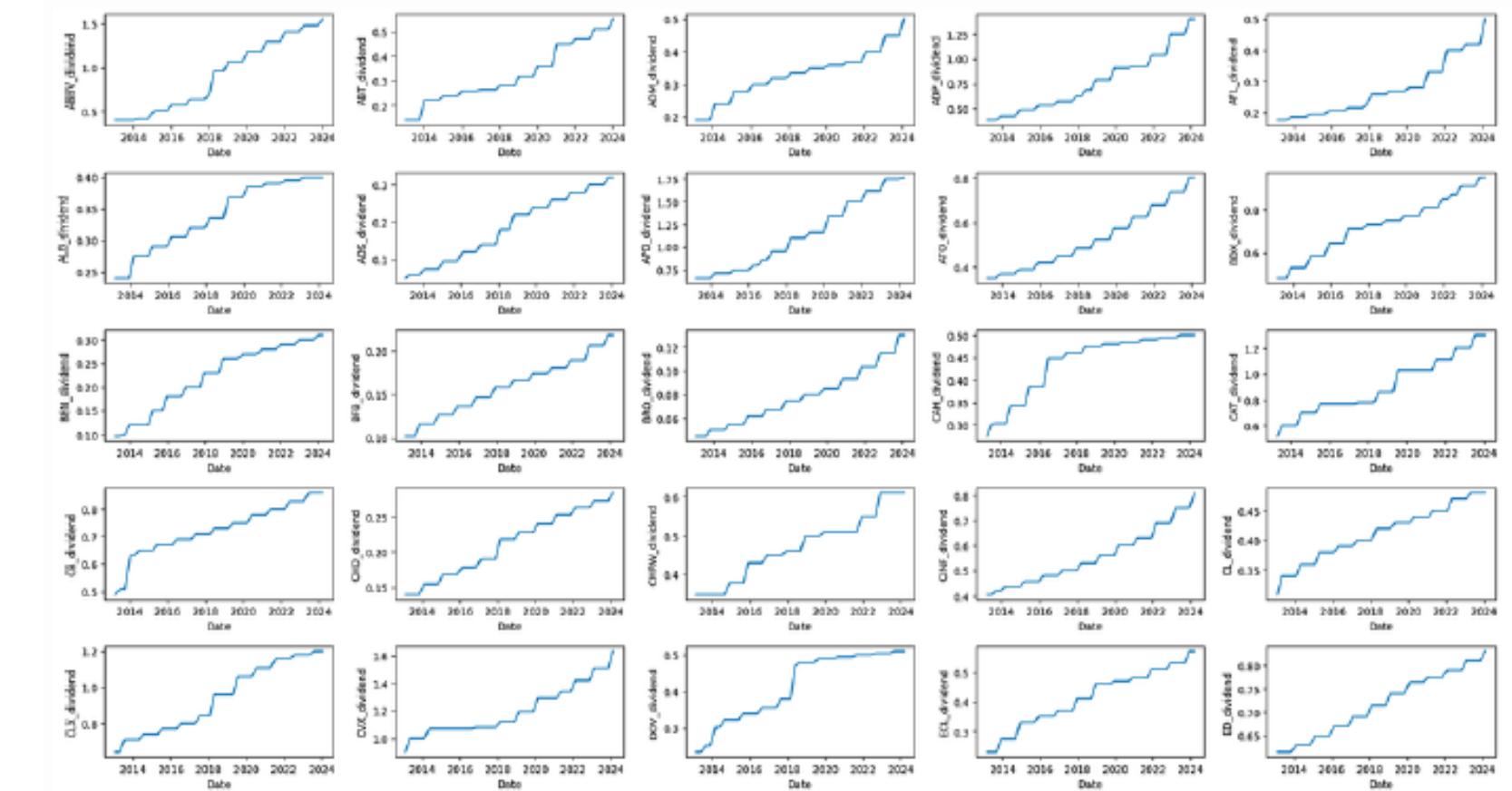
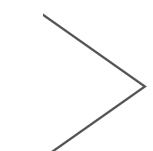
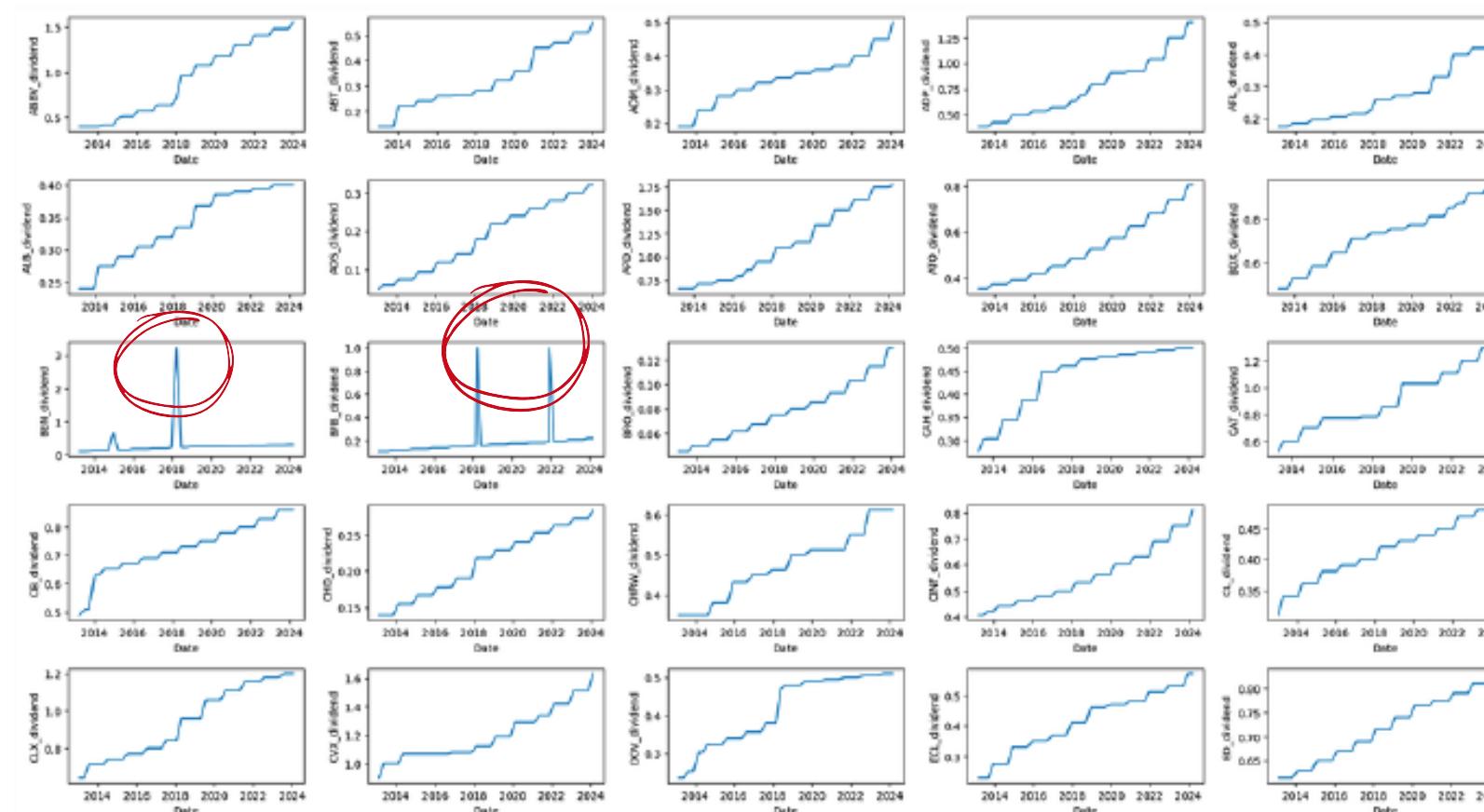
\* 예시 < 티커별 배당금 데이터(raw data) >

YearMonth	ABBV	ABT	ADM	ADP	AFL	ALB	AMCR	AOS	APD	...	SHW	SJM	SPGI	SWK	SYY	TGT	TROW	WMT	WST	XOM	
0	1962-01	0.00	0.00	0.00	0.0	0.00	0.0	0.000	0.00	0.00	0.000	0.00	0.00	0.00	0.0	0.0	0.00	0.0	0.00000		
1	1962-02	0.00	0.00	0.00	0.0	0.00	0.0	0.000	0.00	0.00	0.000	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.01875		
2	1962-03	0.00	0.00	0.00	0.0	0.00	0.0	0.000	0.00	0.00	0.000	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.00000		
3	1962-04	0.00	0.00	0.00	0.0	0.00	0.0	0.000	0.00	0.00	0.000	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.00000		
4	1962-05	0.00	0.00	0.00	0.0	0.00	0.0	0.000	0.00	0.00	0.000	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.01875		
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...		
731	2023-11	0.00	0.00	0.45	0.0	0.42	0.0	0.125	0.00	0.00	0.605	1.06	0.90	0.81	0.0	1.1	0.00	0.000	0.2	0.95000	
732	2023-12	0.00	0.00	0.00	1.4	0.00	0.4	0.000	0.00	1.75	...	0.000	0.00	0.00	0.0	0.0	1.22	0.190	0.0	0.00000	
733	2024-01	1.55	0.55	0.00	0.0	0.00	0.0	0.000	0.32	0.00	...	0.000	0.00	0.00	0.5	0.0	0.00	0.000	0.2	0.00000	
734	2024-02	0.00	0.00	0.50	0.0	0.50	0.0	0.125	0.00	0.00	0.715	1.06	0.91	0.00	0.0	1.1	0.00	0.000	0.0	0.95000	
735	2024-03	0.00	0.00	0.00	1.4	0.00	0.4	0.000	0.00	1.77	...	0.000	0.00	0.00	0.81	0.0	0.0	1.24	0.208	0.0	0.00000

귀족배당주 선정 이유: 배당금을 안정적으로 지급하는 기업을 선정함으로써 주가 차익보다 배당 수익에 더 초점을 맞춘 포트폴리오 구성. 이를 통해 더 정밀한 시장 리스크 해지 방안 모색

