Fast Portfolio Diversification

Magnus Erik Hvass Pedersen

January 14, 2022

Abstract

This is a concise description of a new algorithm for diversifying an investment portfolio, that was previously described and tested in great detail in another paper, which also showed that the algorithm is very fast and guaranteed to converge to the optimal solution in just a few iterations, and the algorithm improves the portfolio on several performance metrics, while being extremely robust to estimation errors in the correlation matrix. This paper further shows how to implement the algorithm for a sparse correlation matrix, and it compares the time-usage of the sparse and dense versions of the algorithm, as well as parallel and serial versions.

1 Introduction

A previous paper [Pedersen 2021] presented a new method for optimizing an investment portfolio, which was called the "filter-diversify" method because it consisted of two separate phases: First it used a simple filter to only select the assets whose investment returns were estimated to be sufficiently high, and then it used a new algorithm for improving the portfolio's diversification by lowering correlations.

The diversification algorithm was dubbed "Hvass Diversification" and has several advantages. But because the original paper is over 160 pages long, and the section with the diversification algorithm is nearly 30 pages long, it may give the false impression that the new diversification algorithm is very complicated, when in fact it is very simple. This paper is meant to give a more concise presentation.

The diversification algorithm can be described in just a few lines of pseudo-code, and it is extremely fast to run because it only requires a small number of loop-iterations to find the optimal solution, where each iteration has time-complexity $O(N^2)$ for a portfolio of N assets, so the algorithm has near-optimal performance as the correlation matrix has N^2 elements. The sparse algorithm has even lower time-complexity, but it is only faster in practice when the correlation matrix is very sparse.

The main paper also proves that the diversification algorithm always converges to the optimal solution. And the main paper also has thousands of tests on real-world stock-data, which show that the algorithm significantly improves the portfolio on several performance metrics, even when the correlation matrix is very noisy or even completely malformed. This is probably because the diversification algorithm only allows the portfolio weights to decrease, so the worst that can happen is that it moves too much of the portfolio into cash (or another low-risk asset of your choice).

2 Full Exposure

The main idea behind the diversification method, is to consider the so-called “Full Exposure” of each asset in the portfolio, which measures how much the portfolio is exposed to each asset, both directly through the portfolio’s investment in that particular asset, but also indirectly through correlations with other assets in the portfolio.

Let $Weight_i$ denote the originally desired portfolio weight for Asset *i*, which has been calculated by some other process, for example using a simple filtering process on the estimates for the future asset returns, as was done in the main paper.

For example, say we have invested $Weight_A = 9\%$ of the portfolio in Asset A and $Weight_B = 12\%$ in Asset B, and they are correlated with coefficient $\rho_{A,B} = 0.5$, so the two assets tend to move up or down together in price. Because of the correlation between the two assets, it seems like we have actually invested more than 9% of the portfolio in Asset A and more than 12% of the portfolio in Asset B.

Intuitively we might define the Full Exposure of an asset to be its own portfolio weight plus the sum of all the other correlated portfolio weights, each multiplied with the correlation coefficient between the two assets. So in this example we would have the following Full Exposure for Asset A:

$$FullExposure_A = Weight_A + \rho_{A,B} \cdot Weight_B = 9\% + 0.5 \cdot 12\% = 15\% \quad (1)$$

And the following Full Exposure for Asset B, because the correlation is symmetrical $\rho_{A,B} = \rho_{B,A}$:

$$FullExposure_B = Weight_B + \rho_{B,A} \cdot Weight_A = 12\% + 0.5 \cdot 9\% = 16.5\% \quad (2)$$

So we thought we had only invested 9% of the portfolio in Asset A, but through its correlation with Asset B, the portfolio’s Full Exposure to Asset A is in fact 15% of the portfolio. Similarly, we thought we had only invested 12% of the portfolio in Asset B, but through its correlation with Asset A, the portfolio’s Full Exposure to Asset B is in fact 16.5% of the portfolio.

