Quantitative Finance

Multistage Financial Optimization

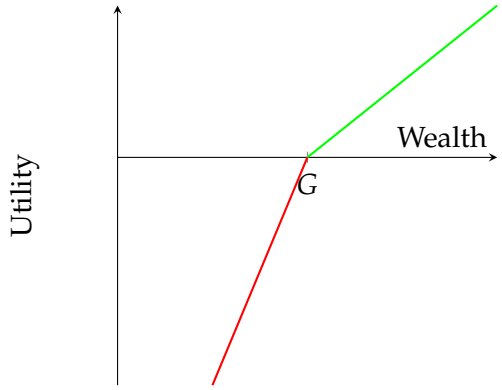
Patrick Hénaff

Version: 28 janv. 2024

Contents


```
library(foreach)
library(doParallel)
library(kableExtra)
library(linprog)
library(gtools)
library(pracma)
library(kernlab)
library(optiSolve)
library(piqp)
library(latex2exp)
```

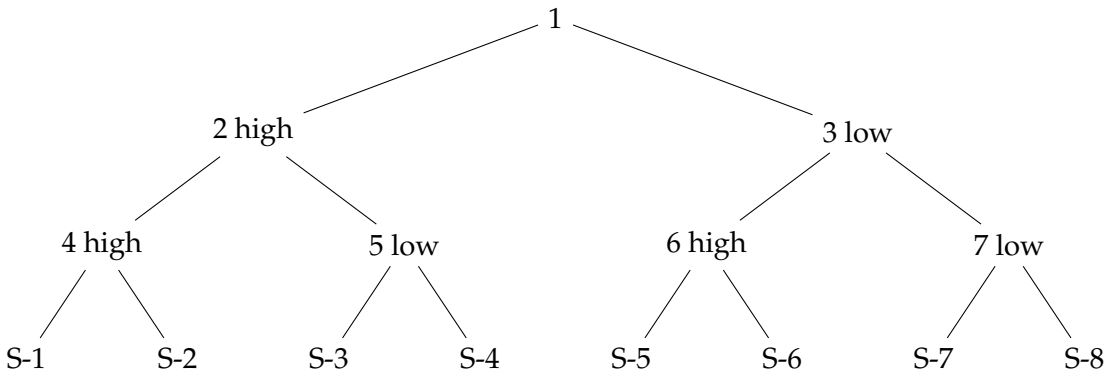
Consider the problem of managing a portfolio of stocks and bonds in order to secure a capital of G at horizon T (says, 15 years from now), starting from an initial budget W_0. At the investment horizon, any excess wealth over G will earn an annual return r%, while a shortfall will have to be funded at the cost of q% per year, such that $q > r$:



The investment horizon is divided into 3 periods, and the investment portfolio may be adjusted at the start of each period. The returns of the assets are uncertain, and may follow one out of two scenarios during each period. The possible scenarios are identical in each time interval, and have equal probability.

scenario	Stock	Bond	Prob
High	1.25	1.14	0.5
Low	1.06	1.12	0.5

In total, we have 8 possible scenarios, labeled $S-1$ to $S-8$ for the 3 time periods:



Given this information, what should the optimal investment policy be like?

We first consider a formulation named the *extensive form* of the stochastic problem, because it considers explicitely all possible scenarios that may unfold in the future periods.

1 The stochastic problem in extensive form

Let x_S^i and x_B^i be the wealth invested is stock and bond, in state i. The uncertain returns in that state may take the values (r_S^+, r_B^+) or (r_S^-, r_B^-). We write a set of accounting identities to connect the time intervals of the investment process:

At the initial stage, the available budget W_0 must be allocated between the stock and the bond:

$$x_S^1 + x_B^1 = W_0$$

There are two scenarios for the next time step. Again, the wealth under each scenario must be reallocated between the stock and the bond:

$$\begin{aligned}(1+r_S^+)x_S^1 + (1+r_B^+)x_B^1 &= x_S^2 + x_B^2\\ (1+r_S^-)x_S^1 + (1+r_B^-)x_B^1 &= x_S^3 + x_B^3\end{aligned}$$

and so forth at the following time step. Finally, there are 8 distinct scenarios at the investment horizon, and we compute the utility of the wealth under each scenario.