# 04 데이터 수집 및 처리

## 데이터 전처리: 배당금 데이터



배당금 데이터 처리 과정에서 이상치(첨값) 확인. 이에 대한 이유로서 특별배당 지급 건이 있었음을 확인  
각 시기별로 지급된 특별배당이 분기 외 별도 지급건인 경우 삭제, 대체지급인 경우 직전 직후 지급 액수의 평균으로 대체

이외에도 액면 분할 데이터를 추가 확보하여 배당 지속 지급 및 배당 지속 상승 여부 등 확인 후 데이터 전처리 진행

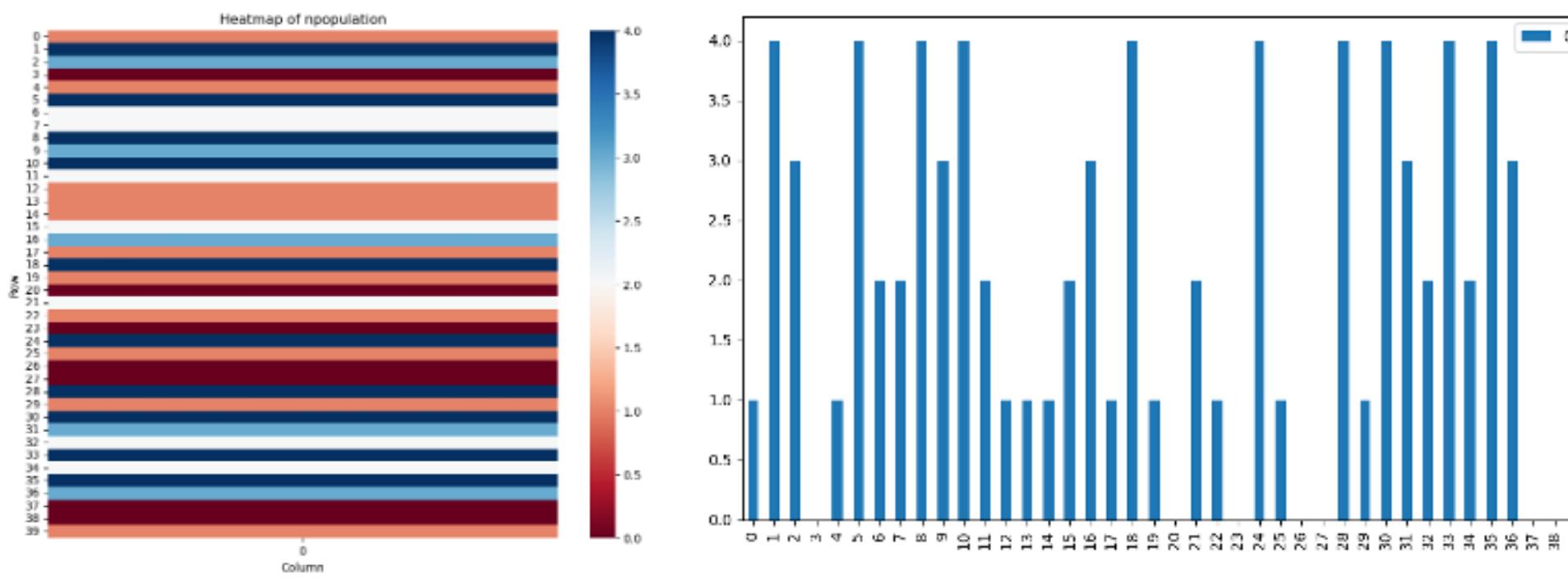
\* 특별배당: 추가배당이라고도 불리며, 정기적으로 지급되는 배당 시기에, 혹은 그 외의 별도의 시기에 지급하는 배당을 의미함. 기존 지급 배당에 비해 액수가 큰 편이고, 기업의 실적이 좋은 경우 추가 지급하는 경우임.

# 05 최적화 : 메타 휴리스틱

## 5.1 유전 알고리즘

목적식인 샤프지수가 공분산으로 나누어지는 형태로 선형식이 아니고, 이를 적절한 선형식으로 변환하여 풀이하기 어렵기 때문에 LP로는 해결하기 어렵다고 판단함. 해의 공간이 40개 주식종목의 비율을 정해야 하는 만큼 광범위한 해공간을 찾고자 했고, 수업시간에 배운 메타휴리스틱 방법 중 TS와 SA에비해 상대적으로 전역 최적해를 찾는데 유연성이 뛰어난 GA를 사용하여 해결해보기로 함.

메타휴리스틱을 사용하므로 40개 주식 종목의 비율은 정수라는 제약을 추가적으로 사용함.



유전 알고리즘에 사용된 염색체는 40개의 유전자로 각 주식 종목의 비율로 구성되어있음.

왼쪽은 50개의 초기 세대의 예시중 한개이며 초기해는 합이 100(%) 인 제약 조건아래 40개 항목에 랜덤한 값으로 채워져 있음.

# 05 최적화 : 메타 휴리스틱

## 제약식 및 유전, 돌연변이 구현

```
# 교차 지점 선택
crossover_point = np.random.randint(0, parent1.shape[0])

# 부모로부터 자식 생성
child = np.concatenate((parent1[:crossover_point], parent2[crossover_point:]), axis=0)

# 정규화: 합이 budget이 되도록
if np.sum(child) != 0:
    child = child / np.sum(child) * budget
else:
    child = child

# 돌연변이
if random.random() < mutation_rate:
    mutation_point = np.random.randint(0, parent1.shape[0])
    mutation_value = np.random.randint(1, budget // 40) # 돌연변이 값도 정수로 설정 # 돌연변이 1~10 일경우 망함 0.02 # 돌연변이 1~2 일경우 0.05까지 나옴
    child[mutation_point] = mutation_value
    if np.sum(child) != 0:
        child = child / np.sum(child) * budget
    else:
        child = child
```

1. 초기 50개의 인구에서 매 세대마다 랜덤포인트에서 교차시킨 후 20% 확률로 돌연변이를 발생시켜 자손을 만든다.

왼쪽의 파이썬 코드와 같이 자식 염색체를 만들 때 40개 중 랜덤한 교차지점을 한개 골라 부모세대에서 서로 교차된 자손 염색체를 만든다. 이 경우 자손염색체는 부모염색체와 달리 전체 합이 100% 가 되지 않는다는 문제점이 발생한다.

2. 이를 해결하기 위해 자손염색체를 100%로 만드는 정규화를 진행하였다.

자손을 정규화할 경우 부모가 거의 비슷한세대에서 나온 자식염색체는 변형정도가 적지만 부모가 매우 상이할경우 자식세대의 정규화로 인한 변형정도가 크다. 세대를 거듭할수록 부모세대가 일정한 optimal로 수렴한다는 가정하에 정규화를 진행해도 무방하다 판단하여 이를 추가하였고, 만들어진 자식세대 모두 합이 100(%)라는 제약조건을 만족한다.

3. 돌연변이의 경우 자손세대에서 20% 확률로 특정종목 값이 랜덤 값으로 변경되게 진행하였다. 마찬가지로 정규화로 제약조건을 만족시켰다.

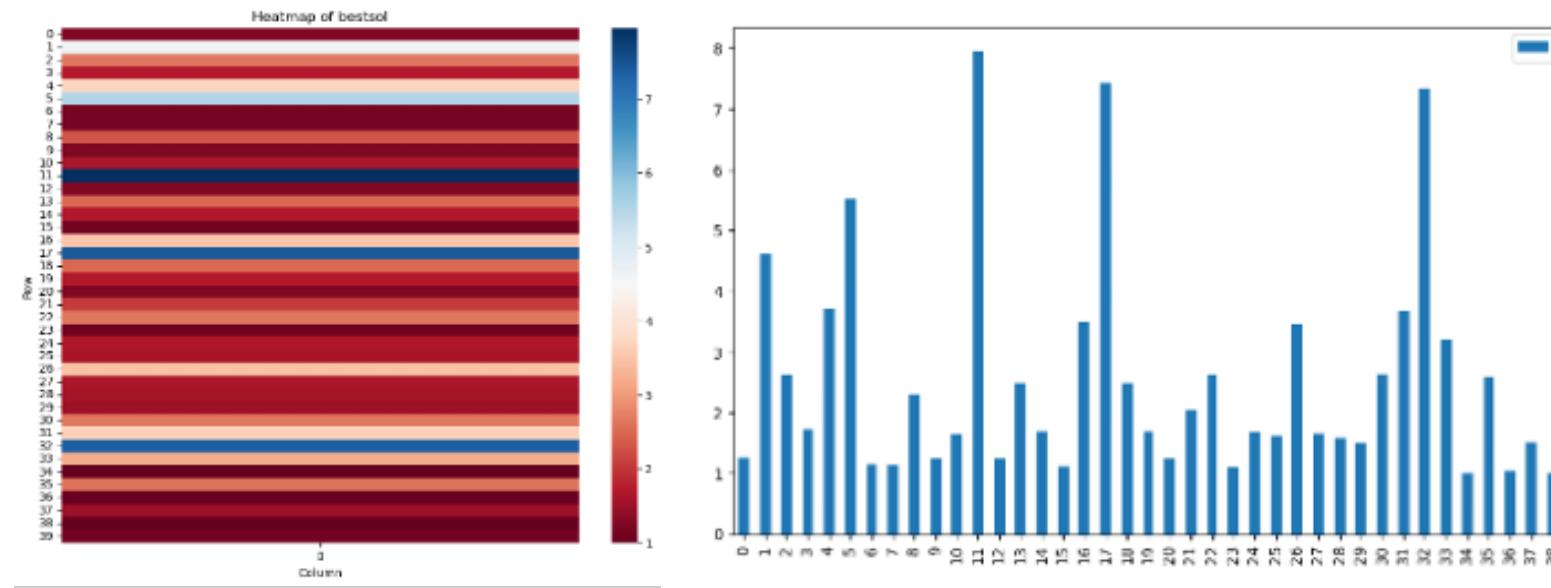
4. 최종 10000세대를 거치고 난 후(2분 소요)의 최고성능의 결과를 부모선택법에 따라 비교하였다.

