A Fast Algorithm for Computing High-dimensional Risk Parity Portfolios*

Théophile Griveau-Billion
Quantitative Research
Lyxor Asset Management, Paris
theophile.griveau-billion@lyxor.com

Jean-Charles Richard
Quantitative Research
Lyxor Asset Management, Paris
jean-charles.richard@lyxor.com

Thierry Roncalli
Quantitative Research
Lyxor Asset Management, Paris
thierry.roncalli@lyxor.com

September 2013

Abstract

In this paper we propose a cyclical coordinate descent (CCD) algorithm for solving high dimensional risk parity problems. We show that this algorithm converges and is very fast even with large covariance matrices ($n > 500$). Comparison with existing algorithms also shows that it is one of the most efficient algorithms.

Keywords: Risk parity, risk budgeting, ERC portfolio, cyclical coordinate descent algorithm, SQP algorithm, Jacobi algorithm, Newton algorithm, Nesterov algorithm.

JEL classification: G11, C60.

1 Introduction

In this paper, we focus on risk parity (or risk budgeting) portfolios. The underlying idea is to do allocation by risk, not by capital. In this case, the portfolio manager defines a set of risk budgets and then compute the weights of the portfolio such that the risk contributions match the risk budgets.

From a mathematical point of view, a risk budgeting (or RB) portfolio is defined as follows (Roncalli, 2013):

$$\begin{cases} \mathcal{RC}_i(x) = b_i \mathcal{R}(x) \\ b_i > 0 \\ x_i > 0 \\ \sum_{i=1}^n b_i = 1 \\ \sum_{i=1}^n x_i = 1 \end{cases}$$

*We are grateful to Florin Spinu for providing us his code on the Newton-Nesterov algorithm and for stimulating discussions on risk parity optimization.

where x_i is the weight of the asset i, $x = (x_1, \ldots, x_n)$ is the vector of portfolio weights and b_i is the risk budget of the asset i. $\mathcal{R}(x)$ is the risk measure of the portfolio x whereas $\mathcal{RC}_i(x)$ is the risk contribution of the asset i for the portfolio x.

A first route to solve the previous problem is to find the portfolio x such that:

$$\frac{\mathcal{RC}_i(x)}{b_i} = \frac{\mathcal{RC}_j(x)}{b_j}$$

This route has been explored by Maillard *et al.* (2010) in the case of the ERC portfolio[1]. They propose to minimize the sum of squared differences using the SQP algorithm. However, this algorithm is time-consuming and does not always converge for high-dimensional problem, i.e. when the number n of assets is larger than 200.

Another route is to consider the alternative optimization program (Roncalli, 2013):

$$\begin{aligned} y^\star &= \arg\min \mathcal{R}(y) \\ &\text{u.c.} \quad \begin{cases} \sum_{i=1}^n \ln y_i \geq c \\ y \geq \mathbf{0} \end{cases} \end{aligned} \tag{1}$$

where c is an arbitrary constant. The RB solution is then $x^\star = y^\star / \left(\mathbf{1}^\top y^\star\right)$ because of the budget constraint $\mathbf{1}^\top x = 1$. This second formulation has been used by Chaves *et al.* (2012) to define a Newton algorithm. In a recent paper, Spinu (2013) improves the convergence of the algorithm by noticing that the objective function is self-concordant. In this case, one can use the tools developed by Nesterov (2004). To our knowledge, the Newton algorithm was until now the best algorithm to solve high-dimensional risk parity portfolios.

In this paper, we present another competitive algorithm by noting that the optimization problem (1) is very standard. It is the minimization of a quadratic function with a logarithm barrier. That's why we consider a cyclical coordinate descent algorithm used in machine learning to solve regression problems with non-differentiable constraints. It appears that the method is very simple to implement and is very efficient to solve high-dimensional risk parity portfolios ($n > 250$).

2 Cyclical coordinate descent algorithm

The main idea behind the cyclical coordinate descent (CCD) algorithm is to minimize a function $f(x_1, \ldots, x_n)$ by minimizing only one direction at each step, whereas classical descent algorithms consider all the directions at the same time. In this case, we find the value of x_i which minimizes the objective function by considering the values taken by x_j for $j \neq i$ as fixed. The procedure repeats for each direction until the global minimum is reached. This method uses the same principles as Gauss-Seidel or Jacobi algorithms for solving linear systems.