$$\begin{aligned}(1+r_S^+)x_S^4 + (1+r_B^+)x_B^4 &= G + v^1 - w^1\\ (1+r_S^-)x_S^4 + (1+r_B^-)x_B^4 &= G + v^2 - w^2\\ (1+r_S^+)x_S^5 + (1+r_B^+)x_B^5 &= G + v^3 - w^3\\ (1+r_S^-)x_S^5 + (1+r_B^-)x_B^5 &= G + v^4 - w^4\\ (1+r_S^+)x_S^6 + (1+r_B^+)x_B^6 &= G + v^5 - w^5\\ (1+r_S^-)x_S^6 + (1+r_B^-)x_B^6 &= G + v^6 - w^6\\ (1+r_S^+)x_S^7 + (1+r_B^+)x_B^7 &= G + v^7 - w^7\\ (1+r_S^-)x_S^7 + (1+r_B^-)x_B^7 &= G + v^8 - w^8\end{aligned}$$

The objective function to be maximized is the expected utility of the investment policy at horizon T:

$$\max \sum_{i=1}^{8} (qv^i - rw^i)$$

1.1 Solution

The LP problem involves 30 variables, 15 accounting identities constraints and 30 non-negative constraints on all the variables.

```
Amat <- matrix(0, nrow=15, ncol=30)

sce_up = c(1.25, 1.14)
sce_down = c(1.06, 1.12)
sce = rbind(sce_up, sce_down)

Amat[1,1:2] = -1
Amat[2,1:2] = sce_up; Amat[2,3:4] = -1
Amat[3,1:2] = sce_down; Amat[3,5:6] = -1
Amat[4,3:4] = sce_up; Amat[4,7:8] = -1
Amat[5,3:4] = sce_down; Amat[5,9:10] = -1
Amat[6,5:6] = sce_up; Amat[6,11:12] = -1
```

```
Amat[7,5:6] = sce_down; Amat[7,13:14] = -1

Amat[8,7:8] = sce_up; Amat[8,15:16] = c(1, -1)
Amat[9,7:8] = sce_down; Amat[9,17:18] = c(1, -1)
Amat[10,9:10] = sce_up; Amat[10,19:20] = c(1, -1)
Amat[11,9:10] = sce_down; Amat[11,21:22] = c(1, -1)
Amat[12,11:12] = sce_up; Amat[12,23:24] = c(1, -1)
Amat[13,11:12] = sce_down; Amat[13,25:26] = c(1, -1)
Amat[14,13:14] = sce_up; Amat[14,27:28] = c(1, -1)
Amat[15,13:14] = sce_down; Amat[15,29:30] = c(1, -1)

bvec = matrix(0, nrow=15, ncol=1)

W.0 = 55000
G = 80000
r_short = 4/100
r_long = 1/100
prob = 1/8

bvec[1,1] = -W.0
bvec[8:15,1] = G
cvec = matrix(0, nrow=1, ncol=30)
cvec[1, seq(15,30,by=2)] = -r_short * prob
cvec[1, seq(16,30,by=2)] = r_long * prob

bvec = as.vector(bvec)
cvec = as.vector(cvec)

res = solveLP(cvec, -bvec, -Amat, maximum=TRUE,verbose=1)

sol = matrix(0, nrow=7, ncol=3)
for(i in seq(7)) {
  sol[i,] = c(i, res$solution[(2*i-1):(2*i)])
}
colnames(sol) <- c("Node", "Stock", "Bond")
```

The excess or shortfall at horizon T for each scenario is displayed in Table 3.

1.2 Expected Value Solution

In order to appreciate the contribution of the stochastic model, we may consider a deterministic model where the decision is based on the expected return. Since the expected return of the stock (1.155) is higher than the expected return of the bond (1.13), the entire budget will be allocated to the stock. Table 4 shows the outcome of the strategy for the 8 scenarios, compared to the result of the stochastic optimization.

Table 2: Optimal asset allocation at each node in the 3-period tree

Node	Stock	Bond
1	41479	13521
2	65095	2168
3	36743	22368
4	83840	0
5	0	71429
6	0	71429
7	64000	0

Table 3: Shortfall or excess by scenario

Scenario	Shortfall	Excess
1	0	24800
2	0	8870
3	0	1429
4	0	0
5	0	1429
6	0	0
7	0	0
8	12160	0

The objective value may be interpreted as the expected annual cost of funding the shortfall.

```
expected.cost.ev = mean( -df$W1.Short*r_short + df$W1.Long*r_long)
expected.cost.sp = mean( -df$W2.Short*r_short + df$W2.Long*r_long)
VSS = expected.cost.sp - expected.cost.ev
```

For the stochastic solution, this value is -15.14 and -37.88 for the deterministic solution. The difference between the two, 22.74 is the value of recourse solution.

Another point of comparison is to consider the probability of reaching the goal G. The deterministic model reaches the goal 50% of the time, while the stochastic formulation succeeds in 7 out of 8 scenarios.

On all counts, the multi-stage model clearly adds value, compared to a simpler model that optimizes the expected value at each time step. This advantage, however, comes at the cost of a very large optimization problem, with one variable per state, time step and asset. We consider next various methods for simplifying the optimization problem. It turns out that the problem has a special structure that provides opportunities for simplification. In order to exploit such structure, however, a quick review of optimization theory is in order.