# 05 최적화 : 메타 휴리스틱

## 부모 선택법에 따른 결과 비교

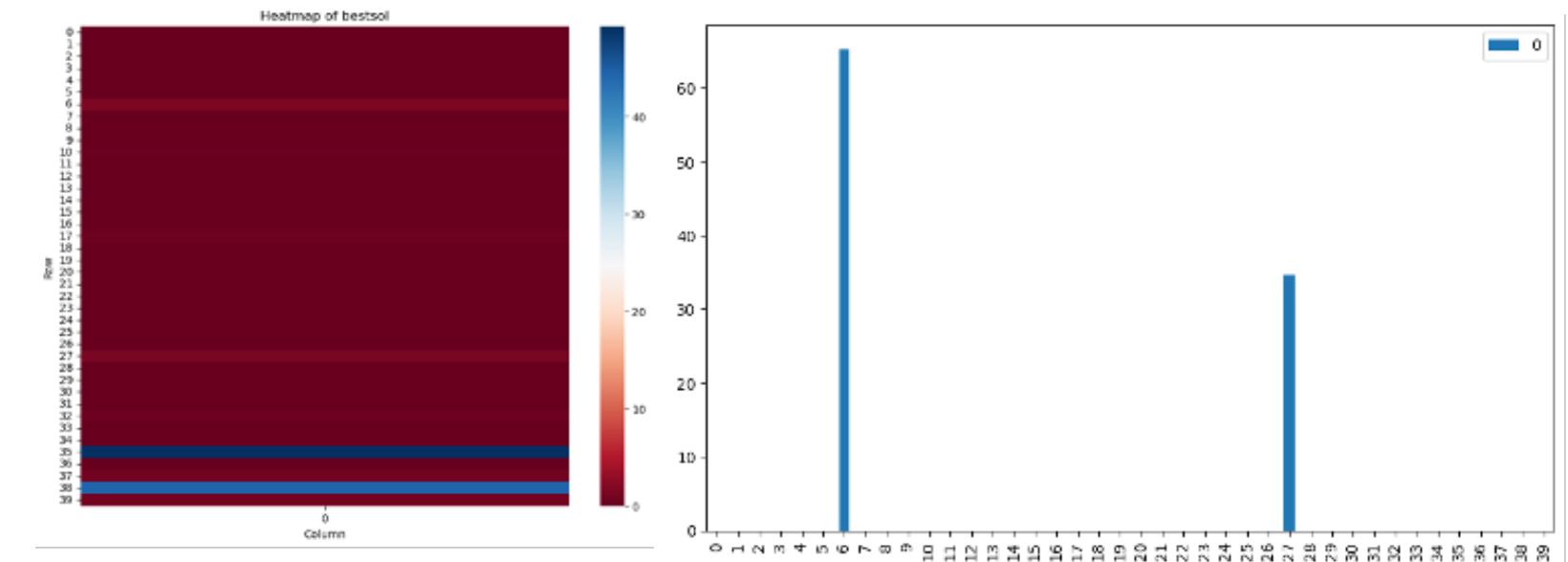
자손형성법과 돌연변이 방법을 동일하게 두고 10000세대를 거치고 난 후(각 약 2분 소요) 최고성능의 결과 도출.

### 5.1.1. 랜덤부모 선택



목적식 값이  $-0.17260188130208268$ 로 optimal한 값을 찾지 못함.  
10000세대 내에 랜덤한 부모선택 방법으로는 전체 인구를 모두 최적화시키지 못했다는 결론을 내고, 부모선택방법을 변화시켜보았음.

### 5.1.2. 우성부모 1 랜덤부모 1 선택

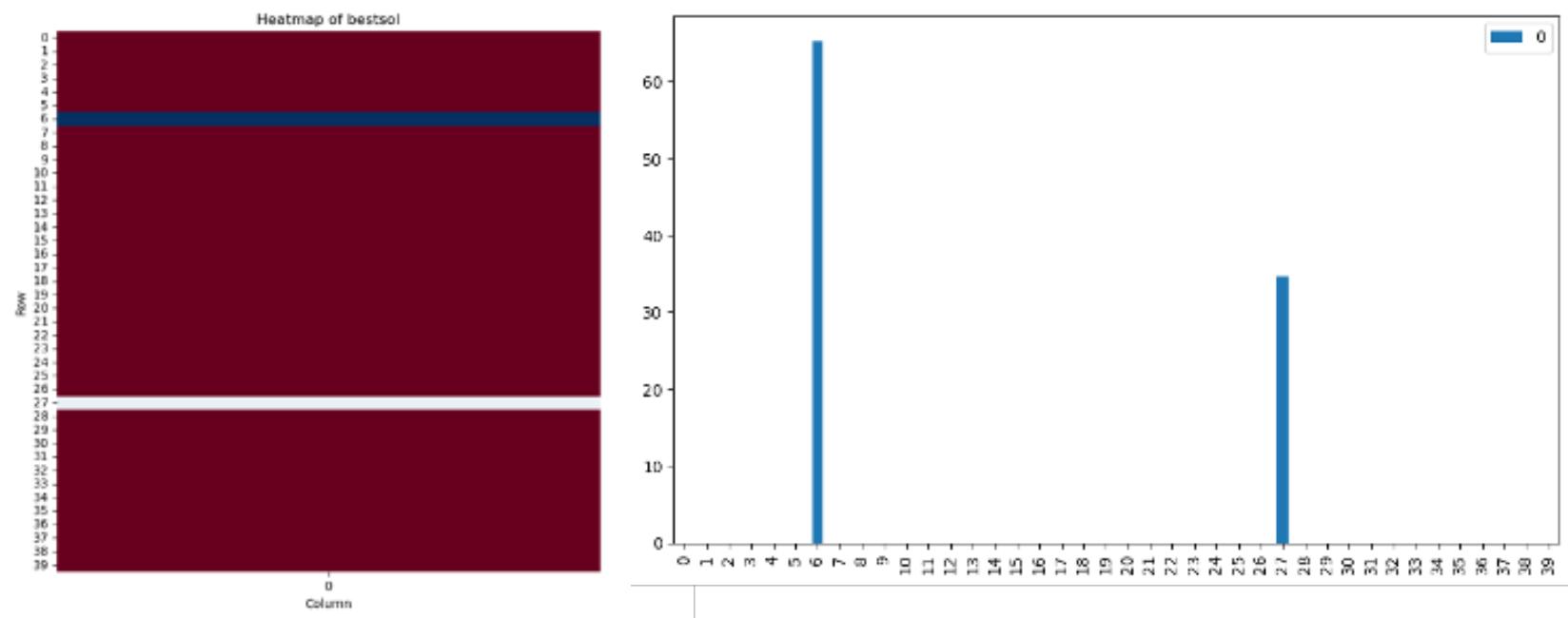


동일한 세대수와 돌연변이 확률에서 목적식 값이  $0.23617516200643723$ 로, 랜덤부모일때에 비해 해성능이 크게 좋아진 것을 확인함.

# 05 최적화 : 메타 휴리스틱

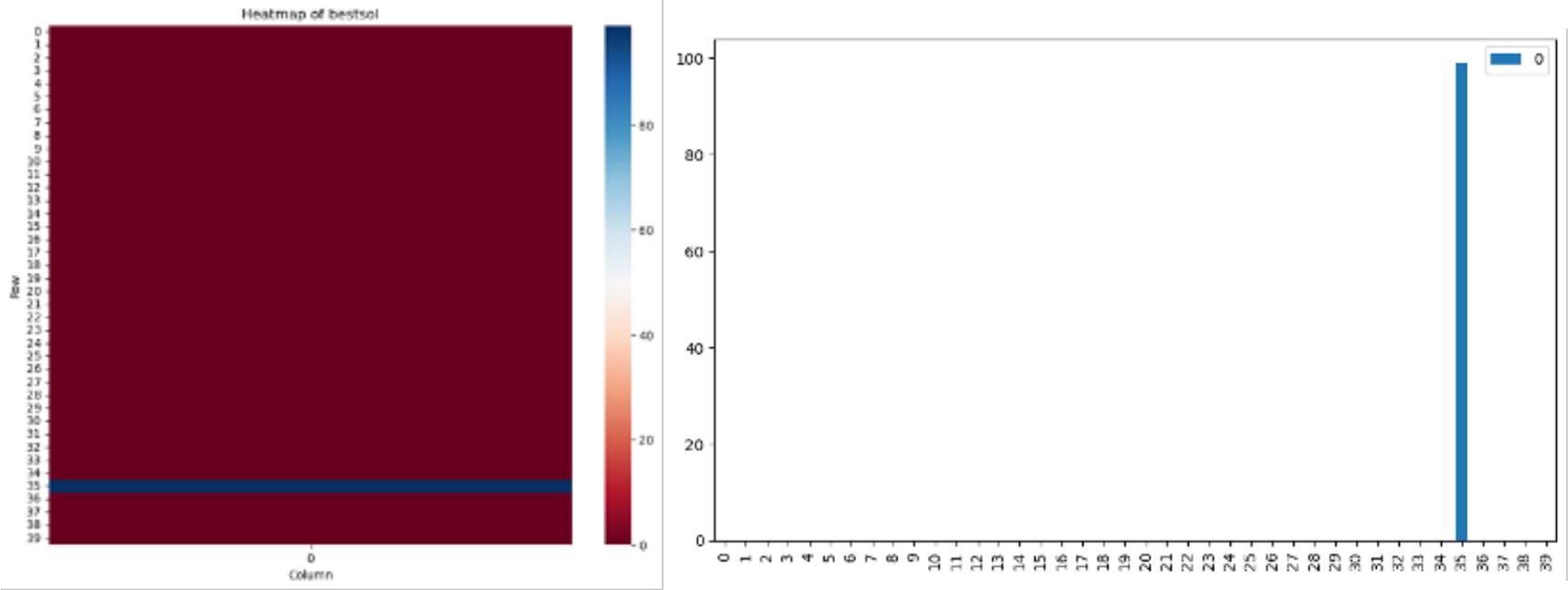
## 부모 선택법에 따른 결과 비교

### 5.1.3. 우성부모 2 선택



목적식 값이 0.21746086970300943로 우성부모1, 랜덤부모1에 비해 성능이 낮아짐.