Convergence of coordinate descent methods requires that $f(x)$ is strictly convex and differentiable. However, Tseng (2001) has extended the convergence properties to a non-differentiable class of functions:

$$f(x_1, ..., x_n) = f_0(x_1, ..., x_n) + \sum_{k=1}^m f_k(x_1, ..., x_n) \tag{2}$$

[1]The ERC (or equal risk contribution) portfolio is a special case of risk budgeting portfolios when the risk budgets are the same ($b_i = b_j = \frac{1}{n}$).

where f_0 is strictly convex and differentiable and the functions f_k are non-differentiable.

Some properties make this algorithm very attractive. First, it is very simple to understand and to implement. Moreover, the method is efficient for solving large scale problems. That's why it is used in machine learning theory for computing constrained regressions or support vector machine problems (Friedman *et al.*, 2010). A further advantage is that the method don't need stepsize descent tuning as opposed to gradient based methods.

Remark 1 *This algorithm has been already used in computing mean-variance optimization with norm constrained, because this problem is very close to the lasso regression (Yen and Yen, 2013).*

2.1 Derivation of the algorithm

Let us derive the algorithm in the case where the risk measure is the portfolio volatility. The Lagrangian function of the problem (1) is given by:

$$\mathcal{L}(x;\lambda) = \arg\min \sqrt{x^\top \Sigma x} - \lambda \sum_{i=1}^{n} b_i \ln x_i \tag{3}$$

Without loss of generality, we can fix $\lambda = 1$. The first-order conditions are

$$\frac{\partial\, \mathcal{L}(x;\lambda)}{\partial\, x_i} = \frac{(\Sigma x)_i}{\sigma(x)} - \frac{b_i}{x_i}$$

At the optimum, we have $\partial_{x_i} \mathcal{L}(x;\lambda) = 0$ or:

$$x_i \cdot (\Sigma x)_i - b_i \sigma(x) = 0$$

It follows that:

$$x_i^2 \sigma_i^2 + x_i \sigma_i \sum_{j \neq i} x_j \rho_{i,j} \sigma_j - b_i \sigma(x) = 0$$

By definition of the RB portfolio we have $x_i > 0$. We notice that the polynomial function is convex because we have $\sigma_i^2 > 0$. Since the product of the roots is negative[2], we always have two solutions with opposite signs. We deduce that the solution is the positive root of the second degree equation:

$$x_i^\star = \frac{-\sigma_i \sum_{j \neq i} x_j \rho_{i,j} \sigma_j + \sqrt{\sigma_i^2 \left(\sum_{j \neq i} x_j \rho_{i,j} \sigma_j\right)^2 + 4 b_i \sigma_i^2 \sigma(x)}}{2\sigma_i^2} \tag{4}$$

If the values of $(x_1, \cdots, x_n)$ are strictly positive, it follows that $x_i^\star$ is strictly positive. The positivity of the solution is then achieved after each iteration if the starting values are positive. The coordinate-wise descent algorithm consists in iterating the equation (4) until convergence.

Remark 2 *The convergence of the previous algorithm is obtained because the function (3) verifies the technical assumptions (B1)-(C2) necessary to apply Theorem 5.1 of Tseng (2001).*

Remark 3 *We notice that the algorithm is not well-defined if some risk budgets are set to zero. This enhances the specification of the risk budgeting problem with strictly positive values of b_i.*

[2] We have $-b_i \sigma_i^2 \sigma(x) < 0$.