The goal is then to find new portfolio weights denoted $Weight_i^*$ (marked with an asterisk * to indicate they are the new or adjusted weights), so that both of the Full Exposures that are calculated using these new weights, will be equal to the originally desired $Weight_A$ and $Weight_B$. That is, we want to find $Weight_A^*$ and $Weight_B^*$ that solve these two equations:

$$\begin{aligned} Weight_A &= FullExposure_A^* = Weight_A^* + \rho_{A,B} \cdot Weight_B^* \\ Weight_B &= FullExposure_B^* = Weight_B^* + \rho_{B,A} \cdot Weight_A^* \end{aligned} \quad (3)$$

The solution is $Weight_A^* = 4\%$ and $Weight_B^* = 10\%$, so that is how much of the portfolio we should actually invest in Assets A and B, if we want their Full Exposures to equal 9% and 12% respectively.

This is fairly easy to solve for a portfolio of only two assets. But it is much more difficult for larger portfolios, especially if we also allow negative portfolio weights and correlations. Furthermore, the definition of Full Exposure that was used above is actually over-simplified and slightly incorrect.

2.1 Formula

In order for the mathematical definition of the Full Exposure to be valid, it must satisfy these criteria:

$$\begin{array}{lll} (1) & \mathit{Full\,Exposure}_i = 0 & \text{if } \mathit{Weight}_i = 0 \\ (2) & \mathit{Full\,Exposure}_i = \mathit{Weight}_i & \text{if } \mathit{Weight}_j = 0 \text{ or } \rho_{i,j} = 0 \text{ for all } j \neq i \\ (3) & \left|\mathit{Full\,Exposure}_i\right| \geq \left|\mathit{Weight}_i\right| & \\ (4) & \operatorname{sign}\left(\mathit{Full\,Exposure}_i\right) = \operatorname{sign}\left(\mathit{Weight}_i\right) & \end{array} \tag{4}$$

These criteria must also hold for the adjusted portfolio weights Weight_i^* and the Full Exposure that is calculated using the adjusted weights and which is denoted $\mathit{Full\,Exposure}_i^*$ (also with an asterisk *).

The criteria are explained in more detail in the main paper, so let us just briefly explain them here:

1) If the portfolio weight for Asset *i* is zero, then its Full Exposure must also be zero, otherwise it would cause the adjusted portfolio weights for the other correlated assets to also become zero.
2) If all other portfolio weights are zero, or if all correlations with other assets are zero, or a combination of the two, then the Full Exposure of the asset must equal the portfolio weight.
3) The magnitude of the Full Exposure must be greater than or equal to the magnitude of the portfolio weight. This ensures the adjusted portfolio weights will only get smaller, so we will not increase the portfolio weights to try and improve the correlations between assets.
4) The sign of the Full Exposure must be equal to the sign of the portfolio weight.

There are many mathematical definitions of the Full Exposure that would satisfy the criteria above. Through experimentation it was found that the following definition made the portfolios perform well in practice. You may check that this definition of the Full Exposure satisfies all the criteria above:

$$\mathit{Full\,Exposure}_i = \operatorname{sign}\left(\mathit{Weight}_i\right) \cdot \sqrt{\sum_{j=1}^{N} \left|\mathit{Weight}_i \cdot \mathit{Weight}_j \cdot \rho_{i,j}^2 \cdot \mathit{Use}_{i,j}\right|} \tag{5}$$

Where $\mathit{Use}_{i,j}$ is short for “Use in calculation of the Full Exposure” and is a value of either zero or one, which specifies whether the correlation between Assets *i* and *j* should be included in the calculation of the Full Exposure, so their portfolio weights will be adjusted accordingly. This is easier to understand if we only have positive portfolio weights, where we consider the positive correlations as “bad” and negative correlations as “good”, and we only want to adjust the portfolio weights with regard to the “bad” correlations. This becomes more complicated when we also allow negative portfolio weights, but it turns out that we can easily distinguish between “good” and “bad” correlations using the following simple formula, which is explained in more detail in the main paper:

$$\mathit{Use}_{i,j} = \begin{cases} 1 & \text{if } \operatorname{sign}\left(\mathit{Weight}_i \cdot \mathit{Weight}_j \cdot \rho_{i,j}\right) \text{ is } + \\ 0 & \text{else} \end{cases} \tag{6}$$

We are now able to measure how much the portfolio is really exposed to each asset, both through its direct investment in that asset, but also through correlations with other assets in the portfolio.

