Using MOEA/D Alongside Lévy Flight to Solve Portfolio Optimization Problems

Shankara Abbineni SHANKARA.ABBINENI@YALE.EDU
Department of Statistics & Data Science
Yale University
New Haven, CT, USA

Editor: Not Published

Abstract

This is a project of **Type 2:** Understanding Specific Optimization Problems or Methods.

1 Background and Problem

Portfolio optimization is an extremely important financial task that seeks to best allocate capital amongst a set of financial assets to achieve the best, suitable tradeoff between the risk of the investment and its return. One method used to solve portfolio optimization problems is through the use of multi-objective evolutionary algorithms (MOEAs), which have a natural bi-objective structure. Evolutionary algorithms use a population-based approach where more than one solution is considered in an iteration to create a new population of solutions for each new iteration (Deb, 2011). As a result, a MOEA is capable of obtaining several Pareto efficient solutions within a single run of the algorithm (Coello et al., 2021). One of the main challenges of portfolio optimization is the high-dimensionality of the search space. The paper "Solving Portfolio Optimization Problems Using MOEA/D and Levy Flight" proposes a solution to this challenge by injecting a distribution-based mutation method named Lévy Flight into a decomposition based MOEA named MOEA/D (He and Aranha, 2020). Lévy Flight is a special random walk where the step-length is obtained from a heavy-tailed distribution (He and Aranha, 2020). This allows the algorithm to escape the local optimal and improves the overall performance in the optimization task. By integrating Lévy Flight into the MOEA implementation, the optimization algorithm is able to leverage the efficient global search performance of Lévy Flight (He and Aranha, 2020). As a result, this new method of portfolio optimization should allow for an optimal solution to be found quicker and with fewer iterations. In this paper, we explore the use of Lévy Flight and MOEA/D to optimize a financial portfolio.

The task of portfolio optimization involves calculating an optimal allocation of capital resources between various assets (and asset classes) in order to obtain the best trade-off between investment risk and return. In a simple portfolio optimization problem, we seek to maximize an objective (such as return) while simultaneously minimizing a second objective (such as risk). In most portfolio optimization models, all of the available capital is invested; as such, the weight of each asset in the portfolio (the fraction of the total capital placed in that asset) is represented with the term $\mathbf{w}_i$, for the i-th asset. Furthermore, if we allow r_i

to be the return of the i-th asset and σ_{ij} to be the covariance between the returns of the i-th asset and j-th asset, we can formulate the simple portfolio optimization problem

$$\max_{\mathbf{w}} \quad \text{Return} = \sum_{i=1}^{n} r_i \mathbf{w}_i \tag{1}$$

$$\min_{\mathbf{w}} \quad \text{Risk} = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} \mathbf{w}_i \mathbf{w}_j \tag{2}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \mathbf{w}_i = 1, (0 \leq \mathbf{w}_i \leq 1) \tag{3}$$

From the simple portfolio optimization problem, notice that the task of portfolio optimization inherently belongs to the category of multi-objective optimization problems (MOOPs). MOOPs are used to find an optimal set of solutions for a trade-off between multiple competing objectives (such as return and risk).

The main method to solve MOOPs is through the use of multi-objective evolutionary algorithms (MOEAs) (Huang et al., 2019). MOEA models are diverse with respect to the range of strategies and implementations. A MOEA such as NSGA-II (Deb et al., 2002) obtains a set of solutions through the use of domination; two solutions are considered non-dominates if one solution is better in at least one objective than the other solution. Another variation of MOEAs is MOEA/D, which produces the optimal feasible solutions by deconstructing a multi-objective problem into various single-objective optimization problems and optimizing all of these smaller problems simultaneously (Zhang and Li, 2007). In economics, Pareto efficiency is defined as being the allocation wherein the resources are in such a way that they achieve the maximum level of efficiency, and no change can be made without being worse-off. Similarly, in MOOP problems, we define a Pareto front as an optimal solution set that contains all of the feasible solutions that are not dominated by any other solutions in the problem's feasible region. The MOEA/D obtains the Pareto front by applying a decomposition method on the multi-objective problem to form various single-objective problems. A possible method for the decomposition procedure is the Tchebycheff approach. The Tchebycheff approach utilizes a weight vector λ and a reference point z^* (where z^*_m is the minimum value of the m-th objective) to convert a vector optimization problem into a scalar optimization problem as:

