Portfolio Group Constraints

Magnus Erik Hvass Pedersen

February 12, 2022

Abstract

This paper presents a stand-alone algorithm for adjusting the weights of an investment portfolio, so they satisfy multiple overlapping group-constraints. This is useful for ensuring greater diversity of an investment portfolio. The algorithm is fairly simple and converges to a near-optimal solution in a small number of iterations, which in practice only takes a few milli-seconds to compute for a portfolio of 1000 assets. The algorithm also retains as much as possible of the diversity and relative magnitudes of the original portfolio weights.

1 Introduction

One way of controlling risk in an investment portfolio, is to ensure the portfolio is properly diversified across many different groups of assets, such as different industries, asset-classes, credit-ratings, countries, etc. If the groups are mutually exclusive, then this is an almost trivial problem to solve, because we can just normalize the portfolio weights within each group. But when the groups can be overlapping so one asset can belong to multiple groups, then the problem is much more difficult to find an optimal solution for. In fact, there is no single definition of what optimality even means, and there is no unique solution to this problem, as many different portfolio weights may be considered optimal.

There are few if any academic papers on this subject. Software packages for portfolio optimization often do support group-constraints, but it is unclear how they are calculated and what compromises have been made, and the algorithm often seems to be an integral part of the optimization method.

This paper presents a fairly simple stand-alone algorithm for adjusting portfolio weights, so they remain as close as possible to the original portfolio weights, while also satisfying the group-constraints. The algorithm converges very quickly to a near-optimal solution, that retains as much as possible of the diversity of the original portfolio weights, as well as the relative magnitudes of the original weights.

The group-constraint algorithm works well with the “filter-diversify” portfolio method that was presented in [Pedersen 2021], by using the group-constraint algorithm on the portfolio weights that are produced by the so-called “Hvass Diversification” algorithm, which improves portfolio diversification by lowering the correlations between assets. So this becomes a whole pipeline of portfolio algorithms that compliment each other and optimize different aspects of the portfolio’s risk and return, which may be dubbed the “filter-diversify-constrain” method for optimizing investment portfolios.

2 Algorithm

Let $Weight_i$ be the original portfolio weight for Asset *i* and let $Weight^*_{i,k}$ be the adjusted portfolio weight for the *k*'th iteration of the algorithm. Assume all the portfolio weights have the same sign, which is discussed in the next section. Let $Groups_i$ be the list of groups that Asset *i* belongs to, and conversely let $Assets_g$ be the list of assets that belong to Group *g*. Let $GroupSum_g$ be the sum of all portfolio weights belonging to Group *g*, and let $Group\,Limit_g$ be the required limit for $GroupSum_g$.

The following algorithm adjusts the portfolio weights so they are as close as possible to the originally desired weights, while all the group-sums are below their required group-limits or constraints:

- Initialize the adjusted portfolio weights to equal the original portfolio weights:

$$Weight^*_{i,1} = Weight_i \tag{1}$$

- For each Group *g*, calculate the sum of portfolio weights for all Assets *i* belonging to Group *g*:

$$GroupSum_g = \sum_{i \in Assets_g} Weight_i \tag{2}$$

- Initialize the list of remaining assets whose weights still need to be adjusted:

$$Remaining\ Assets = [1, 2, \ldots, N] \tag{3}$$

- For a number of iterations, or until the list of *Remaining Assets* is empty, do the following:
  - Calculate the ratios between the group-limits and the group-sums for all Groups *g*:

$$Group\ Ratio_g = Group\,Limit_g \ / \ GroupSum_g \tag{4}$$

  - For each Asset *i* still in the list of *Remaining Assets*, do the following:
    - Calculate the minimum group-ratio for all the Groups *g* that are associated with Asset *i*:

$$Min\,Group\ Ratio_i = \operatorname{Min}_{g \in Groups_i} \left(Group\ Ratio_g \right) \tag{5}$$

    - Calculate the ratio between the original and adjusted portfolio weights:

$$Weight\ Ratio_i = Weight_i \ / \ Weight^*_{i,k} \tag{6}$$

    - Calculate the minimum of the weight-ratio and all group-ratios associated with Asset *i*:

$$Min\ Ratio_i = \operatorname{Min}\left(Weight\ Ratio_i, \ Min\,Group\ Ratio_i \right) \tag{7}$$