### 5.1.4. Roulette wheel selection 으로 부모선택



초기해에 목적식의 값이 음수가 되는 문제가 발생함. 따라서 Roulette wheel selection의 부모선택 확률이 문제가 생기므로 다음과 같이 fitness score를 시그모이드 변환시켜서 진행함.

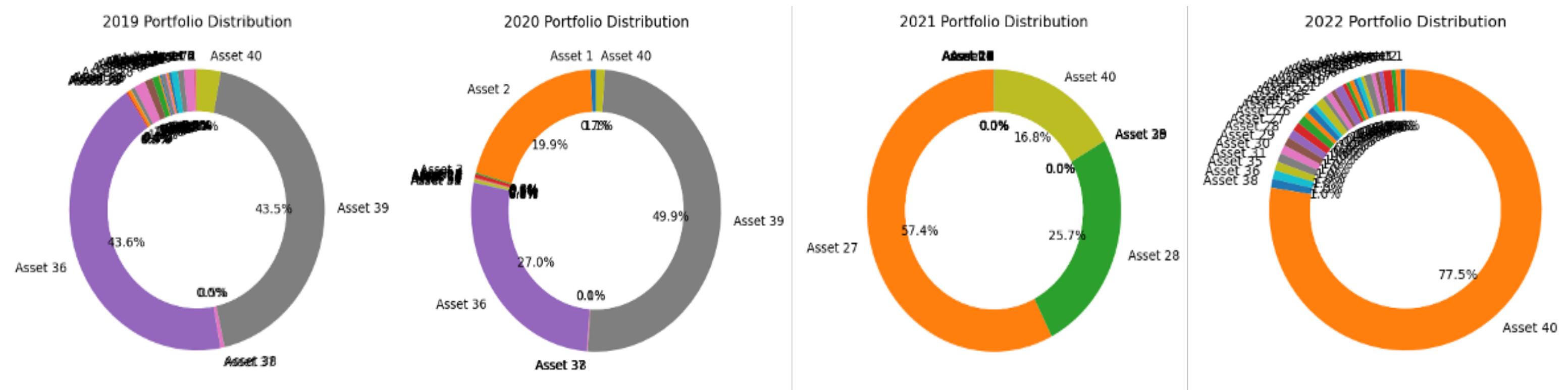
$$1/(1 + \exp(-(fitness\ scores)))$$

변환값에 로짓함수를 적용하여 계산한 결과 목적함수값이 0.20563047828648578로 도출됨.

# 05 최적화 : 메타 휴리스틱

# 연도별 결과 비교

4가지 방법중 가장 성능이 좋았던 5.1.2. 우성부모1,랜덤부모1 방법으로 2019부터 2022까지 연도별 각각 20000세대를 거친 유전 알고리즘 결과를 정리함.



*Sharpe Ratio* 0.21795751667760221

0.93962010398582

0.4508694082914086

0.1500442527358053

## 05 최적화 : SOLVER

### SOLVER 선택 이유

유전 알고리즘 결과

	2019	2020	2021	2022
Sharpe ratio	0.21796	0.93962	0.45087	0.15004

유전 알고리즘이 도출한 최적해가 Global Optimal이 아닐 수 있음

유전 알고리즘이 탐색하지 못한 공간에서 더 좋은 최적해가 있을 수도 있기 때문에  
NLP SOLVER를 활용하여 최적해를 구하고 결과를 비교하고자 함

cf. NLP SOLVER

SciPy  
라이브러리

SOLVER #1  
: Minimize function

SOLVER #2  
: Basin Hopping function

# 05 최적화 : SOLVER

## 해당 SOLVER 선택 이유

SciPy의 optimize 모듈은 비선형 최적화 문제를 해결할 수 있는 여러 알고리즘을 제공함. 목적함수가 비선형 구조로 이루어져 있기 때문에 비선형 최적화 문제를 해결하는데 유용한 SciPy Solver를 선택하였다.

### 1. `scipy.optimize.minimize`

`scipy.optimize.minimize` (optimize 모듈 내의 함수)

Minimize 함수 내에는 목적함수의 최솟값을 찾는 여러 종류의 알고리즘이 존재. 그 중 Sequential Least Squares Programming (SLSQP) 알고리즘은 제약 조건이 있는 비선형 최적화 문제를 해결하는 알고리즘.

각 의사결정변수의 범위가 0부터 1사이 제약이 있는 우리의 문제에 적용하기 적절하다고 판단함. “method = SLSQP”로 설정하여 SLSQP 알고리즘을 이용하여 최적해를 도출함.

Types of optimization problem	Function	Method
Local Opt. (국소)	unconstrained (비제약)	Nelder-Mead BFGS Newton-CG trust-nocg trust-krylov trust-exact
	constrained (제약)	minimize
Global Opt. (전역)	derivative-based (미분 적용)	trust-constr SLSQP
	metahuristic	basinhopping shgo brute(?) differential_evolution dual_annealing

# 05 최적화 : SOLVER

## 2. `scipy.optimize.basinhopping`

`scipy.optimize.basinhopping` (`optimize` 모듈 내의 함수)

전역 최적화 알고리즘. 즉, 로컬 최적화 알고리즘과 메타휴리스틱 방법을 결합하여 전역 최적해를 찾는 함수.

로컬 최적화 방법을 반복적으로 실행하면서, 해를 조금씩 변경하고, 일정 확률로 더 나쁜 해로 이동함으로써 전역 최적해를 찾아나가는 방식으로 수행됨.

`minimize` 함수는 지역 최적점에서 빠져나오지 못 할 수 있기 때문에 둘의 결과를 비교분석함으로써 최적의 결과를 도출.

Types of optimization problem	Function	Method
Local Opt. (국소)	unconstrained (비제약)	minimize
	constrained (제약)	trust-constr SLSQP
Global Opt. (전역)	derivative-based (미분 적용)	basinhopping shgo
	metahuristic	brute(?) differential_evolution dual_annealing

# 05 최적화 : SOLVER

## scipy.optimize.minimize 사용

SciPy 내에서 사용하는 문법에 따라 의사결정변수, 경계 조건(제약식), 목적함수 식(sharpe ratio function)을 새롭게 정의함.  
minimize 내의 알고리즘 중 SLSQP 알고리즘을 사용할 것이기 때문에 method = "SLSQP"로 설정.

```
# 초기 가중치 설정
init_weights = np.ones(40) / 40

# 제약 조건 정의
# 각 의사결정변수의 합이 1이 되도록 하는 제약
constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})

# 경계 조건 설정
# 각 의사결정변수가 0과 1 사이에 있도록 설정
bounds = tuple((0, 1) for _ in range(40))

# 최적화 문제 해결
result = minimize(sharpe_ratio, init_weights, method='SLSQP', bounds=bounds, constraints=constraints)
```

# 05 최적화 : SOLVER

## scipy.optimize.minimize 사용

SciPy의 minimize는 함수의 최솟값을 찾아주기 때문에, 목적함수 값인 샤프지수의 부호를 반대로 처리해주었음.  
따라서 최소의 목적함수 값을 갖는 Solution이 최대의 샤프지수 값을 갖는 Solution.

```
def sharpe_ratio(weights):
    portfolio_returns = returns @ weights

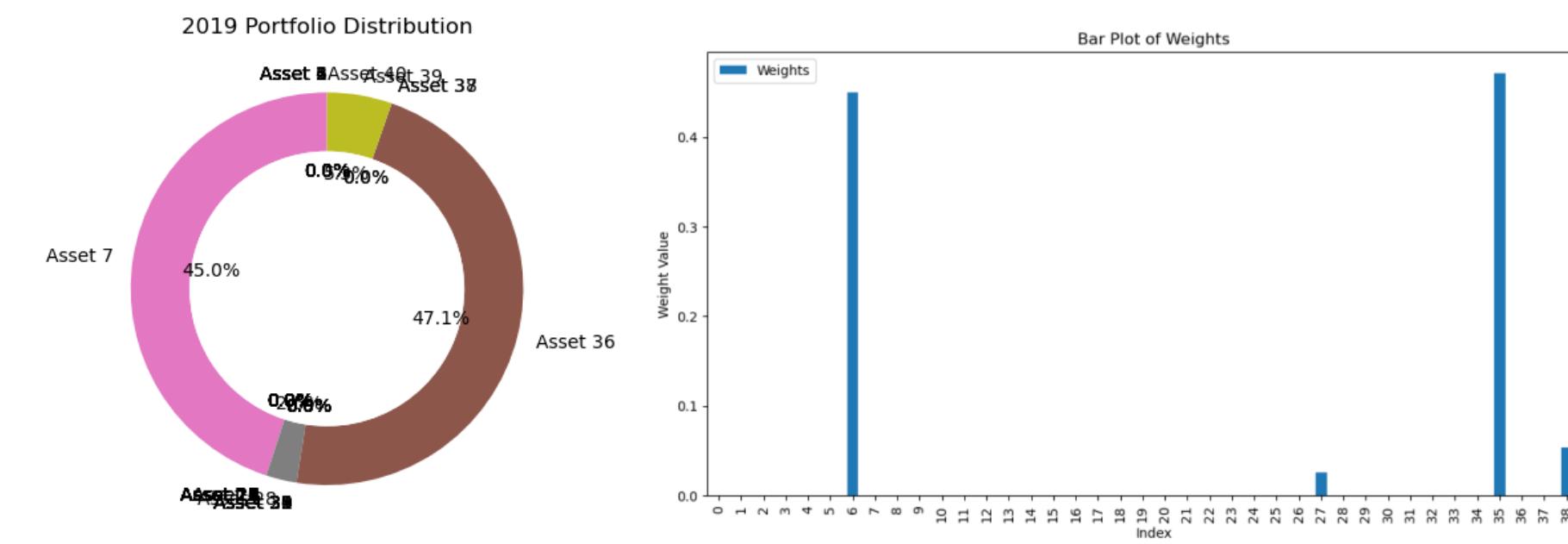
    result_df = pd.DataFrame({
        'Date': df['Date'],
        'R_p': portfolio_returns
    })
    R_p = result_df['R_p'][1]

    sigma_p = np.sqrt(np.dot(weights.T, np.dot(cov_2020, weights)))
    sharpe = (R_p - 0.01) / sigma_p
    return -sharpe
```