Table 4: Expected Value vs. Recourse solutions of the asset allocation problem.

	EV solution		Recourse solution	
Scenario	Shortfall	Surplus	Shortfall	Surplus
1		27422		24800
2		11094		8870
3		11094		1429
4	2752			
5		11094		1429
6	2752			
7	2752			
8	14494		12160	

2 Review of optimization theory

2.1 Duality

2.2 Augmented Lagrangian

For the sake of simplicity, we restrict the discussion to programs with equality constraints. Faced with such program, the idea is to try to simplify the problem by incorporating the constraints in the objective function.

Consider the problem

$$\begin{aligned} &\min \ f(x) \\ &\text{s.t.} \\ &\quad g(x) = 0 \end{aligned}$$

A first idea is to introduce the constraint as a quadratic penalty in the objective function:

$$\min \ f(x) + \frac{1}{2}\rho g(x)^T g(x)$$

In practice, this formulation is not satisfactory since ρ may need to be very large in order to insure feasibility.

A second approach is to consider the Lagrangian $\mathcal{L}(x, \lambda) = f(x) - \lambda^T g(x)$. The first order necessary condition is $\nabla \mathcal{L}(x, \lambda) = 0$. But the solution (x^*, λ^*) is in general a saddle point of $\mathcal{L}(x, \lambda)$ and identifies both the minimum and the maximum of the program.

The basic idea of the augmented Lagrangian method is to combine the two previous ideas:

- Solve the problem

$$\min \ f(x) + \lambda^T g(x) + \frac{1}{2}\rho g(x)^T g(x)$$

where the presence of the multiplier mitigates the need for a very large ρ.

- Optimize the above expression over x only, and implement an iterative procedure to make λ converge to the solution λ^* corresponding to the *minimum*. With λ fixed, the solution is a local minimizer rather than a saddle point.

Let's now consider how to determine λ. In an iterative procedure, with λ and ρ set to values λ_k and ρ_k, we solve

$$\min_x \ \phi_k(x) = f(x) + \lambda_k^T g(x) + \frac{1}{2}\rho_k g(x)^T g(x)$$

the first order necessary condition is

$$\begin{aligned} 0 = \nabla\phi_k(x_{k+1}) &= \nabla f(x_{k+1}) + \nabla g(x_{k+1})\lambda_k + \rho_k \nabla g(x_{k+1}) g(x_{k+1}) \\ \nabla f(x_{k+1}) &= \nabla g(x_{k+1}) \left[\lambda_k - \rho_k g(x_{k+1})\right] \end{aligned}$$

We also want the next iteration to enforce the multiplier condition

$$\nabla f(x_{k+1}) = \nabla g(x_{k+1})\lambda_{k+1}$$

which provides the updating formula for λ:

$$\lambda_{k+1} = \lambda_k + \rho_k g(x_{k+1})$$

For the time being, we will take ρ as fixed. The complete algorithm is therefore:

Algorithm 1: Augmented Lagrangian algorithm

Input : ρ, tol, $k_{\max}$
Output: x^*

$k < -0; \lambda_k < -0$
while $k < k_{max}$ *and* $t > tol$ **do**
  $x_{k+1} < -\underset{x}{\operatorname{argmin}} \phi_k(x)$
  $\lambda_{k+1} < -\lambda_k + \rho g(x_{k+1})$
  $k < -k + 1$
  $t_1 < -||\lambda_{k+1} - \lambda_k||$
  $t_2 < -||g(x_{k+1}||$
  $t < -t_1 \text{and} t_2$
end

3 Progressive Hedging

We introduce the principle of progressive hedging with a stochastic problem with simple recourse.

$$\begin{aligned} \min \quad & f^1(x) + Q(x) \\ \text{s.t.} \quad & g^1(x) = 0 \end{aligned}$$

with:

$$\begin{aligned} Q(x) &= E_w[Q(x,w)] \\ Q(x,w) =& \min_y \quad f^2(x, y(w), w) \\ & \text{s.t.} \quad g^2(x, y(w), w) = 0 \end{aligned}$$

The uncertainty regarding the second stage is modelled by K scenarios, so that the problem may be written:

$$\begin{aligned} \min \quad & f^1(x) + \sum_{k=1}^{K} (\pi_k f^2(x, y_k)) \\ \text{s.t.} \quad & g^1(x_k) = 0 \ \ \forall k \\ & g^2(x, y_k) = 0 \ \ \forall k \end{aligned}$$

or,

$$\begin{aligned} \min & \sum_{k=1}^{K} \pi_k (f^1(x_k) + f^2(x, y_k)) \\ \text{s.t.} \quad & g^1(x) = 0 \\ & g^2(x, y_k) = 0 \quad \forall k \\ & x_k - \hat{x} = 0 \quad \forall k \end{aligned}$$