    - Update the portfolio weight for Asset *i* by multiplying with the minimum ratio:

$$Weight_{i,k+1} = Weight_{i,k} \cdot Min\ Ratio_i \tag{8}$$

    - Update the group-sums for all Groups *g* that are associated with Asset *i*:

$$\forall\, g \in Groups_i \ : \ GroupSum_g = GroupSum_g - Weight^*_{i,k} + Weight^*_{i,k+1} \tag{9}$$

    - If $\left| Weight^*_{i,k+1} - Weight^*_{i,k} \right| < Required\ Precision$ then the update was negligible, so remove Asset *i* from the list of *Remaining Assets*, so it will not be processed any further.

3 Positive & Negative Weights

The algorithm above works for either positive or negative portfolio weights, but not both at the same time. We want the positive portfolio weights constrained by positive group-limits, and we want the negative portfolio weights constrained by negative group-limits. For example, we may want max 10% of our portfolio in so-called “long” investments (positive weights) for a given industry, while we may only want max 5% of our portfolio in “short” investments (negative weights) for the same industry.

There are two ways of using the algorithm above for both positive and negative portfolio weights:

The first way is to split the portfolio weights into the positive and negative weights, so we would have an array with the positive weights and another array with the negative weights, and we would then run the algorithm on each of these arrays individually, and then combine the results at the end. The advantage of this way is that the main algorithm is more concise, but the disadvantage is that it has to be run twice from a wrapper that splits and combines the positive and negative portfolio weights.

The second way also separates the computations for positive and negative weights, but it is done in an interleaved manner inside the main algorithm, so it always checks whether it is currently processing a positive or negative portfolio weight, and then it uses either the positive or negative group-sums and group-ratios, etc. The advantage of doing it this way, is that the main algorithm is only run once, but the disadvantage is that the algorithm is more complicated as it handles both positive and negative weights. This is the way it has been implemented in Section 13, because the computer code runs slightly faster.

4 Convergence

It can be rigorously proven that the algorithm converges, but we will just give a somewhat informal argument here. The portfolio weights are updated by multiplying with the minimum of the weight-ratio and the group-ratios for all the groups that the asset belongs to. Each of these ratios tells us how far the current portfolio weight is from one of its limits. If a ratio is below 1 then it means the given limit has been exceeded and the portfolio weight must be decreased by multiplying with that ratio, so the weight is brought down to its limit. Conversely, if a ratio is above 1 then it means the portfolio weight can be increased by multiplying with that ratio, and thereby bringing the weight up to its limit.

The weight-ratio calculated in Eq. (6) tells us how far the portfolio weight is from the originally desired portfolio weight, which is the upper limit for the weight, so the adjusted weight can never exceed the originally desired weight. And the group-ratios calculated in Eq. (4) tell us how far each group-sum is from its limit. We are then taking the minimum of all these ratios in Eq. (7) and using that minimum ratio to adjust the portfolio weight in Eq. (8). Because we are using the minimum of the weight-ratio and all the group-ratios associated with the asset, it ensures that all the limits are satisfied afterwards.

As we will see in the experiments below, the first iteration of the algorithm decreases the portfolio weights so all the group-limits are satisfied. But the first weight adjustments are usually excessive, so in the following iterations, the portfolio weights are gradually increased until they become as large as possible, while still being below the originally desired portfolio weights, and also still guaranteed to satisfy all the group-limits. Once the adjustment of a portfolio weight has become very small it means the weight has converged, so that asset is removed from the list of assets that are still being processed.

5 Time & Space Complexity

The algorithm has three nested loops. The outer loop has K iterations. The middle loop has N iterations; one for each asset in the portfolio. And the inner loop has G iterations; one for each group. So the worst-case time-complexity of the algorithm is:

$$O(K \cdot N \cdot G) \tag{10}$$

However, the assets usually do not belong to all the groups, so the inner loop often has far fewer than G iterations. And because the assets are removed from the processing once their weights have converged, the middle loop often has far fewer than N iterations. So in practice the time-usage is often much less than the worst-case time-complexity above.