2.2 Dense Algorithm

The following algorithm calculates the Full Exposure from Eq. (5) for all assets in the portfolio, when the correlation matrix is dense so it mostly contains non-zero correlation coefficients:

- For each Asset *i* in the portfolio, do the following (note that this loop can be run in parallel):
  - Initialize the sum of correlated exposures with zero for each Asset *i*:

$$Sum\ Corr\ Exposure_i = 0 \tag{7}$$

  - For each Asset *j* in the portfolio (also including Asset *i* itself), update the sum of correlated exposures with the weighted correlation between Assets *i* and *j*:

$$Sum\ Corr\ Exposure_i = Sum\ Corr\ Exposure_i + \left| Weight_i \cdot Weight_j \cdot \rho_{i,j}^2 \cdot Use_{i,j} \right| \tag{8}$$

  - Finalize the calculation of the Full Exposure for each Asset *i*:

$$Full\ Exposure_i = \mathrm{sign}\left(Weight_i\right) \cdot \sqrt{Sum\ Corr\ Exposure_i} \tag{9}$$

2.3 Sparse Algorithm

The following algorithm calculates the Full Exposure from Eq. (5) for all assets in the portfolio, when the correlation matrix is sparse so most of the coefficients are zero. The sparse matrix is given as a list of tuples $\left(i, j, \rho_{i,j}\right)$ which specify the non-zero correlation $\rho_{i,j}$ between Assets *i* and *j*. The algorithm assumes that only the upper-triangle of the symmetrical correlation matrix is given as input:

- Initialize the sum of correlated exposures for each Asset *i* with its own squared weight, because the diagonal of ones has been omitted from the sparse correlation matrix:

$$Sum\ Corr\ Exposure_i = Weight_i^2 \tag{10}$$

- For each tuple $\left(i, j, \rho_{i,j}\right)$ in the sparse correlation matrix, do the following:
  - Calculate the correlated exposure between Assets *i* and *j*:

$$Corr\ Exposure_{i,j} = \left| Weight_i \cdot Weight_j \cdot \rho_{i,j}^2 \cdot Use_{i,j} \right| \tag{11}$$

  - As the correlation coefficient is symmetrical $\rho_{i,j} = \rho_{j,i}$, the correlated exposure is also symmetrical, so it is added to the sums of correlated exposures for both Assets *i* and *j*:

$$\begin{aligned} Sum\ Corr\ Exposure_i &= Sum\ Corr\ Exposure_i + Corr\ Exposure_{i,j} \\ Sum\ Corr\ Exposure_j &= Sum\ Corr\ Exposure_j + Corr\ Exposure_{i,j} \end{aligned} \tag{12}$$

- Finalize the calculation of the Full Exposure for all Assets *i* in the portfolio:

$$Full\ Exposure_i = \mathrm{sign}\left(Weight_i\right) \cdot \sqrt{Sum\ Corr\ Exposure_i} \tag{13}$$

3 Weight Update

We now need a method for finding the new portfolio weights denoted $Weight_i^*$ so the Full Exposure calculated using these new weights and denoted $Full\,Exposure_i^*$ (also with an asterisk *), equal the originally desired portfolio weights denoted $Weight_i$ (without the *). So we want to find $Weight_i^*$ that solves the following equation for all Assets *i* simultaneously:

$$Full\,Exposure_i^* \;=\; Weight_i \tag{14}$$

Because the magnitude of the Full Exposure is by definition greater or equal to the portfolio weight (see criterion (3) in Eq. (4)), we can only lower the portfolio weights to satisfy Eq. (14), which therefore guarantees the new portfolio weights are always smaller than the original portfolio weights:

$$\left|Weight_i^*\right| \;\le\; \left|Weight_i\right| \tag{15}$$

The relation between the original and adjusted portfolio weights, and their Full Exposure, is shown conceptually in Figure 1.