# 05 최적화 : SOLVER

# SOLVER #1. Minimize Function

< 2019년 포트폴리오 >

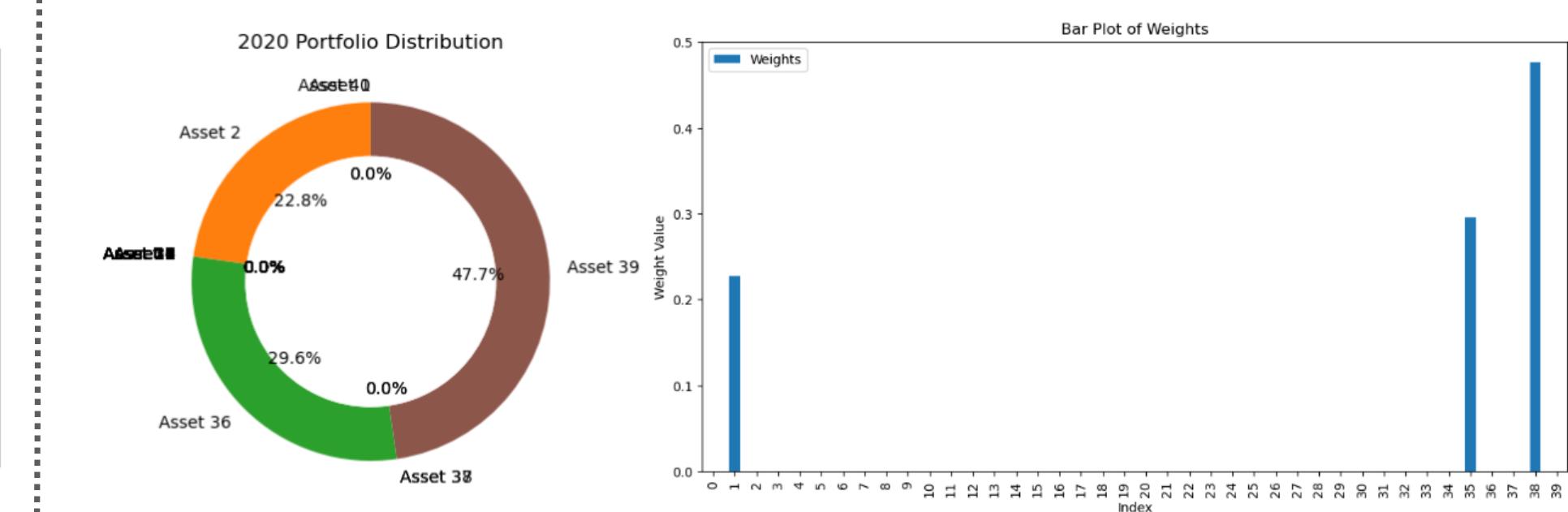


## 〔 4개의 종목으로 구성된 포트폴리오 〕

- 7, 36번 종목의 비중이 대다수를 차지함
  - 27, 38번 종목은 얼마 포함되지 않음

Optimal Sharpe Ratio = 0.27017

<2020년 포트폴리오>



[ 3개의 종목으로 구성된 포트폴리오 ]

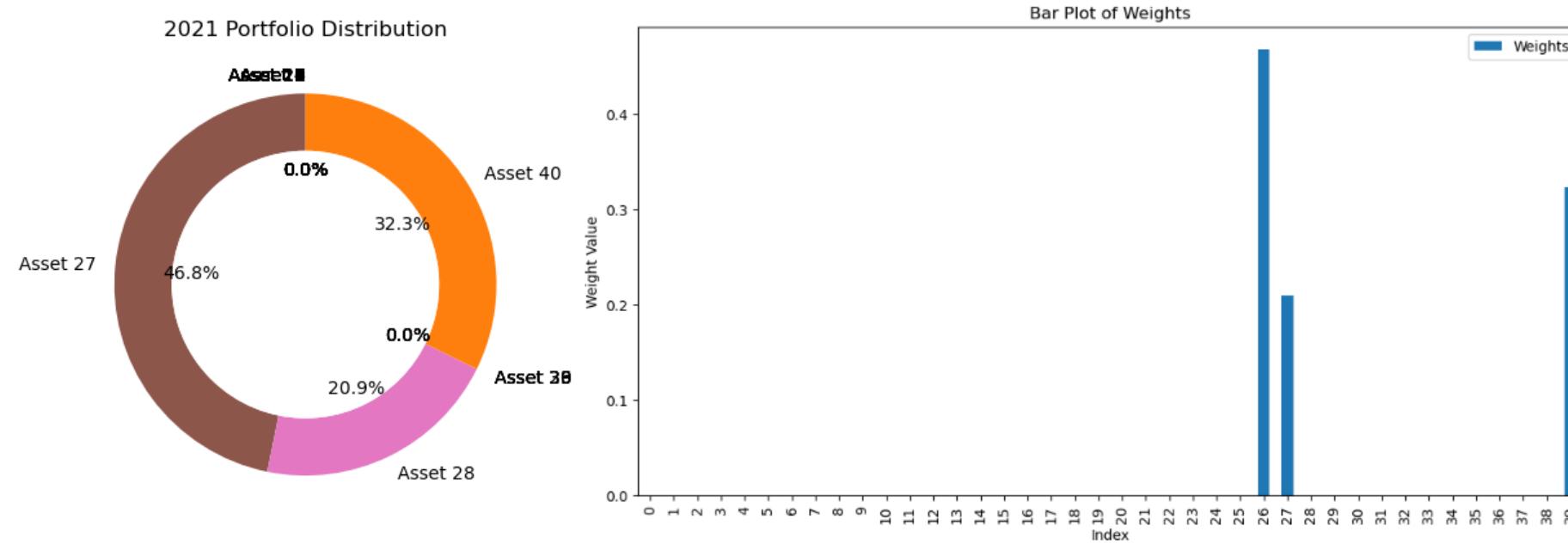
- 2번 < 36번 < 39번 종목 순으로 많은 비중을 차지함

Optimal Sharpe Ratio = 0.95244

# 05 최적화 : SOLVER

## SOLVER #1. Minimize Function

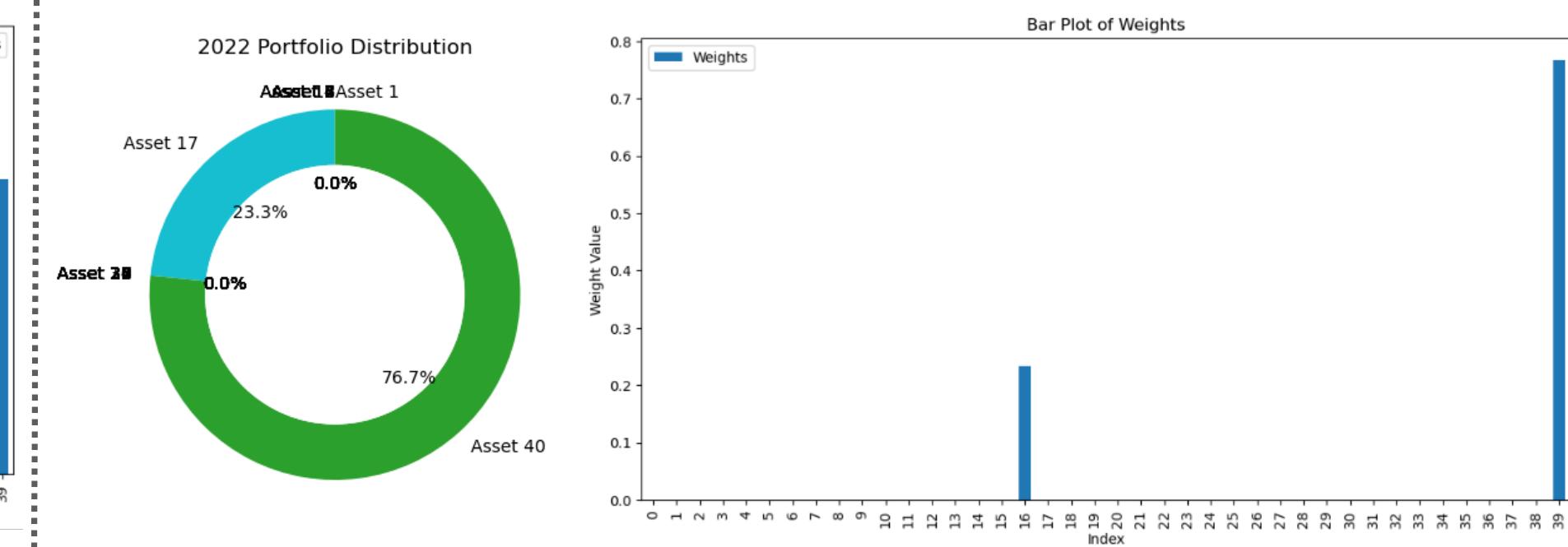
< 2021년 포트폴리오 >



[ 3개의 종목으로 구성된 포트폴리오 ]  
- 28번 < 40번 < 27번 종목 순으로 많은 비중을 차지함

Optimal Sharpe Ratio = 0.45105

< 2022년 포트폴리오 >



[ 2개의 종목으로 구성된 포트폴리오 ]  
- 40번 종목이 포트폴리오의 3/4을 차지하며, 17번 종목은 1/4을 차지함

Optimal Sharpe Ratio = 0.19830

# 05 최적화 : SOLVER

## scipy.optimize.basinhopping 사용

basinhopping에서의 목적함수, 의사결정변수, 제약 조건은 scipy.optimize.minimize에서의 것들과 동일함.  
코드 구현에 있어서 달라지는 부분은 result 정의 부분으로, basinhopping의 문법에 맞게 작성하여 최적화를 진행함.

```
# 경계 조건 설정
bounds = [(0, 1) for _ in range(40)]

# Define the minimization function with bounds and constraints
minimizer_kwargs = {
    'method': 'SLSQP',
    'bounds': bounds,
    'constraints': constraints
}

# Perform Basin-Hopping optimization
result = basinhopping(sharpe_ratio, init_weights, minimizer_kwargs=minimizer_kwargs, niter=200)
```