5.1 Actual Time-Usage

Figure 1 and Figure 2 show the actual time-usage for many random portfolios generated as follows:

- The number of assets in the portfolio is: 50, 100, 250, 500, 750, 1000, 1250, 1500, 1750, 2000.
- The number of groups in the portfolio is: 1, 10, 50, 100, 200, 300, 400, 500.
- For each combination of the number of assets and groups, 100 random portfolios are generated, where each random portfolio is generated as follows:
  - The portfolio weights are taken from a normal distribution with zero mean and 0.5% standard deviation, and then clipped between ±100% (although that is not necessary here).
  - The association between assets and groups is generated randomly with uniform probability. The min number of assets per group is 1 and the max is the number of groups available.
  - The pos/neg group-limits are from uniform distributions between ±5% and ±20%, and about 5% of the group-limits are randomly set to ±infinity to disable those group-limits.

The group constraint algorithm from Section 2 is allowed to run for max 30 iterations, and the experiment is run on a computer with CPU speed of 2.6 GHz (3.5 GHz boost).

Figure 1 shows the number of assets on the x-axis, the number of groups as the colour-hue, and the time-usage on the y-axis. This shows the relation between the number of assets and the time-usage is roughly linear with some noise. But as we also noted above when discussing the time-complexity of the algorithm, the relation between portfolio size and time-usage is actually sub-linear, because the algorithm stops processing the portfolio weights once they have converged, so the time-usage generally scales with a factor below 1 as the number of assets increases.

Figure 2 shows the number of groups on the x-axis and the number of assets as the colour-hue. Once again the relation is roughly linear with some noise. But this relation is even more sub-linear, because the assets typically only belong to some of the groups and not all of them, so the time-usage scales with a factor much below 1 as the number of groups increases.

5.2 Space Complexity

The data for the association between assets and groups has max $O(N \cdot G)$ elements, but when stored as a list-of-lists of varying lengths, then it is typically much less. In the implementation in Section 13, for runtime efficiency we need to copy this data-structure to a different format, which takes around 5 msec and requires the same amount of storage. The algorithm then only needs an array of size $O(N)$ for the adjusted portfolio weights, and another couple of arrays of size $O(G)$ for the group-sums and ratios.

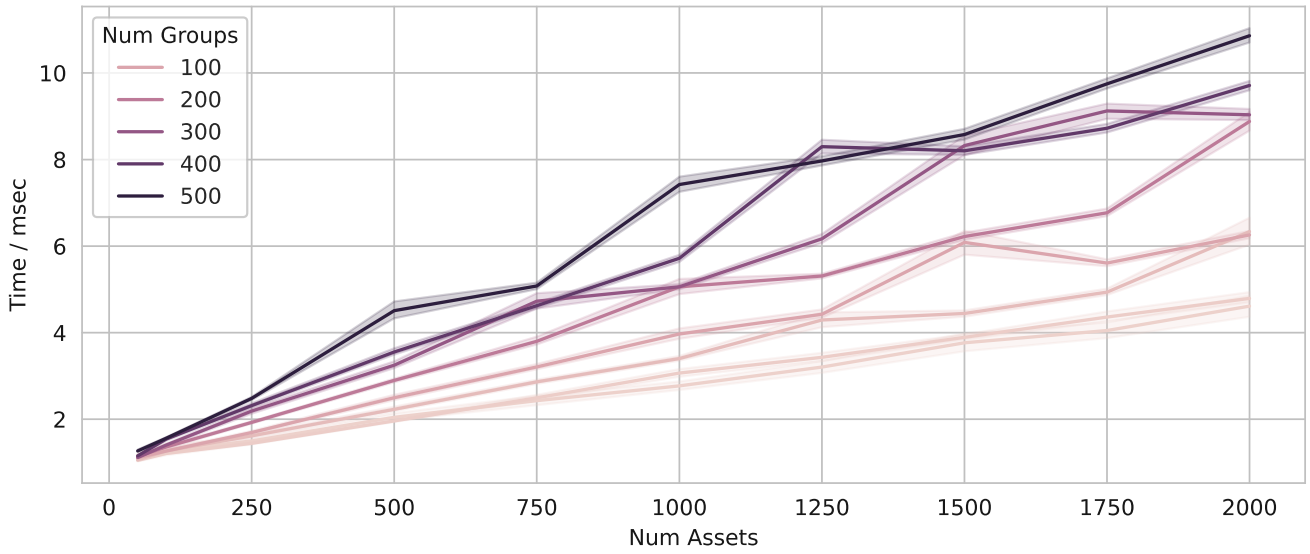

Figure 1: Time-usage when varying the number of assets (x-axis) and the number of groups (hue). The lines show the mean of 100 trials and the shaded areas are the 99% confidence intervals.