We can improve the efficiency of the algorithm because some quantities may be easily updated. If we rewrite the equation (4) as follows:

$$x_i^\star = \frac{-(\Sigma x)_i + x_i\sigma_i^2 + \sqrt{((\Sigma x)_i - x_i\sigma_i^2)^2 + 4\sigma_i^2 b_i \sigma(x)}}{2\sigma_i^2}$$

we deduce that Σx and $\sigma(x)$ must be computed at each iteration of the algorithm. We note $x = (x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ and $\tilde{x} = (x_1, \ldots, x_{i-1}, x_i^\star, x_{i+1}, \ldots, x_n)$ the vector of weights before and after the update of the i^{th} weight x_i. Simple algebra show that:

$$\Sigma\tilde{x} = \Sigma x - \Sigma_{.,i}x_i + \Sigma_{.,i}\tilde{x}_i$$

and:

$$\sigma(\tilde{x}) = \sqrt{\sigma^2(x) - 2x_i\Sigma_{i,.}x + x_i^2\sigma_i^2 + 2\tilde{x}_i\Sigma_{i,.}\tilde{x} - \tilde{x}_i^2\sigma_i^2}$$

where $\Sigma_{i,.}$ and $\Sigma_{.,i}$ are the i^{th} row and column of Σ. Updating Σx and $\sigma(x)$ is then straightforward and reduces to the computation of two vector products. These operations dramatically reduce the computational time of the algorithm.

2.2 Extension to standard deviation-based risk measure

Roncalli (2013) considers the standard deviation-based risk measure:

$$\begin{aligned}\mathcal{R}(x) &= -\mu(x) + c\cdot\sigma(x)\\ &= -x^\top\mu + c\cdot\sqrt{x^\top\Sigma x}\end{aligned}$$

In this case, the update step of the cyclical coordinate descent algorithm becomes:

$$x_i^\star = \frac{-c\left(\sigma_i\sum_{j\neq i}x_j\rho_{i,j}\sigma_j\right) + \mu_i\sigma(x) + \sqrt{\left(c\left(\sigma_i\sum_{j\neq i}x_j\rho_{i,j}\sigma_j\right) - \mu_i\sigma(x)\right)^2 + 4cb_i\sigma_i^2\sigma(x)}}{2c\sigma_i^2}$$

3 Comparison of performances

In this section, we compare the efficiency of five algorithms: the SQP algorithm with BFGS update, the Jacobi algorithm, the Newton algorithm, the Nesterov algorithm and the CCD algorithm[3]. In order to obtain comparable results, we use the correlation matrix instead of the covariance matrix[4] and we assume that each algorithm is initialized with the equally-weighted portfolio. We also use the same convergence criterion. The algorithm is stopped when the normalized risk contributions satisfy the following inequality:

$$\sup_i\left(\mathcal{RC}_i^\star - b_i\right) \leq 10^{-8}$$

We consider the smart beta application of Cazalet *et al.* (2013) with the Eurostoxx 50 and S&P 500 indexes from December 1989 to December 2012. For each asset universe, we compute every month the ERC portfolio. In Tables 1 and 2, we report some statistics about the convergence and the computational time[5] (measured in hundredths of a second). p_s

[3] Details about these algorithms are provided in Appendix A

[4] Because scaling the weights by the volatilities gives the RB portfolio.

[5] The numerical tests are done with the programming language Gauss 10 and an Intel T8400 3 GHz Core 2 Duo processor and 3.5 GB RAM.

indicates the convergence frequency, $\bar{T}$ is the average computational time for each iteration whereas $T_{\max}$ is the maximum of the computational times. We notice that the SQP algorithm is the slowest algorithm and do not not converge for all the rebalancing dates. We also observe the same convergence problem for the Jacobi algorithm. Thus, it converges only 7 times out of ten in the case of the S&P 500 ERC backtesting. In terms of speed, the Jacobi method is the fastest method followed by the Newton algorithm when the number of assets is small ($n = 50$) and by the CCD algorithm when the number of assets is large ($n = 500$).

Table 1: Results with the Eurostoxx 50 ERC backtesting

Statistics	SQP	Jacobi	Newton	Nesterov	CCD
p_s	94.96	92.02	100.00	100.00	100.00
$\bar{T}$	2.00	0.04	0.04	0.05	0.12
$T_{\max}$	4.70	1.60	1.60	1.60	1.60

Table 2: Results with the S&P 500 ERC backtesting

Statistics	Jacobi	Newton	Nesterov	CCD
p_s	69.14	100.00	100.00	100.00
$\bar{T}$	1.15	17.90	38.78	4.33
$T_{\max}$	18.80	21.90	54.70	9.40

We now consider a second application. Using the algorithm of Davies and Higham (2000), we simulate correlation matrices such that the singular values are arithmetically distributed. The computation time of the ERC portfolio is reported in Table 3 for different values of the size n. We notice that the Jacobi algorithm does not converge. We also observe how the computational time of the Newton (or Nesterov) algorithm evolves with respect to the size of the universe due to the linear system equation. It follows that the CCD algorithm is more efficient than the Newton algorithm when the number of assets is larger than 250 (see Figure 1).