$$\min_{x} \quad g^{te}(x \mid \lambda, z^* = \max_{m=1,\ldots,M} \{\lambda_m \mid f_m(x) - z^*_m\} \tag{4}$$

$$\lambda = (\lambda_1, \ldots, \lambda M), \lambda_1 + \ldots + \lambda_M = 1 \tag{5}$$

$$z^* = (\min f_1, \ldots, \min f_M) \tag{6}$$

In the original MOEA/D algorithm, each individual in the population is optimized to be the solution of one SOOP using one weight vector (He and Aranha, 2020). In the optimization process, the neighbors — which are defined as the solutions of neighboring SOOPs — provide information to guide the solution. An offspring is produced in MOEA/D by selecting parents from the neighbors of the individual in question. After generating the offspring, the neighbors of the current individual can be updated. The **Algorithm 1** block demonstrates what the original MOEA/D algorithm consists of.

Algorithm 1 Original MOEA/D

```
1: Generate a set of weight vectors $\{\lambda_1, \ldots, \lambda_N\}$
2: Determine neighbors
3: Initialize the population as $\{x_1, \ldots, x_N\}$
4: Compute the reference point $z^*$
5: while stopping criteria do
6:     for $x_i$ in population:
7:         Select the parents from the neighbors of $x_i$
8:         Reproduce an offspring $y$ by GA operator
9:         Update the reference point $z^*$
10:        for $x_p$ in neighbor of $x_i$:
11:            if $g^{te}(y \mid \lambda_p, z^*) \leq g^{te}(x_p \mid \lambda_p, z^*)$:
12:                Set $x_p$ = y
```

One of the main limitations of the original MOEA/D algorithm is its limited search capability. In hopes of improving the search capabilities of the original MOEA/D algorithm, researchers have taken its basic framework and modified it with mutation methods such as Particle Swarm Optimization (Peng and Zhang, 2008) and Differential Evolution (Li and Zhang, 2008). Though well researched, these methods still contain performance inefficiencies that can be greatly improved through the use of different methods. Lévy Flight (LF) is one such method. Lévy Flight is a special type of random walk wherein the length of a step is obtained through a heavy-tailed distribution. As a result, Lévy Flight combines short motions and long trajectories to avoid being stuck at any locally optimum points — thus performing well on optimization tasks. Furthermore, because it is better at capturing feasible solutions in higher dimenions, Lévy Flight's optimization for random search (He and Aranha, 2020). Due to this attribute, Lévy Flight has been used in various realms for the purpose of optimization.

2 Optimization Formulation

By incorporating Lévy Flight with MOEA/D, we can improve the original MOEA/D algorithm by making it faster and less prone to becoming stuck at local optimums. Recall that the optimization problem for portfolio optimization is given by

$$\max_{\mathbf{w}} \quad \text{Return} = \sum_{i=1}^{n} r_i \mathbf{w}_i \tag{7}$$

$$\min_{\mathbf{w}} \quad \text{Risk} = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} \mathbf{w}_i \mathbf{w}_j \tag{8}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \mathbf{w}_i = 1, (0 \leq \mathbf{w}_i \leq 1) \tag{9}$$