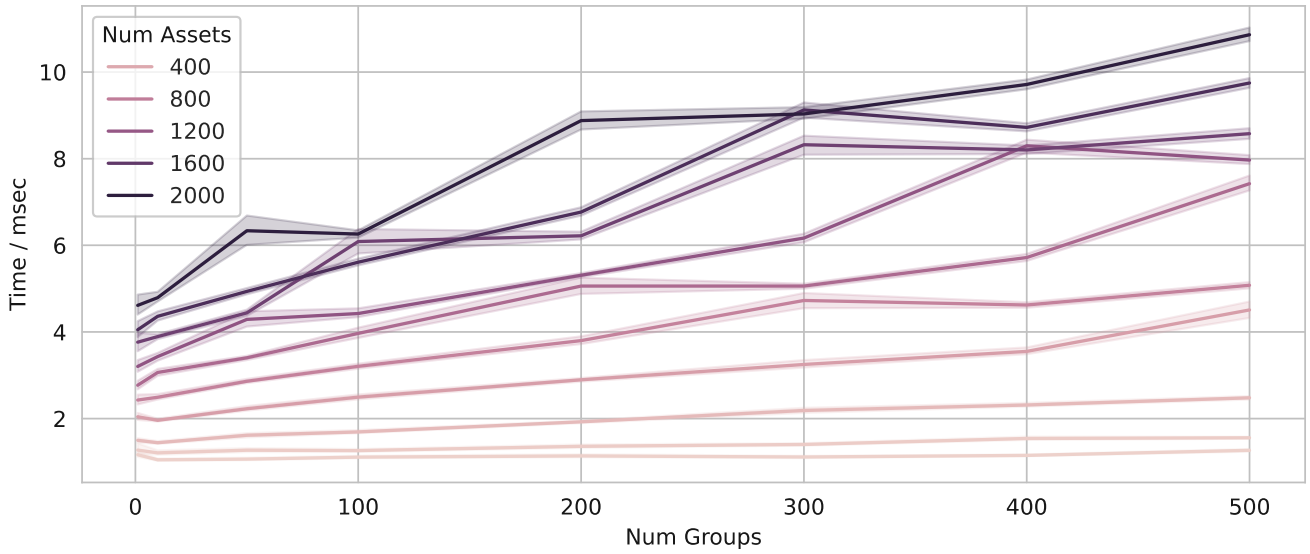

Figure 2: Time-usage when varying the number of groups (x-axis) and the number of assets (hue). The lines show the mean of 100 trials and the shaded areas are the 99% confidence intervals.

6 Simple Example – Only Positive Weights

Let us first consider a simple example where the portfolio has 4 assets with only positive weights:

$$Weight_1 = 5\%,\ Weight_2 = 10\%,\ Weight_3 = 15\%,\ Weight_4 = 20\% \tag{11}$$

There are 3 groups and the associations between the groups and assets are as follows:

$$Group_A = [Asset\,1,\ Asset\,2]\quad Group_B = [Asset\,2,\ Asset\,3]\quad Group_C = [Asset\,3,\ Asset\,4] \tag{12}$$

The limits or constraints for the groups are as follows, where we only use the positive limits because the portfolio does not have any assets with negative weights:

$$Group\ Limit\ Pos_A = 5\%,\ Group\ Limit\ Pos_B = 10\%,\ Group\ Limit\ Pos_C = 20\% \tag{13}$$

Using the algorithm from Section 2 gives the following results for all the iterations until convergence:

	Weights				**Group-Sums**			**Group-Ratios**		
k	W_1	W_2	W_3	W_4	G_A	G_B	G_C	G_A	G_B	G_C
1	5%	10%	15%	20%	15%	20%	35%	0.33	0.4	0.57
2	1.67%	3.33%	6%	11.43%	5%	9.33%	17.43%	1.0	1.07	1.15
3	1.67%	3.33%	6.43%	13.11%	5%	9.76%	19.54%	1.0	1.02	1.02
4	1.67%	3.33%	6.58%	13.42%	5%	9.91%	20%	1.0	1.01	1.0
5	1.67%	3.33%	6.58%	13.42%	5%	9.91%	20%	1.0	1.01	1.0

The final adjusted portfolio weights from the last row of the table are as follows:

$$Weight_1^* = 1.67\%,\ Weight_2^* = 3.33\%,\ Weight_3^* = 6.58\%,\ Weight_4^* = 13.42\% \tag{14}$$

The group-sums for these weights are also shown in the last row of the table, and they are below the limits given in Eq. (13). The group-ratios from the last iteration in the table are all 1.0 or slightly above, which again means that all the group-sums are at or below their group-limits. So the algorithm has found a solution in just a few iterations, that satisfies all the group-constraints in a near-optimal way.

Now consider the following portfolio weights as an alternative solution:

$$Weight_1^* = 5\%,\ Weight_2^* = 0\%,\ Weight_3^* = 10\%,\ Weight_4^* = 10\% \tag{15}$$

These weights result in the following group-sums, which exactly equal the group-limits from Eq. (13):

$$GroupSum_A = 5\%,\ GroupSum_B = 10\%,\ GroupSum_C = 20\% \tag{16}$$

So these alternative portfolio weights are maximizing the group-sums, while the weights in Eq. (14) that were found by the algorithm, resulted in the sum for Group B to be only 9.91% instead of 10%. Does this mean that the alternative weights are better? In both cases the sum of the portfolio weights is 25%, but we might be able to find examples where the weight-sums are not equal, and the algorithm had a slightly lower weight-sum. But the most important difference lies in the diversification aspects. The algorithm retains as much as possible of the diversification of the original portfolio weights, as well as the relative magnitude of the different portfolio weights, which is important because larger weights either indicate that we have more confidence in those assets, or that we have estimated their future returns to be higher. So the algorithm found a very good solution for this simple example.

7 Simple Example – Positive & Negative Weights

Let us now consider a simple example where the portfolio has both positive and negative weights:

$$Weight_1 = -5\%,\ Weight_2 = -10\%,\ Weight_3 = 15\%,\ Weight_4 = 20\% \tag{17}$$

We are using the same associations between groups and assets as in the previous example:

$$Group_A = [Asset\,1,\ Asset\,2]\quad Group_B = [Asset\,2,\ Asset\,3]\quad Group_C = [Asset\,3,\ Asset\,4] \tag{18}$$

The positive group-limits or constraints are:

$$Group\ Limit\ Pos_A = 5\%,\ Group\ Limit\ Pos_B = 10\%,\ Group\ Limit\ Pos_C = 20\% \tag{19}$$

The negative group-limits or constraints are:

$$Group\ Limit\ Neg_A = -5\%,\ Group\ Limit\ Neg_B = -10\%,\ Group\ Limit\ Neg_C = -20\% \tag{20}$$

Using the algorithm from Section 2 gives the following results for all the iterations until convergence:

	Weights				**Group-Ratios Pos**			**Group-Ratios Neg**		
k	W_1	W_2	W_3	W_4	G_A	G_B	G_C	G_A	G_B	G_C
1	-5%	-10%	15%	20%	Inf	0.67	0.57	0.33	1.0	Inf
2	-1.67%	-3.33%	8.57%	11.43%	Inf	1.17	1.0	1.0	3.0	Inf
3	-1.67%	-3.33%	8.57%	11.43%	Inf	1.17	1.0	1.0	3.0	Inf

The final adjusted portfolio weights from the last row of the table are as follows:

$$Weight_1^* = -1.67\%,\ Weight_2^* = -3.33\%,\ Weight_3^* = 8.57\%,\ Weight_4^* = 11.43\% \tag{21}$$

The table also shows the positive and negative group-ratios:

- The positive group-ratio is infinity for Group A, because that group consists of Assets 1 and 2 whose weights are negative, so the positive group-sum is zero and the group-ratio is infinity.
- The positive group-ratio for Group B is 1.17 because that group consists of Assets 2 and 3, but only Asset 3 has a positive weight of 8.57% while the positive group-limit for Group B is 10%, so the group-ratio of 1.17 tells us that the weight for Asset 3 could be increased by a factor 1.17 before it reaches the positive group-limit for Group B.
- The positive group-ratio for Group C is 1.0 which means it is exactly at its limit.
- The negative group-ratio for Group A is 1.0 because the sum of the weights for Asset 1 and 2 is -5% which is exactly equal to the negative group-limit for Group A.
- The negative group-ratio for Group B is 3.0 because the group consists of Asset 2 and 3, but only Asset 2 has a negative weight of -3.33% while the negative group-limit is -10% for that group, so the weight could be 3 times as large before it reaches the group-limit.
- The negative group-ratio for Group C is infinity because that group consists of Assets 3 and 4 which have positive weights, so the negative group-sum is zero and the group-ratio is infinity.

8 Big Example

Let us now consider a big example where the portfolio has 1000 assets and 20 groups, which are generated using the same method that was described in Section 5.1, so both the portfolio weights, the associations between assets and groups, and the group-limits are all randomly generated.

Figure 3 shows the weight- and group-ratios for the iterations of the algorithm. Iteration 0 here is for the original portfolio weights (which is denoted as iteration 1 in the algorithm description in Section 2).

The top-plot in Figure 3 shows the weight-ratio – this is actually the inverted weight-ratio from Eq. (6), because the ratios are then limited between zero and one so they are easier to display in the plot. The middle-plot shows the positive group-ratios, which are calculated using Eq. (4) for all the positive portfolio weights. And the bottom-plot shows the negative group-ratios, which are also calculated using Eq. (4) but for all the negative portfolio weights instead.

For the initial iteration of the algorithm, all the weight-ratios are exactly equal to 1 because the initial weights are set to equal the original weights. After one iteration of the algorithm, the top-plot shows that nearly all the portfolio weights have decreased substantially relative to their original weights, because most of the weight-ratios are now below 0.2. The middle- and bottom-plots show this was an over-adjustment, because all the group-ratios are now above 1, and many of the ratios are well above 1.

This means the portfolio weights that are below their original weights, and all their associated group-sums are below their group-limits, those portfolio weights can be increased again without violating any of the constraints. And this is what happens in the following iterations of the algorithm, as can be seen from the top-plot where the (inverted) weight-ratios gradually increase towards 1, which means the adjusted portfolio weights get increasingly closer to their original weights. And the group-ratios gradually decrease towards 1, which means the group-sums get closer to their group-limits.

After around 10 iterations of the algorithm, most of the group-ratios seem to have converged, but as we can see from the top-plot, a small number of weight-ratios are still increasing until the plot stops at 20 iterations, and perhaps if we had let the algorithm run for 30 iterations instead, the algorithm would have increased some of the portfolio weights a bit more before they all converged.

What these plots show, is that after the first iteration of the algorithm, all the group-constraints are satisfied, because all the group-sums are below their group-limits, which shows in the middle- and bottom-plots as the group-ratios being above 1. But the first iteration was actually an over-adjustment, because a lot of the portfolio weights could be increased again without violating the group-constraints. So in the remaining iterations, the algorithm gradually increases some of the portfolio weights to push them back up towards the limits. And some of the adjusted portfolio weights can actually become equal to their original weights, which shows in the top-plot as the (inverted) weight-ratios being equal to 1.

Many of the group-ratios cannot get down to equal 1, because there is a conflict between the overlapping groups, so if the affected portfolio weights are increased any more, it would cause some of the group-sums to exceed their group-limits. The algorithm has found a near-optimal compromise that satisfies all the group-constraints, while keeping the adjusted portfolio weights as close as possible to the original portfolio weights.