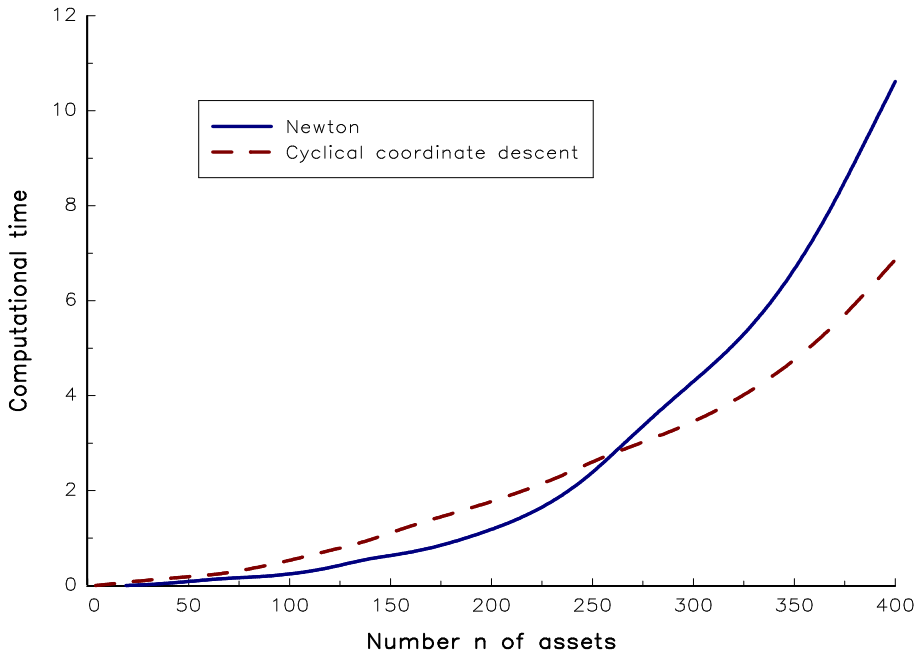
Table 3: Computational time with simulated correlation matrix

n	Jacobi	Newton	Nesterov	CCD
500	NC	24	37	13
1000	NC	215	384	45
1500	NC	790	1575	110

Remark 4 *The previous numerical results are sensitive to the programming language and the algorithm used to compute the Newton step:* $\Delta x_k = \left[\partial_x^2 f(x_k)\right]^{-1} \partial_x f(x_k)$. *For instance, it is better to solve the linear system* $\partial_x^2 f(x_k)\, \Delta x_k = \partial_x f(x_k)$ *using the Cholesky decomposition than computing the inverse of the symmetric positive definite matrix* $\partial_x^2 f(x_k)$[6]. *Moreover, the computational time of the CCD algorithm depends on the efficiency of the programming language in terms of loops. For instance, the Newton algorithm is faster than the CCD algorithm in Matlab and in R because of their poor implementation of loops. On*

[6] If we use the inverse matrix, the computational time of the Newton algorithm is 35.40 seconds instead of 7.90 seconds for the last example with $n = 1500$.

Figure 1: Computational time with respect to the size n



the contrary, the CCD algorithm is faster than the Newton algorithm in native programming languages (C or Fortran) and in Gauss.

References

[1] Bruder B. and Roncalli T. (2012), Managing Risk Exposures using the Risk Budgeting Approach, *SSRN*, `www.ssrn.com/abstract=2009778`.

[2] Cazalet Z., Grison P. and Roncalli T. (2013), The Smart Beta Indexing Puzzle, *SSRN*, `www.ssrn.com/abstract=2294395`.