Furthermore, some additional modifications are made to the MOEA/D-Lévy algorithm in order to improve the efficiency of the algorithm and thoroughness of the search. The work of Zhang et al. (2010) introduces a Normal Boundary Intersection-style Tchebycheff decomposition approach. With this approach, the vector optimization is transformed into a scalar optimization with λ, a normal vector of the convex hull of minima, and evenly distributed reference points $\{r_1, \ldots, r_N\}$ on the convex hull as given in the following equations (where F^1 and F^2) are extreme points (Zhang et al., 2010). With this method, we get that the i-th optima is the intersection of the Pareto front and a normal line found using λ and r_i. The equations that Zhang et al. (2010) present are

$$\min_x g^{tn}(x \mid \lambda, r) = \max_{m=1,2} \{\lambda_m (f_m(x) - r_m\} \tag{10}$$

$$\lambda = (\lambda_1, \lambda_2), \lambda_1 = |F_2^2 - F_s^1|, \lambda_2 = |F_1^2 - F_1^1| \tag{11}$$

$$r_i = a_i \cdot F^1 + (1 - a_i) \cdot F^2, a_i = \frac{N - i}{N + i} \tag{12}$$

Another important aspect of the MOEA/D-Lévy algorithm is the incorporation of Lévy Flight mutation. This mutation is similar to DE mutation as described by Ali and Pant (2011), as it uses the difference between individuals, but it is different in the sence that the scaling factor found in Lévy Flight mutations is a vector generated using a tail-heavy distribution, rather than a constant. This can be formulated as the following, where x_i and x_j are the parents, y is the offspring, α_0 is a scaling factor, and $\otimes$ is entry-wise multiplication. Note that b is a stability parameter that controls the shape of a stable distribution, thus controlling the balance between the local and global search in Lévy Flight (He and Aranha, 2020).

$$y = x_i + \alpha_0 \cdot (x_i - x_j) \otimes \text{Levy}(b) \tag{13}$$

$$\text{Levy}(b) \ \frac{u}{|v|^{\frac{1}{b}}}, u \ N(0, \sigma_u^2), v \ N(0, \sigma_v^2) \tag{14}$$

$$\sigma_u = \left(\frac{\Gamma(1 + b) \sin(\pi b/2)}{\Gamma((1 + b)/2) \cdot b 2^{(b-1)/2}} \right) \tag{15}$$

In general, the offspring that are the result of the Lévy Flight mutation may not satisfy the constraint on the weights given by equation (9). As such, we must apply a repair process to handle the constraint properly after reproduction (He and Aranha, 2020). In **Algorithm 2**, we can see how the repair method is applied (He and Aranha, 2020). In this repair process, we first set the negative variables to zero and then proceed to scale the entire vector (offspring) in such a way that the summation of all the individual elements will equal one. This is exactly what is necessary to satisfy the constraint for the portfolio weights in the portfolio optimization problem.

Algorithm 2 Repair Method

```
1: for $y_i$ in offspring $y$
2:     if $y_i < 0$:
3:         Set $y_i = 0$
4: Compute $s = \sum y_i$
5: for $y_i$ in offspring $y$
6:     Set $y_i = y_i / s$
```

3 Solution Method

Algorithm 3 MOEA/D-Lévy

```
1: Determine neighbors
2: Initialize the population as $\{x_1, \dots, x_N\}$
3: Compute the extreme points $F^1, F^2$
4: while stopping criteria do
5:     for $x_i$ in population:
6:         if rand(0, 1) $< \sigma$:
7:             Set $B(i)$ as neighbor of $x_i$
8:         else:
9:             Set $B(i)$ as population of $\{x_1, \dots, x_N\}$
10:        Select parents from $B(i)$
11:        Reproduce an offspring $y$ by Lévy Flight mutation and polynomial mutation
12:        Repair $y$ to satsify constraints
13:        Update extreme points $F_1, F_2$
14:        Set update counter $n_c = 0$
15:        Re-arange $B(i)$ in a random order
16:        for $x_p$ in neighbors of $B(i)$:
17:            if $g^{tn}(y \mid \lambda, r_p) \leq g^{tn}(x_p \mid \lambda, r_p$:
18:                Set $x + p = y$
19:                Set $n_c = n_c + 1$
20:                if $n_r < nc$:
21:                    break
```