Let's now use the augmented lagrangian formulation to move the constraints $x_k - \hat{x} = 0$ in the objective function. The problem becomes:

$$\begin{aligned} \min & \sum_{k=1}^{K} \pi_k \left(f^1(x_k) + f^2(x, y_k)) + \lambda_k^T (x_k - \hat{x}) + \frac{1}{2} \rho ||x_k - \hat{x}|| \right) \\ \text{s.t.} \quad & g^1(x) = 0 \\ & g^2(x, y_k) = 0 \quad \forall k \end{aligned}$$

and the problem can be split into K subproblems of the form

$$\begin{aligned} \min & f^1(x_k) + f^2(x, y_k)) + \lambda_k^T (x_k - \hat{x}) + \frac{1}{2} \rho ||x_k - \hat{x}|| \\ \text{s.t.} \quad & g^1(x_k) = 0 \\ & g^2(x_k, y_k) = 0 \end{aligned} \tag{1}$$

The progressive hedging algorithm is almost identical to the original augmented lagrangian method; the only difference being that the minimization problem over x is split into K independent minimization problems.

Algorithm 2: Progressive Hedging algorithm

Input : ρ, tol, $i_{\max}$
Output: x^*

$i < -0; \lambda_i < -0;$ converged $< -False$
while $k < k_{max}$ *and* !*converged* **do**
  Solve the K subproblems 1 to obtain $x_k^{i+1}, k = 1, \ldots, K$
  Compute $\hat{x}^{i+1} = \sum_{k=1}^{K} \pi_k x_k^{i+1}$
  Update the multipliers: $\lambda_k^{i+1} = \lambda_k^i + \rho(x_k^{i+1} - \hat{x}^{i+1})$
  $i < -i + 1$
  $t_1 < -||\lambda^{i+1} - \lambda^i|| < \text{tol}_1$
  $t_2 < -||g^1(x^{i+1}|| < \text{tol} < \text{tol}_2$
  $t_3 < -||g^2(x^{i+1}|| < \text{tol} < \text{tol}_3$
  converged $< -t_1$ andt_2 andt_3
end

3.1 Introductory example

Consider an asset allocation problem where the initial wealth is 10,000$, and the goal is to reach a wealth of 25,000$ next period. The uncertain return is modelled with two scenarios:

Table 5: Asset return by scenario.

Scenario	Asset A	Asset B
1	0%	300%
2	400%	200%

In each scenario, a shortfall below the 25,000$ goal will be subject to a quadratic penalty. There is no reward for exceeding the goal. Given this objective, what should be the allocation between assets A and B?

The extensive form of the stochastic optimization problem is a quadratic problem with linear constraints:

$$
\begin{aligned}
\min \ & \frac{1}{2}(y_1^2 + y_2^2) \\
\text{s.t.} \\
& x_A + x_B <= 10 \\
& x_A + 3x_B + y_1 >= 25 \\
& 4x_A + 2x_B + y_2 >= 25 \\
& x_A, x_B, y_1, y_2 >= 0
\end{aligned}
$$

The problem is readily solved to yield the optimal allocation in assets A and B. Note however that the matrix in the quadratic term is not positive definite: the quadratic term only involves variables y_1 and y_2, while the problem has 4 variables in total. As a result, the solver must chosen with care.

```
P <- matrix(0, nrow=4, ncol=4)
P[3,3] = 1
P[4,4] = 1
c <- rep(0, 4)
G <- matrix(c(1, 1, 0, 0,
              -1, -3, -1, 0,
              -4, -2, 0, -1), nrow=3, byrow=TRUE)
h <- c(10, -25, -25)
x_lb <- rep(0, 4)
sol <- solve_piqp(P=P, c=c, G=G, h=h, x_lb=x_lb )
```

The solution is $x_A = 2.5$, $x_B = 7.5$.

3.1.1 Progressive hedging solution

Initialize $w = 0$, let $x_1^0 = (x_{1A}^0, x_{1B}^0) = (0, 10)$, $x_2^0 = (x_{2A}^0, x_{2B}^0) = (10, 0)$. The corresponding value of $\hat{x}^0 = (5, 5)$.