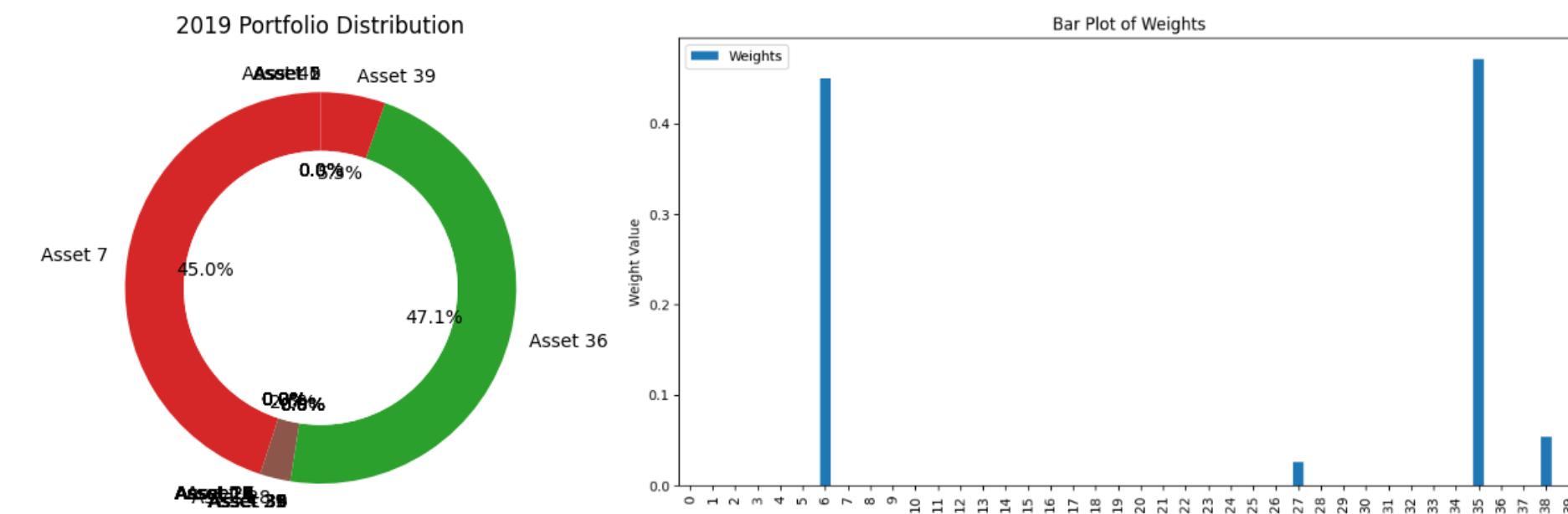
Basinhopping 역시 SLSQP 알고리즘을 활용하여 최적해를 탐색해 나가도록 설정함.

Basichopping은 로컬 최적화 방법을 반복적으로 실행하면서, 해를 조금씩 변경하고, 일정 확률로 더 나쁜 해로 이동함으로써 전역 최적해를 찾아나간다는 점에서 Minimize funtion과 차이가 있음.

# 05 최적화 : SOLVER

## SOLVER #2. Basin Hopping Function

< 2019년 포트폴리오 >

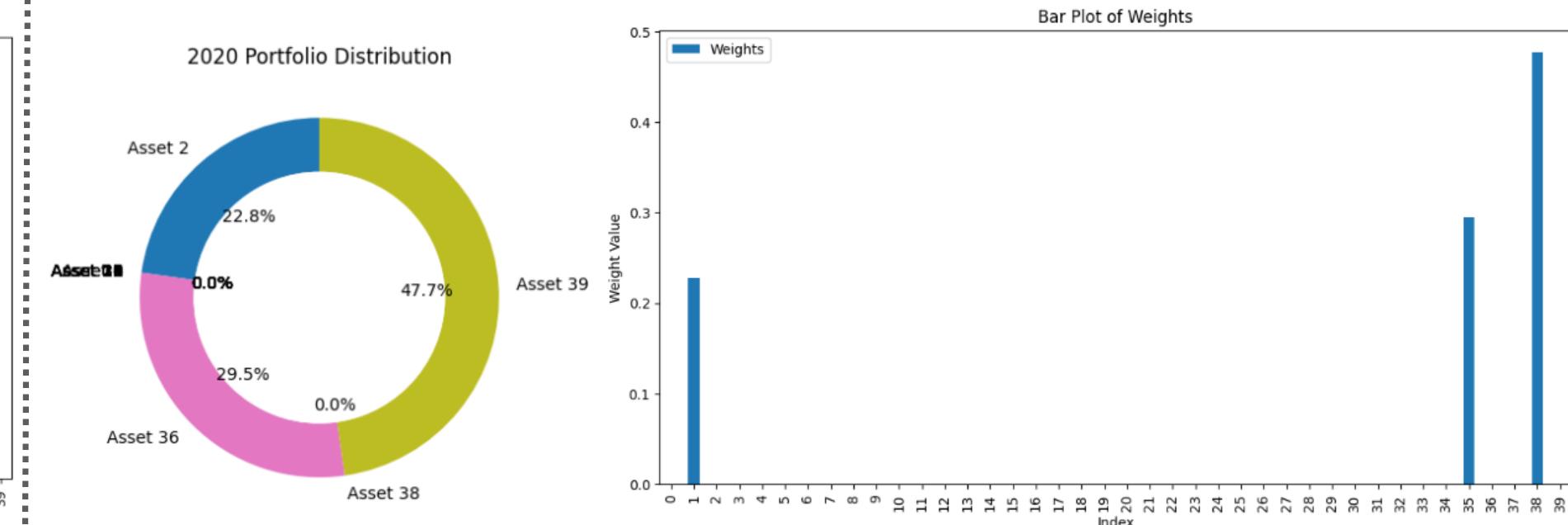


[ 4개의 종목으로 구성된 포트폴리오 ]

- 7, 36번 종목의 비중이 대다수를 차지함
- 27, 38번 종목은 얼마 포함되지 않음

Optimal Sharpe Ratio = 0.27018

< 2020년 포트폴리오 >



[ 3개의 종목으로 구성된 포트폴리오 ]