In order to combine the Lévy Flight algorithm with MOEA/D, we follow the procedure outlined by He and Aranha (2020). The MOEA/D-Lévy algorithm incorporates Lévy Flight mutations and various polynomial mutation operators into the original MOEA/D algorithm. Since the main constraint in a portfolio optimization problem is the unit constraint keeping the sum of the weights (which are non-negative) equal to one, we can represent the portfolio as a vector in the algorithm. The i-th component of this vector corresponds to the rate of return of the i-th asset in the portfolio. In the MOEA/D-Lévy algorithm, the current individual is selected as x_i and x_j is randomly chosen from the neighbors of x_i. After an offspring is generated, the algorithm uses the repair steps in order to fulfill the weight constraint. After this, MOEA/D-Lévy uses Normal Boundary Intersection-style Tchebycheff decomposition, applied to handle the different scales of the two objective functions in portfolio optimization. Furthermore, to maintain diversity in the set, some random $1 - \sigma$ proportion of the population is set as a neighbor of x_i.

4 Verification

In order to better understand the MOEA/D-Lévy algorithm, we can follow the experiments presented by He and Aranha (2020) and run them on financial data from various markets. Because we would like to see how the addition of the Lévy Flight algorithm affects MOEA/D, we can compare the MOEA/D-Lévy algorithm with different versions of MOEA/D and

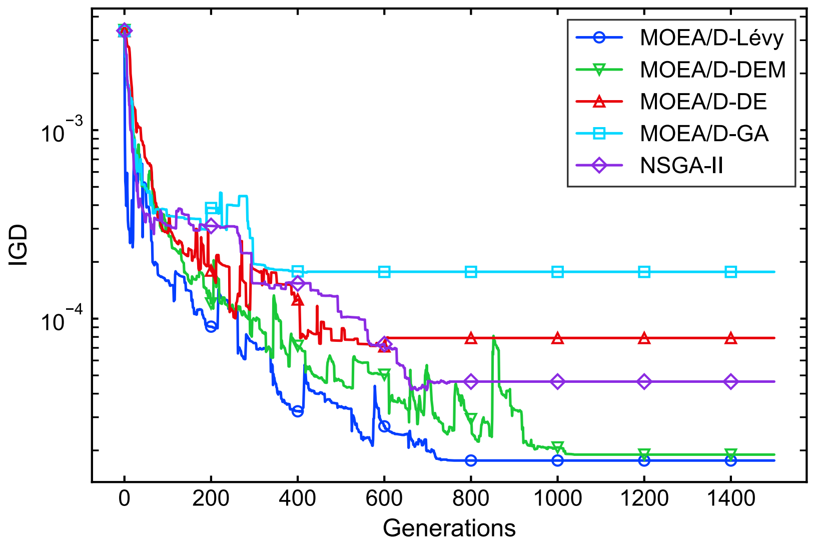

Figure 1: IDG metric versus generations on Nikkei.

NSGA-II. Furthermore, the effectiveness of the MOEA/D-Lévy algorithm can be better understood by comparing variants of the algorithm that utilize different distributions. In the evaluation of a MOEA, we must consider the performance on both convergence and diversity. In order to measure convergence, we make use mainly of the inverted generation distance (IGD). The inverted generation distance is defined as

$$\frac{\sum_{v \in A} d(v, P^*)}{|A|}$$

where $d(v, P^*)$ is the minimum Euclidean distance between a solution v in the Pareto front P^* and non-dominated set A.

From Figure 1, we can see how the MOEA/D-Lévy algorithm is able to achieve the best IGD value out of various different MOEA algorithms. In the test, we can observe how the algorithms quickly reduce the IDG within the first few generations, but then obtain different IDG values at there convergence points. In Figure 1, the MOEA/D-GA algorithm performs the worst, with the MOEA/D-DE, NSGA-II, and MOEA/D-DEM all performing worse than MOEA/D-Lévy.