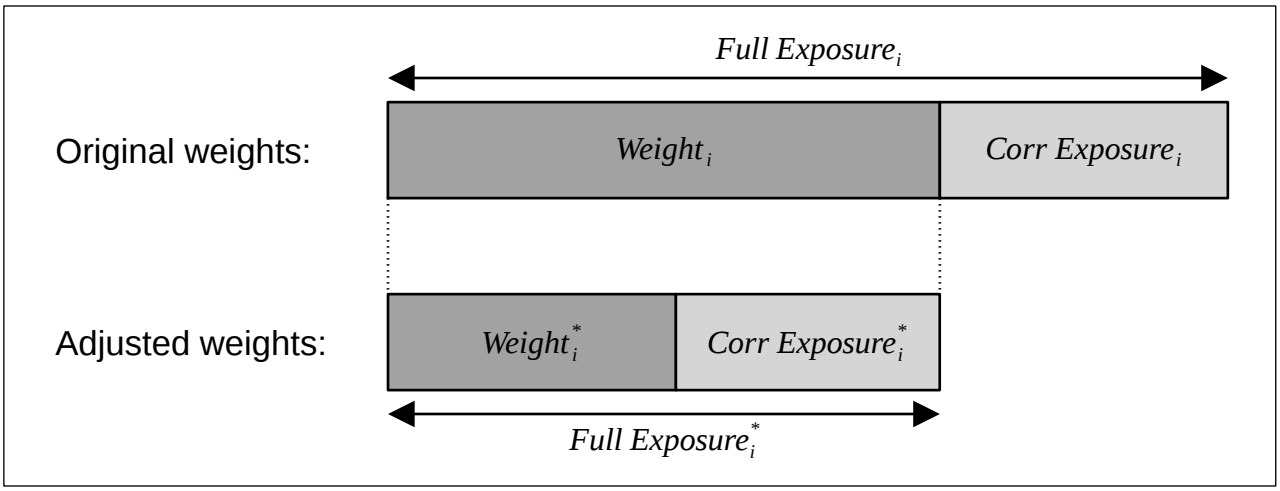

Figure 1: The concept of Full and Correlated Exposure, and how to adjust the portfolio weights.

The main paper shows how to solve this problem with a single calculation (see Section 8.6 of the main paper), by deriving the mathematical inverse of the Full Exposure from Eq. (5). But due to the self-referential nature of the Full Exposure, which is defined in terms of all the correlated portfolio weights, the Full Exposure of one asset changes whenever we change the weights of the other assets in the portfolio. This means we have to adjust the portfolio weights through several iterations, until they converge to the correct solution where Eq. (14) is satisfied for all assets in the portfolio.

3.1 General Algorithm

Sections 8.7 and 8.8 of the main paper show that it is possible to make a more general algorithm for solving this problem, which works for many other definitions of Full Exposure than the one in Eq. (5).

So that we can distinguish the updated portfolio weights for different iterations of the algorithm, let us denote the portfolio weight for Asset *i* in the *k*'th iteration of the algorithm as $Weight^*_{i,k}$, and the starting weight for iteration $k=1$ is denoted $Weight^*_{i,1}$. Similarly the Full Exposure that is calculated using $Weight^*_{i,k}$ is denoted $Full\ Exposure^*_{i,k}$.

The idea is then to update the portfolio weight $Weight^*_{i,k+1}$ by simply taking the portfolio weight from the previous iteration $Weight^*_{i,k}$ and multiply with the ratio between the originally desired $Weight_i$ and $Full\ Exposure^*_{i,k}$. So we are merely changing $Weight^*_{i,k}$ by how far $Full\ Exposure^*_{i,k}$ is from the desired $Weight_i$. This works when the Full Exposure has been defined in such a way, that any change to the portfolio weight $Weight^*_{i,k+1}$ will give a somewhat proportional change in $Full\ Exposure^*_{i,k+1}$, and changes to the portfolio weights for the other Assets *j* will only have a relatively minor impact on $Full\ Exposure^*_{i,k+1}$. The formula for updating each portfolio weight is:

$$Weight^*_{i,k+1} \;=\; Weight^*_{i,k} \;\cdot\; \frac{Weight_i}{Full\ Exposure^*_{i,k}} \tag{16}$$

The algorithm which uses this formula iteratively to update the portfolio weights $Weight^*_{i,k}$ until their $Full\ Exposure^*_{i,k}$ converges to the originally desired $Weight_i$ is as follows:

- Initialize the new weights by setting $Weight^*_{i,1} = Weight_i$
- Repeat the following for a given number of iterations from $k=1$ to some upper limit:
  - Calculate $Full\ Exposure^*_{i,k}$ from $Weight^*_{i,k}$ using either the dense algorithm from Section 2.2 or the sparse algorithm from Section 2.3, depending on the correlation matrix.
  - For each Asset *i* in the portfolio, calculate the new and updated $Weight^*_{i,k+1}$ as follows:

$$Weight^*_{i,k+1} \;=\; Weight^*_{i,k} \;\cdot\; \frac{Weight_i}{Full\ Exposure^*_{i,k}} \tag{17}$$