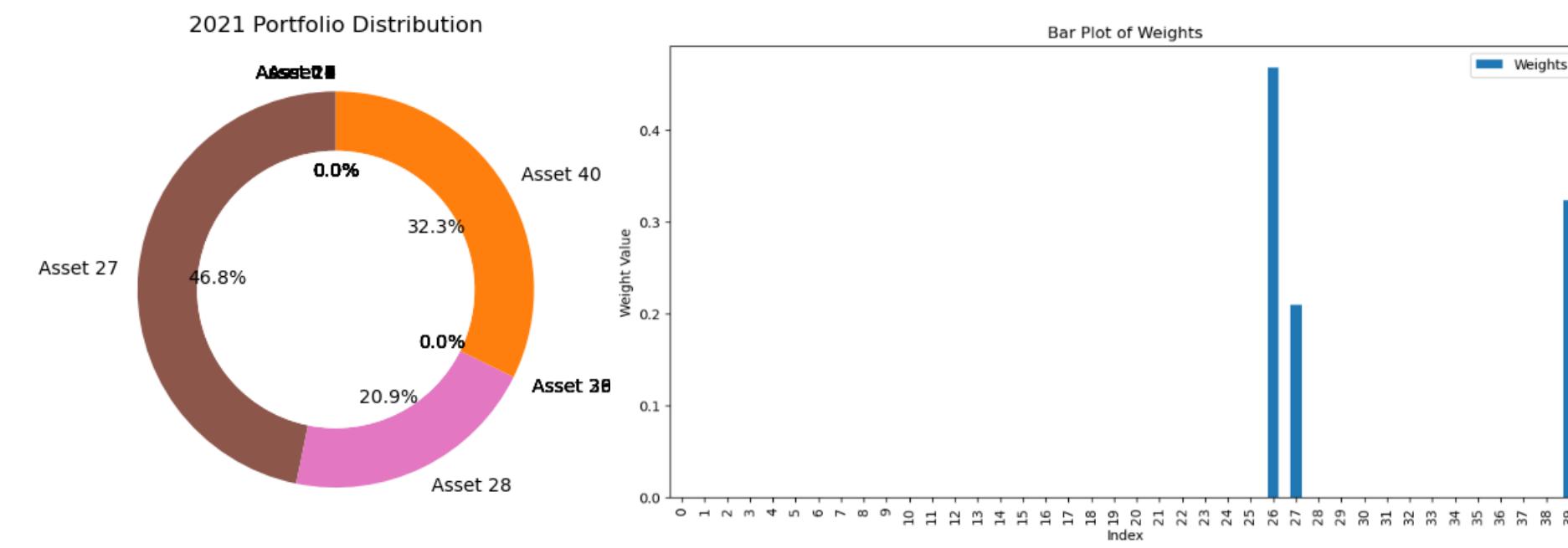
- 2번 < 36번 < 39번 종목 순으로 많은 비중을 차지함

Optimal Sharpe Ratio = 0.95244

# 05 최적화 : SOLVER

## SOLVER #2. Basin Hopping Function

< 2021년 포트폴리오 >

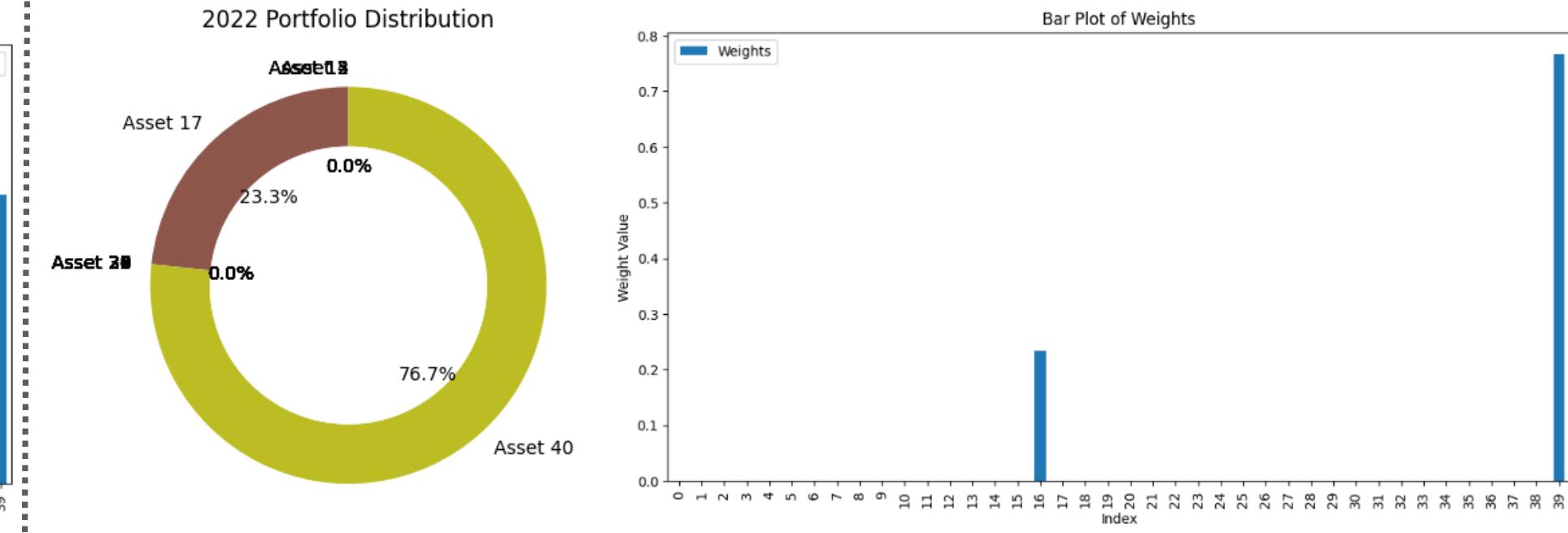


[ 3개의 종목으로 구성된 포트폴리오 ]

- 28번 < 40번 < 27번 종목 순으로 많은 비중을 차지함

Optimal Sharpe Ratio = 0.45105

< 2022년 포트폴리오 >



[ 2개의 종목으로 구성된 포트폴리오 ]

- 40번 종목이 포트폴리오의 3/4을 차지하며, 17번 종목은 1/4을 차지함

Optimal Sharpe Ratio = 0.19830

## 05 최적화 : SOLVER

---

### 각 Function별 최적해 비교

---

SciPy 라이브러리에 구현되어 있는 Minimize, Basin Hopping 함수를 각각 사용하여 결과를 비교하고자 함.



Minimize Function은 Local Optimal, Basin Hopping Function은 Global Optimal을 구하는 데에 초점이 맞추어져있는 함수이기에, Basin Hopping 함수의 성능이 더 좋을 것이라 예상했음.



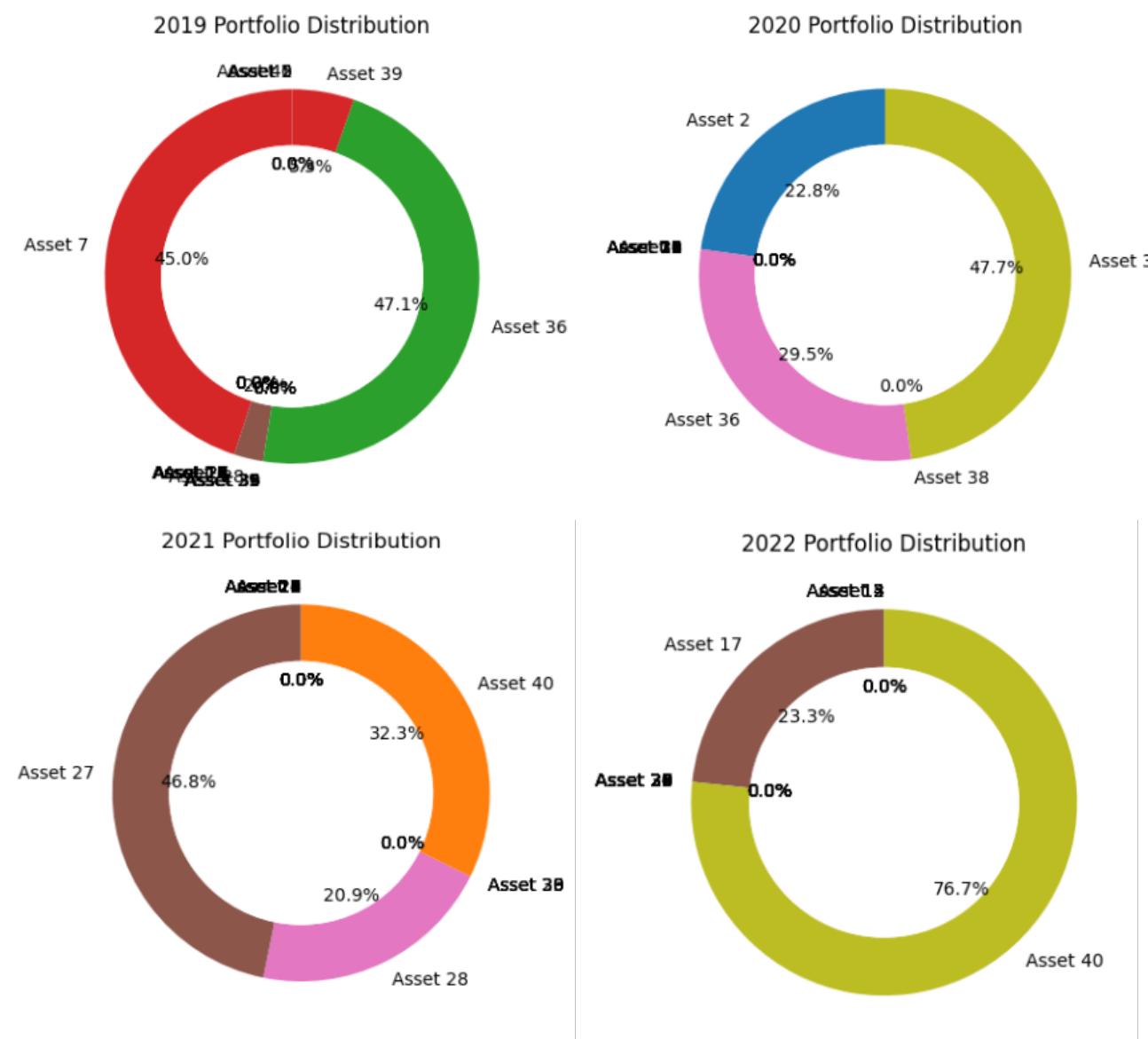
그러나 실제 코드 실행 결과, Basin Hopping 함수가 소수점 5자리 이상에서 더 높은 sharpe ratio값을 나타내긴 했지만, 이는 상대적으로 미세한 차이이므로 무시하기로 했음.



즉, 두 함수를 사용하여 찾아낸 Optimal Solution( $w_i$ ), Optimal Objective value(Sharpe Ratio)가 동일함을 알 수 있었음.

# 06 결론

# Project의 Optimal Solution



샤프비율을 도입함으로써 단순히 수익률이 좋은 주식을 Sorting하는 것이 아닌 위험성과 수익성을 함께 반영한 포트폴리오가 도출된 것을 확인할 수 있으며, 유전 알고리즘과 Solver를 이용하여 NLP 비선형 최적화 문제를 해결했다.

프로젝트 진행 결과, SciPy라이브러리의 basinhopping Solver를 사용한 포트폴리오가 가장 최적임을 알 수 있었다.

- GA를 사용한 결과, 다양한 종목이 알고리즘에서 쉽게 탈락되지 않고 낮은 비율로 포트폴리오에 담겼다. 이들은 주요 종목들에 비해 아주 낮은 비율로 담겼기 때문에, 포트폴리오의 다각화 측면보다는 목적함수 값(샤프비율)의 감소에 더 많은 영향을 끼쳤다.

- Solver를 사용한 결과, GA에서 발생한 위와 같은 문제들을 해결하며 목적함수 값(샤프비율)을 개선할 수 있었다.

## 06 결론

---

### 인사이트

#### 1. 본 최적화 과정을 통해 시장리스크를 헤지할 수 있었는가.

일반적으로 샤프지수는 0.5 이상인 경우 양호한 포트폴리오, 1.0 이상인 경우 좋은 포트폴리오라고 평가하는 지표로서 활용된다. 본 프로젝트에서 최적화 과정을 통해 만든 포트폴리오는, 다소 낮은 범위(0.198 ~ 0.952)의 샤프지수를 도출했다.

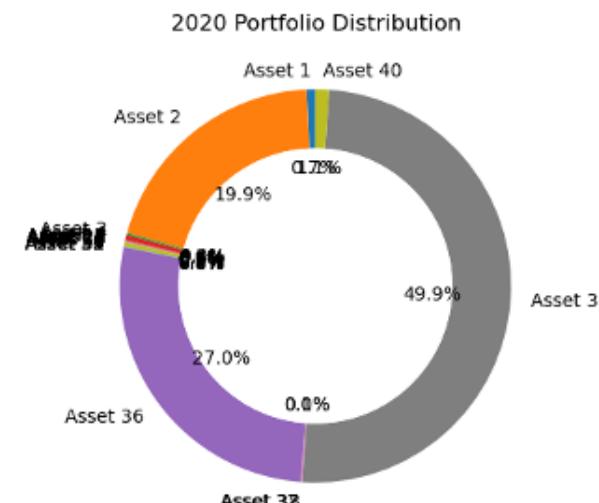
- 샤프지수 0.198은 통상적으로 높지 않은 수치에 해당하지만, 본 프로젝트에서는 투자 유니버스를 '귀족배당주', 즉 안정성이 매우 높고 배당 지급이 우선시 되는 주식만을 대상으로 선정했기 때문에 0.198이라는 수치 역시 낮은 수치가 아니며, 오히려 귀족배당주만을 활용하여 도출해낸 샤프지수임을 감안했을 때는 양호한 수치로 보인다.
- 2020년의 샤프지수 0.952는 매우 우수한 수치다. 귀족배당주가 아닌 일반 자산군을 대상으로 도출해도 쉽게 나오지 않는 수치에 해당한다. 다만 유전 알고리즘과 솔버 모두에서 2020년의 샤프지수가 가장 높게 도출됨을 확인할 수 있다. 이는 샤프지수가 위험 대비 수익률이라는 점을 고려했을 때, 코로나로 인해 위험도가 높았던 당시 시장 상황에서, 배당금을 바탕으로 안정성 있는 수익을 일궈내는 포트폴리오임을 만들어냈음을 검증할 수 있는 수치다.