In Figure 2, we can see MOEA/D-Lévy algorithm, while performing better than other MOEA methods, does not obtain a significant improvement in performance over the second-leader, MOEA/D-DEM. In this figure, notice that the IDG values for MOEA/D-Lévy and MOED/D-DEM are separated significantly for the first 300 generations, with MOEA/D-Lévy having the lower IDG, but then proceed to converge to the roughly the same IDG values.

In Figure 3, we once again see the MOEA/D-Lévy algorithm obtain the lowest IGD value out of the different methods. In fact, when running on the FTSE data, the MOEA/D-Lévy

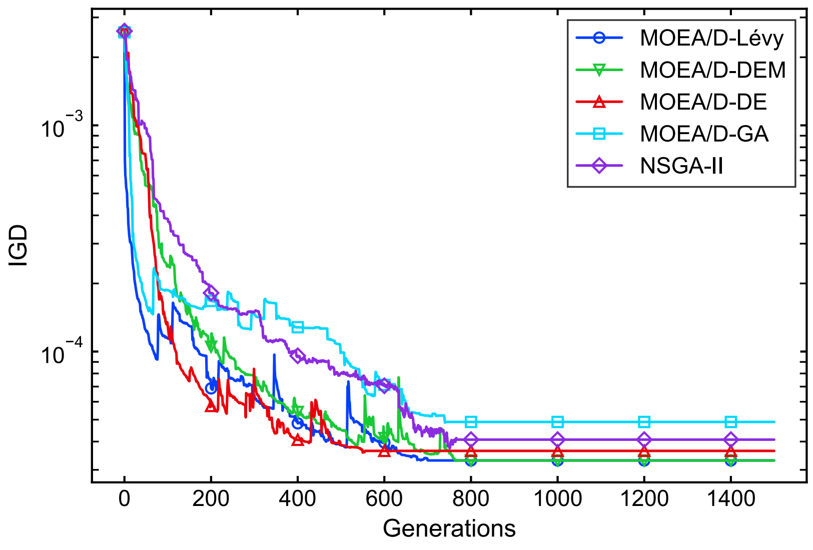

Figure 2: IDG metric versus generations on S&P 100.

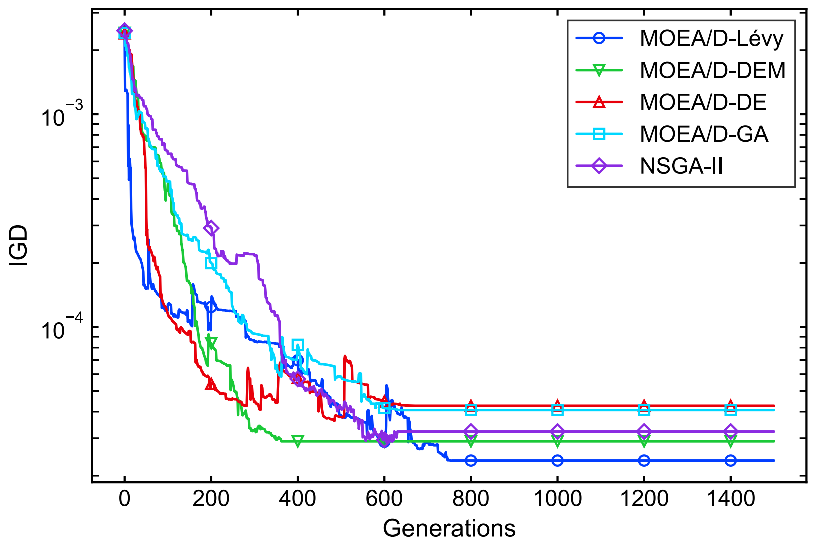

Figure 3: IDG metric versus generations on FTSE.