The problem to be solved at the first iteration is:

$$\begin{aligned}
\min\ & \frac{1}{2}(y_1^2 + y_2^2 + \sum_{k=1}^{2} ||x_k^0 - \hat{x}^0||^2) \\
\text{s.t.}\ & \\
& x_{kA} + x_{kB} <= 10 \quad k = 1, \ldots, 2 \\
& x_{1A} + 3x_{1B} + y_1 >= 25 \\
& 4x_{2A} + 2x_{2B} + y_2 >= 25 \\
& x_{1A}, x_{1B}, x_{2A}, x_{2B}, y_1, y_2 >= 0
\end{aligned}$$

This problem separates into two sub problems of the type

$$\begin{aligned}
\min\ & \frac{1}{2}\left[\begin{pmatrix} x_{1A}^1 \\ x_{1B}^1 \\ y_1 \end{pmatrix}^T \begin{pmatrix} x_{1A}^1 \\ x_{1B}^1 \\ y_1 \end{pmatrix} + \begin{pmatrix} x_{1A}^1 \\ x_{1B}^1 \\ y_1 \end{pmatrix}^T \begin{pmatrix} -10 \\ -10 \\ 0 \end{pmatrix} \right] \\
\text{s.t.}\ & \\
& x_{1A}^1 + x_{1B}^1 <= 10 \\
& x_{1A}^1 + 3x_{1B}^1 - y_1 >= 25 \\
& x_{1A}^1, x_{1B}^1, y_1 >= 0
\end{aligned}$$

which are solved below. This first subproblem is

```
P <- diag(3)
c <- c(-5, -5, 0)
G <- matrix(c(1, 1,  0,
              -1, -3, -1), nrow=2, byrow=TRUE)
```

```
h <- c(10, -25)
x_lb <- rep(0, 3)
sol.1 <- solve_piqp(P=P, c=c, G=G, h=h, x_lb=x_lb )
```

with yields the solution $x^1_{1A} = 3.33$ and $x^1_{1B} = 6.67$

The second subproblem is solved similarly:

```
P <- diag(3)
c <- c(-5, -5, 0)
G <- matrix(c(1, 1,  0,
              -4, -2, -1), nrow=2, byrow=TRUE)
h <- c(10, -25)
x_lb <- rep(0, 3)
sol.2 <- solve_piqp(P=P, c=c, G=G, h=h, x_lb=x_lb )
```

with yields the solution $x^1_{2A} = 5$ and $x^1_{2B} = 5$.

The average solution after this iteration is thus:

$$\hat{x}^1 = \frac{1}{2}\left[\begin{pmatrix}3.33\\6.67\end{pmatrix} + \begin{pmatrix}5\\5\end{pmatrix}\right] = \begin{pmatrix}4.17\\5.83\end{pmatrix}$$

The multiplier λ associated with each subproblem are finally updated, and the next iteration can now be computed.

```
x.hat = c(mean(sol.1$x[1], sol.2$x[1]),
mean(sol.1$x[2], sol.2$x[2]))
w.1 = 0
w.2 = 0
r = 2
w.1 = w.1 + r * (sol.1$x[1:2] - x.hat)
w.2 = w.2 + r * (sol.2$x[1:2] - x.hat)
```

We provide below a program to perform the entire progressive hedging algorithm. First, we define a function that solves the generic subproblem, where $x_k = (x_{kA}, x_{kB})$ and $k = 1, 2$.

$$\begin{aligned}\min\ & y_k^2 + w^T(x_k - \hat{x}_k) + \frac{r}{2}||x_k - \hat{x}_k||^2\\ \text{s.t.}\\ & x_{kA} + x_{kB} <= 10\\ & q_{kA}x_{kA} + q_{kB}x_{kB} + y_k >= 25\\ & x_{kA}, x_{kB}, y_k, >= 0\end{aligned}$$

```
sub.prob <- function(x.hat, r, w, q) {
P <- zeros(3,3)
P[1,1] <- r
P[2,2] <- r
```

```
P[3,3] <- 2
c <- c(w - r*x.hat, 0)
G <- matrix(c(1, 1, 0,
              -q[1], -q[2], -1), nrow=2, byrow = TRUE)
h <- c(10, -25)
x_lb <- rep(0,3)
sol <- solve_piqp(P=P, c=c, G=G, h=h, x_lb = x_lb)
sol
}
```

For verification, we reproduce the calculation of the first iteration for subproblem 1:

```
x.hat <- c(5,5)
r <- 2
w <- c(0,0)
q <- c(1,3)
sol <- sub.prob(x.hat, r, w, q)
print(paste("x_A:", round(sol$x[1],2),
"x_B:", round(sol$x[2],2), sep=" "))
```

```
## [1] "x_A: 3.33 x_B: 6.67"
```