따라서 배당주를 중심으로 최적화한 본 프로젝트의 포트폴리오는, 시장의 위험 국면에서도 안정적인 수익을 창출해낼 수 있는 헤지 방안으로서 활용될 가능성이 충분하다.

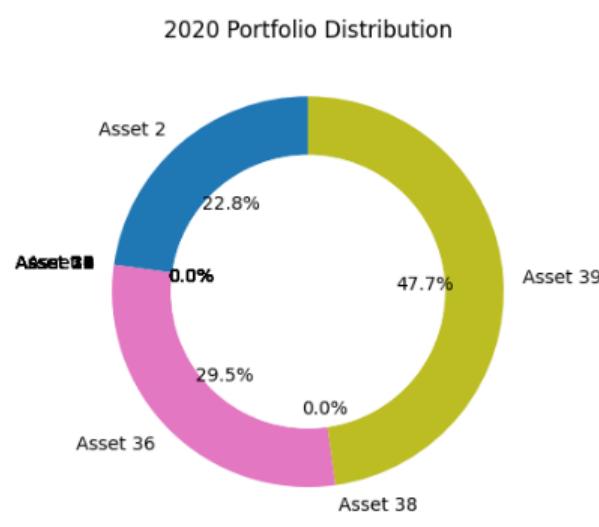
# 06 결론

## 인사이트

### 2. 시장리스크 헤지를 위한 샤프지수 최적 포트폴리오에 메타휴리스틱과 SOLVER 중 어느것이 더 적합한가.



GA를 이용한 2020의 최적 포트폴리오



SOLVER를 이용한 2020의 최적 포트폴리오

옆의 포트폴리오는 각각 샤프지수 0.93, 0.95로 모두 우수한 포트폴리오이다. 두 포트폴리오 모두 세가지 주요 종목에 대한 투자비율은 비슷하나 유전알고리즘의 경우 1퍼센트 미만 비율을 차지하는 종목의 수가 SOLVER를 활용한 결과보다 몹시 많았다. 이는 유전알고리즘이 10000세대 내에 0이하의 적은 비율의 종목을 아예 0으로 만드는 작업을 잘 진행하지 못하는 것으로 보인다. 해당 종목들은 해의 성능이 낮게 나온 원인 중 하나로 보이며 이러한 문제점으로 인해 정확한 비율을 찾는 작업은 SOLVER가 더 좋아보이며, GA는 오히려 초기에 많은 집단이 있을 경우 우수한 종목과 비율을 추려낼 수 있는 방법으로 보인다.

현재의 유전 알고리즘은 4가지의 부모 선택 방법을 도입하여, 그 중 최고의 성능을 발휘하는 방법을 채택하였다. 부모 선택뿐만 아니라 교배와 돌연변이 생성에서도 다양한 방법들이 존재한다. 기존의 GA보다 더 다양한 해공간을 탐색해나가길 원한다면 다양한 교배 전략과 돌연변이 생성 기법을 고려해 볼 수 있을 것이다. 또한 Solver의 특성상 문제의 수가 커질수록 NP-Hard 문제의 특성으로 인하여 Solver가 최적해를 보장하기 어려워진다. 주식 종목의 수가 40개인 현재의 상황에서는 Solver가 최적해를 잘 도출하고 있지만 종목의 수가 급격하게 커지는 경우에는 메타휴리스틱을 적용하는 것이 더 적합할 것으로 보인다.

# 06 결론

## 실제 시장과의 차이점과 한계

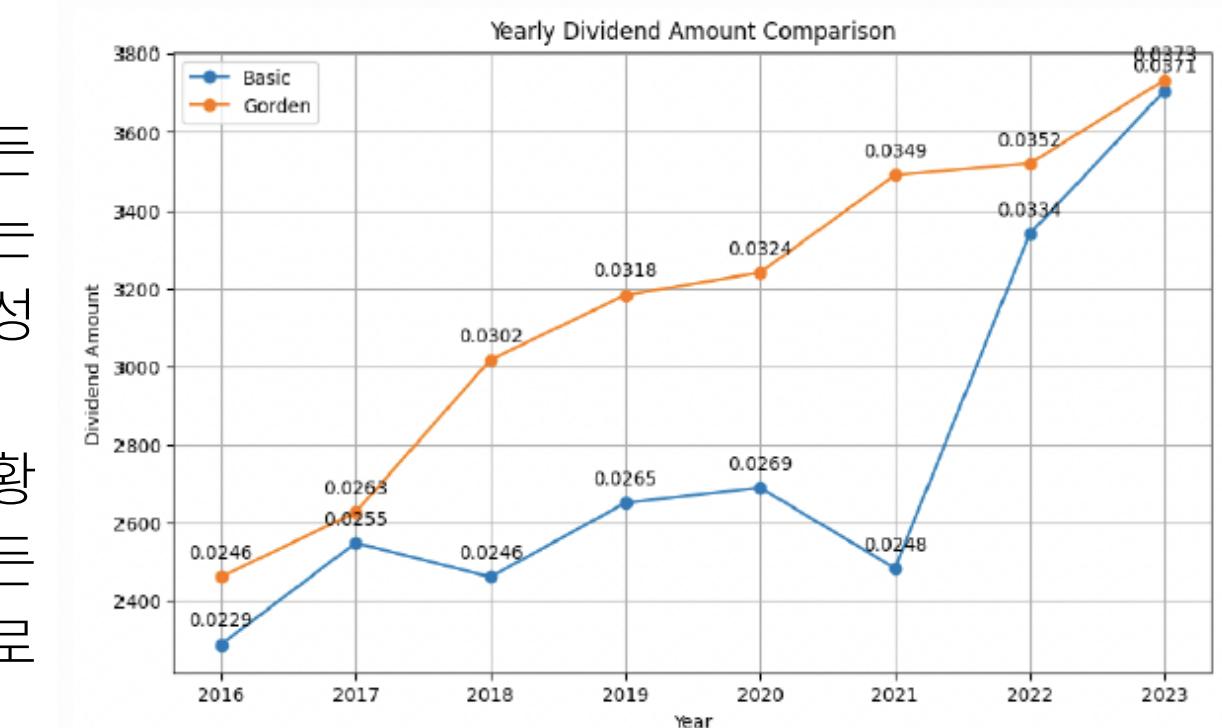
### 1. 실제 거래 상황 반영 미흡

본 프로젝트에서는 배당금과 주가 수익률, 액면분할 비율 등의 데이터를 실제 과거 거래 데이터로 활용했기 때문에 실제 거래 상황을 매우 잘 반영했다고 볼 수 있다. 하지만 특정 부분에서 현실을 전부 반영하는 데엔 어려움이 있었는데, 이를테면 실제 주식 거래시에 적용되는 거래세, 미국 주식 거래시 시기마다 달라지는 환율을 반영하지 못했다. 프로그래밍에서 시계열 상황을 반영하는 데에 어려움을 겪어 역량 상의 이유로 구현이 어려웠지만, 차후 본 프로젝트를 더 디벨롭시킬 기회가 있다면 충분히 시도하여 개선할 수 있을 것으로 보인다.

### 2. 고든성장모형 적용의 타당성 문제

샤프지수에 활용되는 기대수익률을 보다 정밀하게 도출해내고자 고든성장모형을 적용했다. 고든성장모형의 적용을 위해 '배당성장률의 연간 변화율'과 '주가 연간 변화율'의 흐름이 일치한다는 점을 전제했어야 했다. 하지만 8페이지에 첨부된 두 그래프를 보면, 시간에 따른 두 지표의 경향성이 비슷하긴 하지만 완전히 일치하지 않았다.

하지만 우측 그래프에서 확인할 수 있듯이 '고든성장모형을 적용한 샤프지수 최대화 방법(주황색)'과 '일반 샤프지수 최대화 방법(파란색)' 각각에서 산출되는 배당금 액수를 비교했을 때, 고든성장모형을 적용한 경우의 수치가 일반적으로 상회하는 경향성이 있음을 확인했고, 이를 근거로 고든성장모형을 적용할 수 있었다.



# References

---

[1] 유전 알고리즘 개요 & 파이썬으로 예제 구현해보기. (2020, 01 05). MargmaTart blog.  
<https://magmatart.dev/development/2020/01/05/Python-Genetic-Example.html>

[2] 파이썬 scipy 이용한 최적화 (Optimization) 예시. (2021, 05 31). 라이징N폴링.  
[https://blog.naver.com/rising\\_n\\_falling/222361251972](https://blog.naver.com/rising_n_falling/222361251972)

[3] 주식 가치 평가 (고든성장모형). (2024, 03.06). FasterCapital.  
<https://fastercapital.com/ko/content/주식-가치-평가--고든-성장-모델의-비결.html>

[4] Dividend Aristocrats List 2024. (2024, 04.02). MarketBeat.  
<https://www.marketbeat.com/dividends/aristocrats/>

[5] Understanding the Sharpe Ratio. (2024, 01.30). Investopedia.  
[https://www.investopedia.com/articles/07/sharpe\\_ratio.asp](https://www.investopedia.com/articles/07/sharpe_ratio.asp)

감사합니다.