[3] Chaves D.B., Hsu J.C., Li F. and Shakernia O. (2012), Efficient Algorithms for Computing Risk Parity Portfolio Weights, *Journal of Investing*, 21(3), pp. 150-163.

[4] Davies P.I. and Higham N.J. (2000), Numerically Stable Generation of Correlation Matrices and Their Factors, *BIT Numerical Mathematics*, 7(2), pp. 163-182.

[5] Friedman J., Hastie T. and Tibshirani R. (2010), Regularization Paths for Generalized Linear Models via Coordinate Descent, *Journal of Statistical Software*, 33(1), pp. 1-22.

[6] Maillard S., Roncalli T. and Teïletche J. (2010), The Properties of Equally Weighted Risk Contribution Portfolios, *Journal of Portfolio Management*, 36(4), pp. 60-70.

[7] Nesterov Y. (2004), *Introductory Lectures on Convex Optimization: A Basic Course*, Applied Optimization, 87, Kluwer Academic Publishers.

[8] Roncalli T. (2013), *Introduction to Risk Parity and Budgeting*, Chapman & Hall/CRC Financial Mathematics Series.

[9] Roncalli T. (2013), Introducing Expected Returns into Risk Parity Portfolios: A New Framework for Tactical and Strategic Asset Allocation, *SSRN*, `www.ssrn.com/abstract=2321309`.

[10] Spinu F. (2013), An Algorithm for the Computation of Risk Parity Weights, *SSRN*, `www.ssrn.com/abstract=2297383`.

[11] Tseng P. (2001), Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization, *Journal of Optimization Theory and Applications*, 109(3), pp. 475-494.

[12] Yen Y.M. and Yen T-S. (2013), Solving Norm Constrained Portfolio Optimization via Coordinate-Wise Descent Algorithms, *Computational Statistics and Data Analysis*, forthcoming.

A Existing algorithms

In what follows, we consider the standard risk parity approach when the risk measure $\mathcal{R}(x)$ is the portfolio volatility $\sigma(x)$. The original algorithms were developed to compute the ERC portfolio, but the extension to the RB portfolio is straightforward.

A.1 The SQP algorithm

Maillard *et al.* (2010) propose to compute the RB portfolio by considering the following optimization problem:

$$\begin{aligned} x^\star &= \arg\min \sum_{i=1}^n \sum_{j=1}^n \left(\frac{\mathcal{RC}_i(x)}{b_i} - \frac{\mathcal{RC}_j(x)}{b_j} \right)^2 \\ &\text{u.c.} \quad \mathbf{1}^\top x = 1 \quad \text{and} \quad \mathbf{0} \leq x \leq \mathbf{1} \end{aligned}$$

This convex problem can be solved using the sequential quadratic programming (or SQP) algorithm. However, this method may be improved by considering a slight modification of the objective function:

$$\begin{aligned} x^\star &= \arg\min \sum_{i=1}^n \left(\frac{x_i (\Sigma x)_i}{\sigma^2(x)} - b_i \right)^2 \\ &\text{u.c.} \quad \mathbf{1}^\top x = 1 \quad \text{and} \quad \mathbf{0} \leq x \leq \mathbf{1} \end{aligned}$$

In this case, we can compute analytically the associated gradient and Hessian matrices in order to accelerate the computation time.

A.2 The Jacobi algorithm

Let $\beta_i(x)$ be the beta of asset i with respect to the portfolio x. We have:

$$\beta_i(x) = \frac{(\Sigma x)_i}{\sigma^2(x)}$$

In the RB portfolio, the amounts of beta are proportional to the risk budgets:

$$x_i \beta_i (x) \propto b_i$$

Chaves *et al.* (2012) suggest to use the Jacobi power method to find the fixed point. It consists in iterating the previous formula:

$$x_{i,k+1} = \frac{b_i / \beta_i (x_k)}{\sum_{j=1}^{n} b_j / \beta_j (x_k)}$$

where k is the iteration index. Here, the $\beta_i (x_k)$ values are calculated with respect to the portfolio x_k and are used to compute the new weights x_{k+1}.