The algorithm is initialized by solving each subproblem with perfect foresight:

```
q = matrix(c(1,3,4,2), nrow=2, byrow=TRUE)
w = matrix(0, nrow=2, ncol=2)
x.hat <- c(0,0)
x = matrix(0, nrow=2, ncol=2)
tol <- 1.0e-2
x[1,] <- sub.prob(x.hat, r=0, w[1,], q[1,])$x[1:2]
x[2,] <- sub.prob(x.hat, r=0, w[2,], q[2,])$x[1:2]
x.hat <- colMeans(x)

conver <- FALSE
iter.count <- 0
iter.max <- 100
r = 2
x.iter <- matrix(0, nrow=iter.max, ncol=2)

while(!conver & (iter.count < iter.max)) {
iter.count <- iter.count + 1
x.iter[iter.count,] = x.hat
x[1,] <- sub.prob(x.hat, r, w[1,], q[1,])$x[1:2]
x[2,] <- sub.prob(x.hat, r, w[2,], q[2,])$x[1:2]


x.hat.old <- x.hat
```

```
x.hat <- colMeans(x)
w.old <- w
w[1,] = w[1,] + r * (x[1,] - x.hat)
w[2,] = w[2,] + r * (x[2,] - x.hat)

# convergence test
test.1 <- norm(x.hat-x.hat.old, type="2")
test.2 <- norm(w - w.old, type="2")

conver <- (test.1 <= tol) & (test.2 <= tol)
}
```

The progress of the algorithm is illustrated by the following graph.

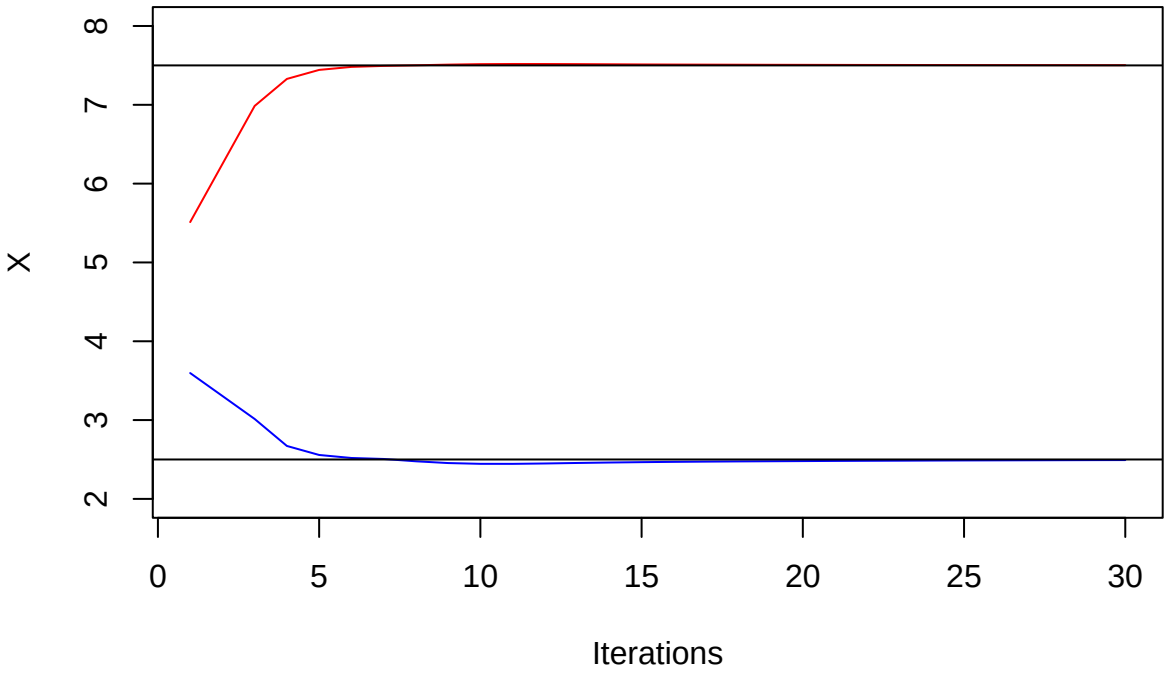


Figure 1: Progressive Hedging Iterations

3.2 Second example: A 3-stage optimization problem

The progressive hedging algorithm is particularly valuable when considering numerous stages, which translates into many scenarios. At each iteration, the corresponding subproblems may be solved in parallel. This of course is only beneficial when the subproblem is sufficiently time consuming. Nevertheless, as a proof of principle, we next revisit the 3 stage asset allocation problem of section 1, with a parallel solution of the sub problems.