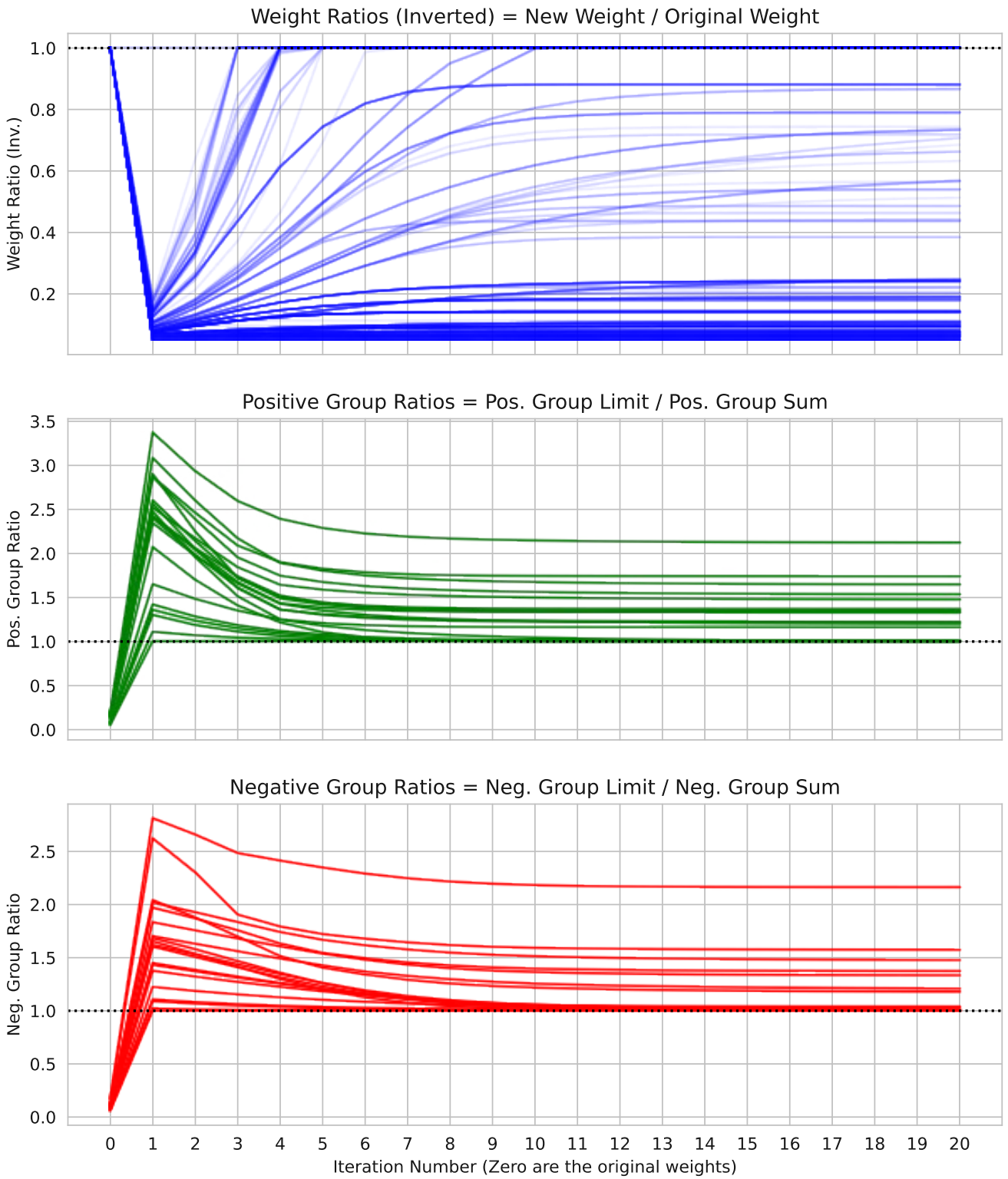

Figure 3: These plots show the weight- and group-ratios for the iterations of the algorithm. The top-plot shows the ratios between the new and original portfolio weights (this is the inverse of Eq. (6)). The middle-plot shows the group-ratios for the positive portfolio weights. And the bottom-plot shows the group-ratios for the negative portfolio weights. All the group-ratios are calculated using Eq. (4).

9 Lower Group-Limits

In software packages for doing portfolio optimization with group constraints, it is often possible to set a lower limit for the group-sum, whereas the algorithm in this paper only supports an upper limit for the group-sum. It can be debated whether it makes any sense to have a lower limit, because if you only want to invest in assets that you have estimated to have a sufficiently high future return, then it would be moronic to force an investment in the assets you have estimated to result in a future loss, just to ensure that your portfolio contains enough assets from certain groups. Furthermore, there can be a conflict between the lower group-limits, so it might be impossible to satisfy all of them simultaneously.