  - Terminate the computation early due to convergence if the following condition is met:

$$\mathrm{Max}\left(\left|Full\ Exposure^*_{i,k} - Weight_i\right|\right) \;<\; Required\ Precision \tag{18}$$

4 Example

Let us again consider the simple example from Section 2, where we have two Assets A and B with the originally desired portfolio weights $Weight_A = 9\%$ and $Weight_B = 12\%$. And let us again assume that their correlation is $\rho_{A,B} = 0.5$. We now want to find new portfolio weights using the algorithm in Section 3.1, so their Full Exposure is equal to their originally desired portfolio weights.

We initialize the new portfolio weights to equal the original weights as the starting point for the search:

$$\begin{aligned} Weight^*_{A,1} &= Weight_A = 9\% \\ Weight^*_{B,1} &= Weight_B = 12\% \end{aligned} \tag{19}$$

We then calculate the Full Exposure for these two weights using Eq. (5). Because the two weights and their correlation are all positive, we know from Eq. (6) that $Use_{A,B} = Use_{B,A} = 1$, so we have:

$$\begin{aligned} Full\,Exposure^*_{A,1} &= \sqrt{9\% \cdot 9\% + 9\% \cdot 12\% \cdot 0.5^2} \simeq 10.4\% \\ Full\,Exposure^*_{B,1} &= \sqrt{12\% \cdot 12\% + 12\% \cdot 9\% \cdot 0.5^2} \simeq 13.1\% \end{aligned} \tag{20}$$

We then update the portfolio weights using Eq. (16) from the algorithm in Section 3.1:

$$\begin{aligned} Weight^*_{A,2} &= Weight^*_{A,1} \cdot Weight_A / Full\,Exposure^*_{A,1} = 9\% \cdot 9\% / 10.4\% \simeq 7.8\% \\ Weight^*_{B,2} &= Weight^*_{B,1} \cdot Weight_B / Full\,Exposure^*_{B,1} = 12\% \cdot 12\% / 13.1\% \simeq 11.0\% \end{aligned} \tag{21}$$

We then calculate the Full Exposure for the adjusted portfolio weights using Eq. (5) again, which are now very close to the originally desired portfolio weights of $Weight_A = 9\%$ and $Weight_B = 12\%$:

$$\begin{aligned} Full\,Exposure^*_{A,2} &= \sqrt{7.8\% \cdot 7.8\% + 7.8\% \cdot 11.0\% \cdot 0.5^2} \simeq 9.1\% \\ Full\,Exposure^*_{B,2} &= \sqrt{11.0\% \cdot 11.0\% + 11.0\% \cdot 7.8\% \cdot 0.5^2} \simeq 11.9\% \end{aligned} \tag{22}$$

Let us try another iteration of the algorithm by inserting the updated weights into Eq. (16) again:

$$\begin{aligned} Weight^*_{A,3} &= Weight^*_{A,2} \cdot Weight_A / Full\,Exposure^*_{A,2} = 7.8\% \cdot 9\% / 9.1\% \simeq 7.7\% \\ Weight^*_{B,3} &= Weight^*_{B,2} \cdot Weight_B / Full\,Exposure^*_{B,2} = 11.0\% \cdot 12\% / 11.9\% \simeq 11.1\% \end{aligned} \tag{23}$$

We can repeat the above steps of the algorithm, to find portfolio weights whose Full Exposure get arbitrarily close to the originally desired portfolio weights. After only a few more iterations, the algorithm converges to approximately $Weight^*_A \simeq 7.72\%$ and $Weight^*_B \simeq 11.07\%$ whose Full Exposure is very close to the originally desired portfolio weights of $Weight_A = 9\%$ and $Weight_B = 12\%$.

This means that if the two assets are correlated with a coefficient of 0.5, and we want to invest 9% of the portfolio in Asset A and 12% of the portfolio in Asset B, then we should actually only invest about 7.72% of the portfolio in Asset A and only about 11.07% of the portfolio in Asset B, in order for the Full Exposure of Asset A to be 9%, and the Full Exposure of Asset B to be 12%, as originally desired. This is because the two assets are positively correlated, so an investment in one asset is also an indirect investment in the other asset through this correlation.