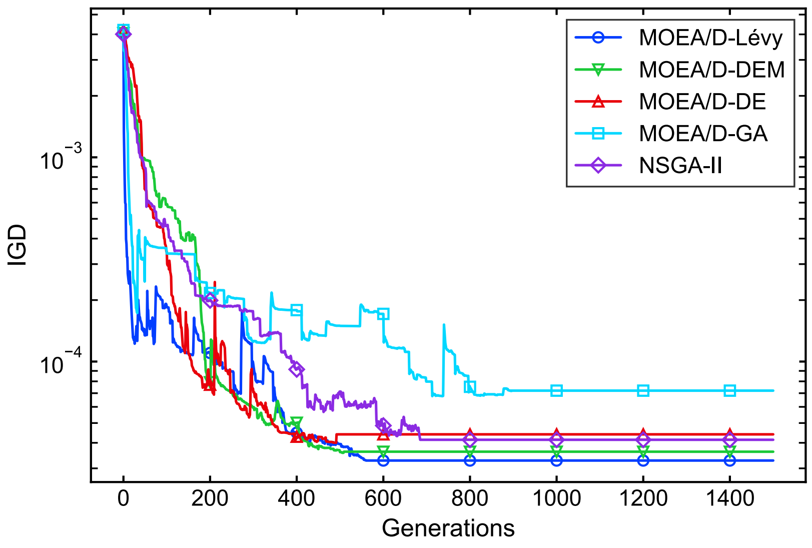

Figure 4: IDG metric versus generations on DAX.

algorithm's lead in terms of the lowest IGD increases, once again demonstrating the more extensive search capability of the algorithm.

If we shift to examining Figure 4, we can see that on the DAX data, the MOEA/D-Lévy algorithm once again presents a significant improvement over the other MOEA methods. In this test, while similar algorithms such as MOEA/D-DEM and MOEA/D-DE present similar IDG values for the first 500 generations, upon convergence (which seems to occur after 600 generations for these three algorithms), the IDG values stabilize at vastly different locations.

In Figure 5 and Figure 6, we can see the final populations in the objective space from the algorithms on the Nikkei and S&P 100 data, respectively. In Figure 5, notice that the MOEA/D-Lévy algorithm pushes closes to the line of Pareto front, closely followed by MOEA/D-DEM and NSGA-II. These three algorithms seem to perform relatively similar to each other at all levels of risk, as the return of the portfolios they produce stay relatively similar. In this figure, it is clear to see how MOEA/D-DE and MOEA/D-GA fail to return the most optimal results in the space. On the other hand, in Figure 6, nearly all of the methods produce roughly the similar line for Pareto front. In this scenario, the similarity of the performance of the algorithms could be attributed to differences in the sizes of the data, as the S&P 100 data used in this experiement was smaller than the Nikkei data. Looking at these graphs in the context of the original optimization problem, it is clear to see how portfolio optimization involves finding the best balance between risk and return. Figure 5 and Figure 6 demonstrate how optimization algorithms, such as the MOEA/D-Lévy algorithm, can be used to generate information reflect the optimal allocation of resources based on two different constraints that often clash with each other.

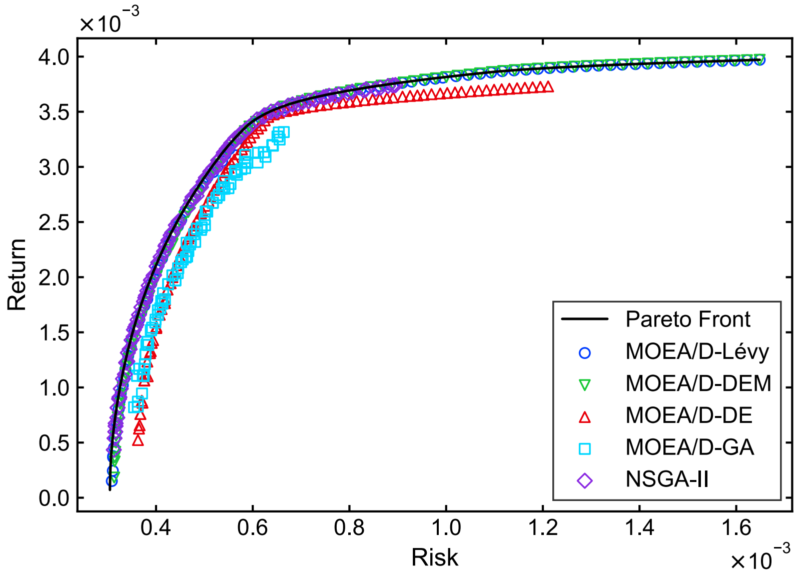

Figure 5: Final population on Nikkei in objective space.