Similarly to the previous example, The subproblem is

$$\min \quad -v^- M_k + v^+ P_k + w^T(x_k - \hat{x}_k) + \frac{r}{2}||x_k - \hat{x}_k||^2$$

s.t.

$$\begin{aligned}
& x^1_{kA} + x^1_{kB} = W_0 \\
& q^1_{kA} x^1_{kA} + q^1_{kB} x^1_{kB} - x^2_{kA} - x^2_{kB} = 0 \\
& q^2_{kA} x^2_{kA} + q^2_{kB} x^2_{kB} - x^3_{kA} - x^3_{kB} = 0 \\
& q^3_{kA} x^3_{kA} + q^3_{kB} x^3_{kB} + M_k - P_k = G \\
& x^{1,2,3}_{kA}, x^{1,2,3}_{kB}, P_k, M_k >= 0
\end{aligned}$$

with $x_k = (x^1_{kA}, x^1_{kB}, x^2_{kA}, x^2_{kB}, x^3_{kA}, x^3_{kB})$

```
nb.vars = 8
A <- matrix(0, nrow=4, ncol=nb.vars)
rownames(A) = c("t_0", "t_1", "t_2", "t_3")
colnames(A) = c("xA.1", "xB.1","xA.2", "xB.2","xA.3", "xB.3","M", "P")
W.0 = 55000
G = 80000
b = c(W.0, 0, 0, G)
r_short = 4
r_long = 1
tol <- 1

pha.sub <- function(sce, r, w, x.hat, r_short, r_long, b) {
  nb.vars=8
  nb.x = 6
A <- matrix(0, nrow=4, ncol=nb.vars)
A[1,1:2] = 1
A[2,1:2] = sce[1,]; A[2,3:4] = -1
A[3,3:4] = sce[2,]; A[3,5:6] = -1
A[4,5:6] = sce[3,]; A[4,7:8] = c(1, -1)
Q = pracma::zeros(nb.vars, nb.vars)
for(i in seq(nb.x)) {
  Q[i,i] = r
}
c = c(w - r * x.hat, r_short, -r_long)
x_lb = rep(0, nb.vars)
piqp::solve_piqp(P=Q, c=c, A=A, b=b, x_lb=x_lb)
}
```

As a verification, we solve the first scenario with perfect foresight. As expected,the optimal decision is to invest the initial budget in asset A, which has higher return than asset B in all three stages.

```
sce = rbind(sce_up, sce_up, sce_up)
nb.x = 6
nb.vars = 8
```

```
nb.scen = 8

x.hat = (55000/2)*rep(1, nb.x)
r = 0
w = rep(0, nb.x)
sol = pha.sub(sce, r, w, x.hat, r_short, r_long, b)
```

The initial allocation is $(x_A, x_B) = (5.5 \times 10^4, 0$

With 3 stages, the non-anticipative constraint is a bit more complex:

- At the initial stage, the allocation must be identical for all 8 scenarios.
- At stage 1, the allocation must be identical for scenarios 1 to 4, and 5 to 8.
- At stage 2, the scenarios bundles are (1,2), (3,4), (5,6), (7,8)

The following function implements the calculation of $\hat{x}$ for all stages and bundles of scenarios.

```
scen = rbind(sce_up, sce_down)
m.scen = as.matrix(expand.grid(stage.2=c(1,2), stage.1=c(1,2), stage.0=c(1,2)))
m.scen = m.scen[,c(3,2,1)]

mean.x <- function(scen.list, i.var) {
  # Compute the mean value of variable i.var over the bundle of scenarios
  # scen.list
  tmp = vector()
  for(i in scen.list) {
    tmp = c(tmp, all_sol[[i]]$x[i.var])
  }
  mean(tmp)
}

calc.x.hat <- function() {
x.hat = matrix(0, nrow=8, ncol=6)
# stage 0
s = seq(8)
x.hat[s,1] = mean.x(s, 1)
x.hat[s,2] = mean.x(s, 2)
# stage 1
s = seq(4)
x.hat[s,3] = mean.x(s, 3)
x.hat[s,4] = mean.x(s, 4)
s = as.integer(c(5,6,7,8))
x.hat[s,3] = mean.x(s, 3)
x.hat[s,4] = mean.x(s, 4)
# stage 2
s = c(1,2)
x.hat[s,5] = mean.x(s, 5)
```

```
x.hat[s,6] = mean.x(s, 6)
s = c(3,4)
x.hat[s,5] = mean.x(s, 5)
x.hat[s,6] = mean.x(s, 6)
s = c(5,6)
x.hat[s,5] = mean.x(s, 5)
x.hat[s,6] = mean.x(s, 6)
s = c(7,8)
x.hat[s,5] = mean.x(s, 5)
x.hat[s,6] = mean.x(s, 6)
x.hat
}
```