5 Time & Space Complexity

First we consider the time-complexity of the algorithm for calculating the Full Exposure. Let N be the number of assets in the portfolio. When the correlation matrix is dense so it contains the correlation coefficients $\rho_{i,j}$ for all pairs of Assets i and j, the correlation matrix has N^2 correlation coefficients. The dense algorithm from Section 2.2 iterates over the entire correlation matrix once, by having two nested for-loops that each iterate over all assets in the portfolio, so the dense algorithm has time-complexity $O(N^2)$. When the correlation matrix is sparse so it only contains M non-zero correlation coefficients, then the sparse algorithm from Section 2.3 has time-complexity $O(M+N)$ because it first iterates over all M non-zero correlations, and then it iterates over all N assets in the portfolio.

The algorithm for updating the portfolio weights from Section 3.1 performs K iterations, which first calculate the Full Exposure for all assets in the portfolio, either using the dense or sparse algorithm, and then it iterates over all the assets in the portfolio to update their weights. So when the correlation matrix is dense, the time-complexity of this entire algorithm is $O(K \cdot N^2)$, and when the correlation matrix is sparse, the time-complexity is $O(K \cdot (M+N))$.

The max number of iterations is typically set to some value like 100, but the algorithm in Section 3.1 actually converges much faster to the optimal solution, as shown in the main paper. Typically the number of iterations K is a fairly small number like 6 or 7, and it only depends on the required precision of the portfolio weights, and it does not depend on the number of assets in the portfolio.

So the time-complexity for the diversification algorithm from Section 3.1 is effectively around $O(N^2)$ for a dense correlation matrix, and it is effectively around $O(M+N)$ for a sparse correlation matrix.

The space-complexity is also $O(N^2)$ for the dense algorithm, as it needs storage for the entire correlation matrix with N^2 elements, but a proper implementation of the dense algorithm only needs additional temporary storage of $O(N)$ elements. The space-complexity is also $O(M+N)$ for the sparse algorithm, as it needs storage for the M non-zero correlation coefficients, and for the N portfolio weights, and a proper implementation only needs additional temporary storage of $O(N)$ elements.

Figure 1 compares the actual time-usage of the dense and sparse algorithms, with both serial and parallel execution, and for various degrees of sparsity for the correlation matrix. For most of these data-points, 100 trials were performed and the portfolio weights and correlation matrix were generated randomly. Although it is possible to make a parallel implementation of the sparse algorithm, it has not been done here. The experiment is run on a 4-core (8 thread) CPU with 2.6 GHz (3.5 GHz boost).

The experiment shows that the sparse algorithm is only faster than the dense algorithms, when the sparsity of the correlation matrix is very high, so most of the correlation coefficients are zero. And the parallel version of the dense algorithm is only faster than the serial version of the algorithm, when the portfolio has more than about 200 assets. For larger portfolios of 1000-2000 assets, the parallel algorithm is only about 2.5-3 times faster than the serial algorithm, even though there are 4 CPU cores, and that is probably because it is only some parts of the algorithm that are run in parallel.