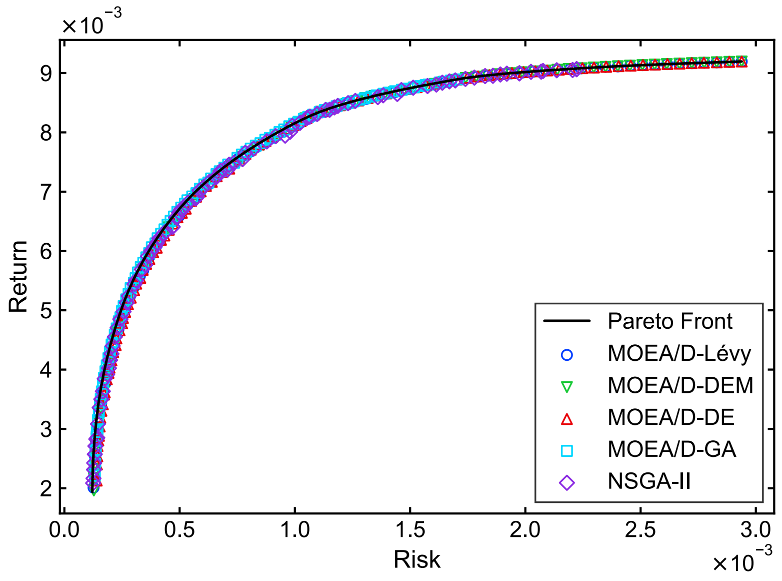

Figure 6: Final population on S&P 100 in objective space.

5 Conclusion

In this paper, we have examined the usefulness of the MOEA/D-Lévy algorithm in the realm of finance for the purpose of portfolio optimization. The multiple-objective nature of a portfolio optimization problem can be tackled in various ways; however, this paper studies how the combination of Lévy Flight in MOEA/D allows us to expand the search space, avoid becoming stuck in local optima, and obtain more optimal results. Based on the results and figures, it can be seen that an algorithm based on Lévy Flight is capable of promoting global search in early generations and can search various regions of the objective space simultaneously. As such, the MOEA/D-Lévy algorithm presents many improvements over other algorithms used to solve portfolio optimization problems.

References

Musrrat Ali and Millie Pant. Improving the performance of differential evolution algorithm using cauchy mutation. *Soft Computing*, 15:991–1007, 2011.

Carlos A Coello Coello, Silvia González Brambila, Josué Figueroa Gamboa, and Ma Guadalupe Castillo Tapia. Multi-objective evolutionary algorithms: past, present, and future. In *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, pages 137–162. Springer, 2021.

Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3–34. Springer, 2011.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

Yifan He and Claus Aranha. Solving portfolio optimization problems using moea/d and levy flight. *Advances in Data Science and Adaptive Analysis*, 12(03n04):2050005, 2020.

Wenlan Huang, Yu Zhang, and Lan Li. Survey on multi-objective evolutionary algorithms. In *Journal of physics: Conference series*, volume 1288, page 012057. IOP Publishing, 2019.

Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE transactions on evolutionary computation*, 13(2):284–302, 2008.

Wei Peng and Qingfu Zhang. A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems. In *2008 IEEE international conference on granular computing*, pages 534–537. IEEE, 2008.

Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

Qingfu Zhang, Hui Li, Dietmar Maringer, and Edward Tsang. Moea/d with nbi-style tchebycheff approach for portfolio management. In *IEEE congress on evolutionary computation*, pages 1–8. IEEE, 2010.