A.3 The Newton-Nesterov algorithm

Chaves *et al.* (2012) propose to apply the Newton method to the problem (1). However, the algorithm may have some difficulties to converge especially when the number of assets is large and the discrepancy between correlations is large. Spinu (2013) notices that the associated Lagrange function (3) is self-concordant and suggests to use the theory developed by Nesterov (2004) to improve the efficiency of the algorithm.

A.3.1 The Newton algorithm with self-concordant functions

Nesterov (2004) consider the following optimization problem:

$$x^* = \underset{x \in \operatorname{dom} f}{\arg\min} f (x)$$

when $f (x)$ is self-concordant[7]. Let us define $\lambda_f (x)$ as follows:

$$\lambda_f (x) = \sqrt{\partial_x f (x)^\top \left[\partial_x^2 f (x)\right]^{-1} \partial_x f (x)}$$

Nesterov (2004) shows that the solution of the problem exists and is unique if $\lambda_f (x) < 1$ and the Hessian is not degenerate. Moreover, he derives the region of the quadratic convergence for the Newton algorithm, which is defined as follows: $\lambda_f (x) < \lambda^*$ where $\lambda^* = \left(3 - \sqrt{5}\right)/2$. In this case, one can guarantee that $\lambda_f (x_{k+1}) < \lambda_f (x_k)$ where x_k is the Newton solution at the iteration k. Finally, the Newton algorithm applied to self-concordant functions becomes:

1. Damped phase
   While $\lambda_f (x_k) \geq \beta$ where $\beta \in [0, \lambda^*]$, we apply the following iteration:

   $$x_{k+1} = x_k - \frac{1}{1 + \lambda_f (x_k)} \Delta x_k$$

   where $\Delta x_k = \left[\partial_x^2 f (x_k)\right]^{-1} \partial_x f (x_k)$.

2. Quadratic phase
   When $\lambda_f (x_k) < \beta$, we apply the standard Newton iteration:

   $$x_{k+1} = x_k - \Delta x_k$$

[7] Let $\phi (x; t) = f (x + tu)$ where $x \in \operatorname{dom} f$ and $u \in \mathbb{R}^n$. The function is self-concordant if it verifies the technical property:

$$\left| D^3 f (x) [u, u, u] \right| \leq M_f \|u\|_{f''(x)}^{3/2}$$

where $D^3 f (x) [u, u, u] = \partial_t^3 \phi (x; t)$, $\|u\|_{f''(x)} = \sqrt{u^\top f'' (x) u}$ and M_f is a positive constant. The underlying idea of this technical property is to define objective functions for which there is no convergence problem.

A.3.2 Application to the risk parity problem

Spinu (2013) apply the previous algorithm to the Lagrange function[8] :

$$f(y) = \frac{1}{2} y^\top C y - \sum_{i=1}^{n} b_i \ln y_i \tag{5}$$

where C is the correlation matrix of asset returns. He deduces that the gradient and Hessian matrices are:

$$\begin{aligned} \partial_x f(y_k) &= C y_k - b y_k^{-1} \\ \partial_x^2 f(y_k) &= C + \operatorname{diag}\left(b y_k^{-2}\right) \end{aligned}$$

Moreover, Spinu (2013) proposes to replace $\lambda_f(y_k)$ by $\delta_f(y_k) = \|\Delta y_k / y_k\|_\infty$ in the Newton algorithm in order to reduce the computation time[9]. For the initial point x_0, he considers the scaled equally-weighted portfolio $x_0 = \left(\mathbf{1}^\top C \mathbf{1}\right)^{-1/2} \cdot \mathbf{1}$. Finally, we deduce the RB portfolio by rescaling the solution $y^\star$ by the volatilities:

$$x_i^\star = \frac{\sigma_i^{-1} y_i^\star}{\sum_{j=1}^{n} \sigma_j^{-1} y_j^\star}$$

The method proposed by Spinu (2013) improves the Newton method described in Chaves *et al.* (2012) when it fails to converge.

[8] We can set the Lagrange coefficient λ equal to 1 because of the scaling property of the RB portfolio (Roncalli, 2013).

[9] Spinu (2013) takes $\beta = 0.95 \times \lambda^*$.