However, if you do not have a strong belief about the future returns of the assets, and you merely want to ensure that your portfolio holds assets from many different groups (such as different industries, countries, asset-classes, credit-ratings, etc.), then a better way to do it, is to use equal portfolio weights within each group, depending on how much you want invested in that group. And then run the algorithm from this paper to ensure that all the portfolio weights satisfy the upper group-limits.

10 Overall Portfolio Constraints

The original portfolio weights that are going into the algorithm, and the adjusted portfolio weights that are produced by the algorithm, can have their overall sums exceed the overall limits of the portfolio. That is, the sums of the positive portfolio weights can exceed 1, which would mean that you would have to invest for borrowed money. And the sums of the negative portfolio weights can exceed -1.

In order to get the sums of the final portfolio weights limited to e.g. 1 for the positive portfolio weights and -1 for the negative portfolio weights, we could use a group that contains all assets in the portfolio, and set the positive and negative limits for this group to 1 and -1 respectively. This is a perfectly valid solution that uses the group-constraint algorithm to ensure the final portfolio weights are within limits.

Another solution is to use ordinary normalization of the portfolio weights as a final step after having run the group-constraint algorithm. This could be slightly faster to run because the normalization algorithm is simpler, and it would also allow for other adjustments between the positive and negative weights, e.g. to ensure that margin-requirements for the short-positions are satisfied.

11 Floating-Point Rounding Errors

In mathematical terms, the group-constraint algorithm is guaranteed to produce portfolio weights that are less or equal to the original portfolio weights, and whose group-sums are less or equal to the group-limits. But when the algorithm is implemented using floating-point operations with limited precision, rounding errors may cause the adjusted portfolio weights to be a tiny bit higher than the original weights, and the group-sums may be a tiny bit higher than the group-limits. The error is something like 1e-17 so it has no practical effect on the portfolio allocation. But we can easily fix this by ensuring the adjusted weights are always less or equal to the original weights after being updated with Eq. (8), and we can allow for a small error-tolerance when comparing the group-sums to the group-limits.

12 Conclusion

This paper presented a stand-alone algorithm for adjusting the weights of an investment portfolio so they satisfy group-constraints for many overlapping groups of assets. This can be used to improve the portfolio's diversification across different industries, asset-classes, countries, credit-ratings, etc.

The algorithm is fairly simple, and it tries to maintain the diversity and relative magnitudes of the original portfolio weights, so as to find a near-optimal solution where the adjusted portfolio weights are as close as possible to the original portfolio weights, while satisfying all the group-constraints. The algorithm is guaranteed to find a valid solution, and it typically converges to a near-optimal solution within a small number of iterations, which in practice only takes a few milli-seconds to compute for a portfolio of 1000 assets.

The group-constraint algorithm can also be used with the "filter-diversify" portfolio method from a previous paper, so as to create a whole pipeline of portfolio algorithms that compliment each other, for optimizing a portfolio with regard to different aspects of return maximization and risk minimization.

13 Computer Code

The following computer code is freely available and is written in the Python programming language:

- The Python Notebook contains the computer code used to run all the tests and generate all the plots and statistics in this paper.
- FinanceOps is the GitHub repository that contains all of my newest research in finance, including the Python Notebook for this paper. It is generally recommended that you download the entire repository if you want to run a Python Notebook, because some of them may require data-files that are included with the repository. You can also run the Python Notebooks entirely in the cloud using the free Google Colab service. Instructions are found in the link above.
- InvestOps is a Python package that you can easily import into your own Python program to use the group constraint algorithm from this paper. Instructions are found in the link. This may also serve as a reference implementation, if you want to implement the algorithm in another programming language. The file named `group_constraints.py` contains the algorithm.
- InvestOps Tutorial shows how to use the algorithm in a small Python program, which can also be run in the cloud using the free Google Colab service.

14 References

[Pedersen 2021] M.E.H. Pedersen, *"Simple Portfolio Optimization That Works!"*, 2021. [PDF]