One of the advantages of the progressive hedging algorithm is to enable a distributed computation of the algorithm, since all the subproblems may be solved in parallel. We take advantage of this feature in the following code:

```
# Detect the number of available cores
nb.available.cores <- parallel::detectCores()
nb.cores <- 8
print(paste("Using ", nb.cores, " cores out of ", nb.available.cores))
```

```
## [1] "Using  8  cores out of  32"
```

```
# Create a cluster and register
cluster <- parallel::makeCluster(nb.cores, type="PSOCK")
doParallel::registerDoParallel(cl=cluster)
```

```
# Initialize with perfect foresight
w = matrix(0, nrow=8, ncol=6)
r = 0
x.hat = matrix(0, nrow=8, ncol=6)

run.subproblems <- function(foo, scen, m.scen, r,w,x.hat, r_short, r_long, b) {

all_sol <- foreach(i = seq(nrow(m.scen))) %dopar% {
  sce = scen[m.scen[i,],]
  foo(sce, r, w[i,], x.hat[i,], r_short, r_long, b)
}
all_sol
}

all_sol <- run.subproblems(pha.sub, scen, m.scen, r, w, x.hat,
r_short, r_long, b)
x.hat <- calc.x.hat()

conver <- FALSE
```

```
iter.count <- 0
iter.max <- 2000
r = .00002
x.iter = matrix(0, nrow=iter.max, ncol=4)
while(!conver & (iter.count < iter.max)) {
iter.count <- iter.count + 1
x.iter[iter.count,] = x.hat[1,1:4]

all_sol = run.subproblems(pha.sub, scen, m.scen, r, w, x.hat, r_short, r_long, b)

x.hat.old <- x.hat
x.hat <- calc.x.hat()
w.old <- w
for(i in seq(nrow(m.scen))) {
  w[i,] = w[i,] + r*(all_sol[[i]]$x[1:6]-x.hat[i,])
}

# convergence test
test.1 <- norm(x.hat-x.hat.old, type="2")
test.2 <- norm(w - w.old, type="2")

conver <- (test.1 <= tol) & (test.2 <= tol)
}
```

The progress of the algorithm is illustrated by the following graph, which shows a linear convergence towards the solution. Strategies for speeding up the convergence is a topic of active research.

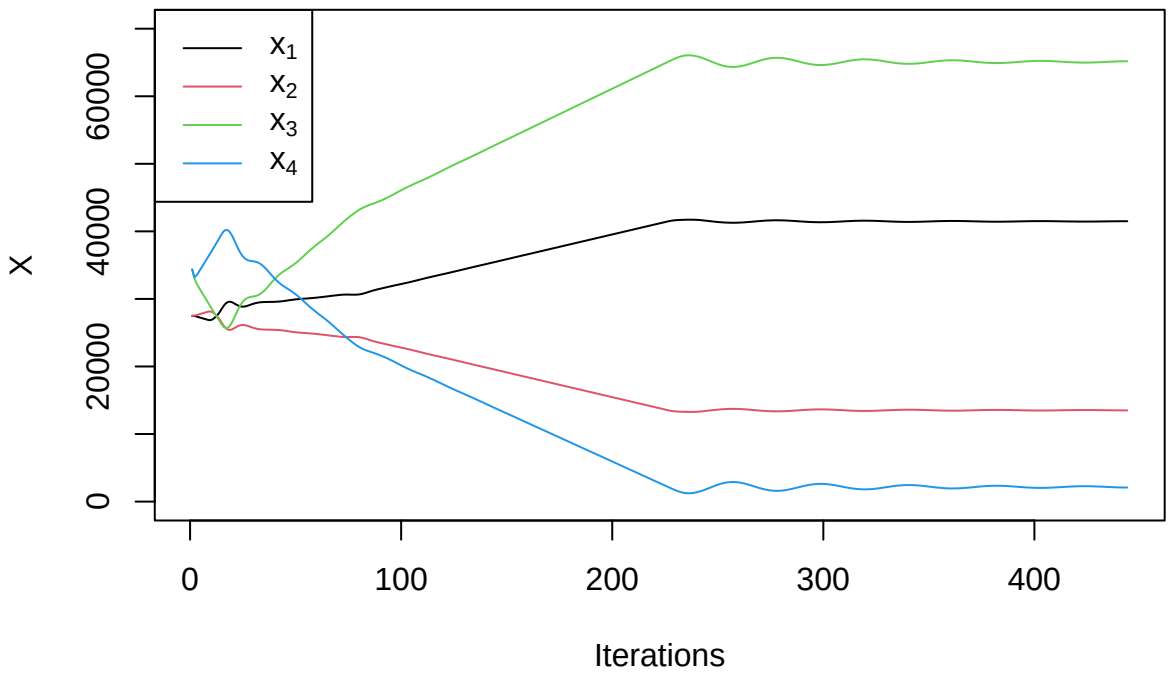


Figure 2: Progressive Hedging Iterationsi, showing the convergence of the first 4 variables of the problem