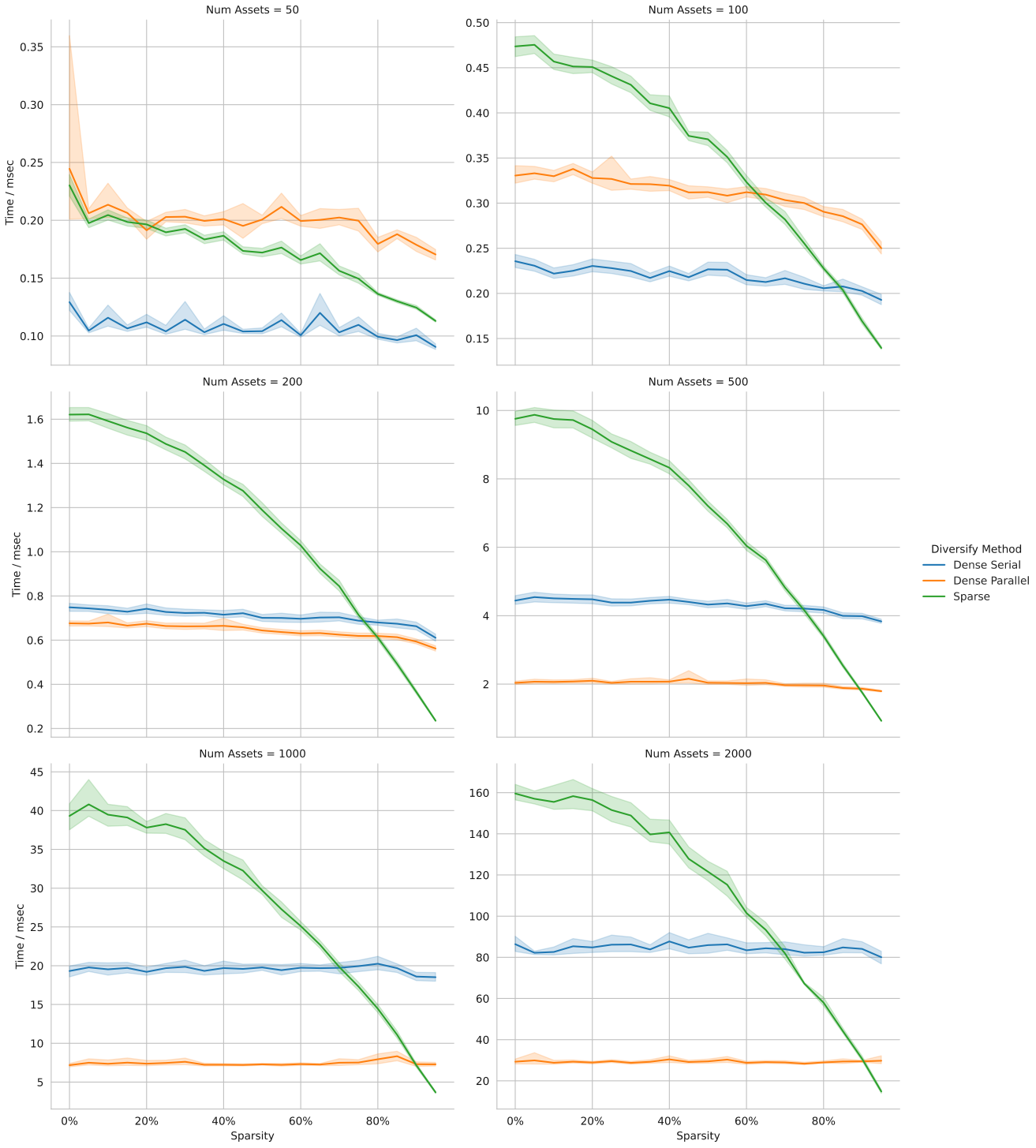

Figure 2: Comparison of time-usage for three versions of the diversification algorithm: Blue is for the dense algorithm from Section 2.2 with serial execution. Orange is for the same dense algorithm but with parallel execution. Green is for the sparse algorithm from Section 2.3 with serial execution. The shaded areas are the 99% confidence intervals of the time-usage.

6 Conclusion

This paper gave a concise description of a new algorithm for diversifying an investment portfolio that has been dubbed “Hvass Diversification” for easy reference, and which was previously described in much more detail in the main paper referenced below. In this paper we also showed how to modify the algorithm for a sparse correlation matrix, and we also compared the actual time-usage of the dense and sparse algorithms, as well as the parallel and serial versions of the dense algorithm.

The main paper contains a proof that the algorithm always converges to the optimal solution, and it also contains numerous tests on real-world stock-data, that show how effective the diversification algorithm is in practice, and how robust it is to very noisy and malformed correlation matrices.

7 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 in your own Python program to use the diversification algorithm from this paper. Instructions are found in the link. This may also serve as a reference implementation, if you want to implement the diversification algorithm in another programming language. The file named `diversify.py` contains the dense algorithm, and the file named `diversify_sparse.py` contains the sparse algorithm.
- InvestOps Tutorial shows how to use the diversification algorithm in a small Python program, which can also be run in the cloud using the free Google Colab service.

8 Bibliography

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

My previous papers and books can all be downloaded through SSRN and GitHub.