Optimisation en Finance

2014-2015

Yoann DUBERNET

Sommaire

Introduction

Durant ce projet, nous allons présenter deux applications de l'optimisation de modèles linéaires en finance. Le premier touchera au domaine de l'optimisation de la CVaR et le second au calcul d'un indexfund.

En utilisant les outils logiciels existants pour la résolution de systèmes linéaires, nous pouvons pousser plus loin que simplement l'écriture de systèmes linéaires comme vu durant le cours. En effet, ceux-ci nous en permettent la résolution et d'en déduire des informations qui nous auraient été impossibles de déduire si une résolution avait été faite à la main, compte tenu du temps que cela aurait pris.

Pour chaque application, nous allons décrire le problème, expliquer les applications de celui-ci et décrire le travail effectué pour le résoudre ainsi que les résultats clés déduits de nos simulations.

Optimisation de la CVaR

I. Introduction

La gestion du risque en finance est une des principales activités des institutions financières. En effet, toute personne raisonnée souhaite en minimiser les pertes qu'impliqueraient un investissement. Le problème est que personne ne connait les valeurs futures d'un produit dont la valeur varie de façon non déterminée par avance. C'est d'ailleurs pour cela que ces produits ont un généralement rendement plus important que les produits dis non-risqués, puisque pour ceux-ci, les rendements futurs sont déjà tous déterminés à l'avance. En conséquence, les investisseurs chercheront toujours à investir dans les produits les moins risqués afin, pas forcément de maximiser les gains, mais à minima de minimiser les pertes potentielles.

Dans ce même contexte, des accords (exemple : les accords de Bâle) ont été définis afin d'obliger de respecter des indicateurs interdisant de prendre plus qu'un certain taux de risque pour éviter quelques catastrophes qui se produisent en temps de crise.

C'est pour cette raison que plusieurs outils de mesure du risque ont été développés. On pourra citer la variance, qui mesure la dispersion d'une variable aléatoire par rapport à sa moyenne. Ainsi, plus la variance est élevée, plus le risque est important car plus le produit mesuré est volatile. Deux autres mesures très utilisées pour mesurer le risque en finance sont la VaR et la CVaR.

La VaR - Value at Risk - est une mesure définie en finance de marché pour mesurer le montant des pertes qui ne devrait être dépassé qu'avec une probabilité donnée sur un horizon de temps donné. La formule mathématique de la VaR est la suivante :

$$VaR\alpha(X) = \{\gamma | P(X \leq \gamma) \geq \alpha\}$$

où X représente la perte possible d'un investissement, et γ et α sont définis tel que la VaR représente le plus petit majorant des pertes (gamma) auxquelles on peut s'attendre avec une probabilité supérieure à alpha.

En des termes moins techniques, elle représente la pire perte attendue sur l'horizon de temps donné pour un certain niveau de confiance alpha. Généralement, cet indice de confiance est de 95 voire 99%. Notons ainsi que pour les banques, la commission de Bâle exige que la VaR soit calculée avec un niveau de confiance de 99%.

La CVaR - Conditionnal Value at Risk - est une autre mesure du risque définie suite aux imperfections de la VaR. Par exemple, la VaR ne reflète pas le fait que diversifier un portefeuille diminue le risque de celui-ci puisqu'au contraire, en ajoutant un nouveau titre dans le portefeuille, le risque augmente. De plus, elle ne possède pas la propriété de sous-additivité. *On dit qu'une fonction est sous-additive lorsque pour tous ses éléments x et y,* $f(x + y) \leq f(x) + f(y)$.

La CVaR est une mesure de risque plus pertinente et aussi puissante que la VaR puisqu'elle tient compte des catastrophes des évènements de grands dommages encourus. Autrement dit, la CVaR, puisqu'elle est aussi une mesure de perte, permet de mieux mesurer les évènements rares. Elle peut ainsi se définir comme la perte potentielle que peut subir un portefeuille avec un seuil de confiance minimal suite à des mouvements défavorables des prix du marché sachant que cette perte dépasse au moins la VaR.

$$CVaR\alpha(x) = \frac{1}{1-\alpha}\int_{f(x,y)\geq VaR\alpha(x)} f(x,y)p(y)d(y)$$

$$CVaR\alpha(X) = VaR + Moyenne\ \ des\ pertes\ dépassant\ la\ VaR$$

On comprend ici tout l'intérêt de minimiser la CVaR. Car si la CVaR (qui est toujours supérieure ou égale à la VaR compte-tenu de sa définition) est strictement égale à la VaR, cela veut dire qu'il ne peut y avoir aucun évènement avec des pertes supérieures à la VaR. On aura donc minimisé les risques de pertes extrêmes. Bien sûr, la limite de cela est le taux de confiance. Ainsi, si alpha vaut 0,90, il y a tout de même 10% de chance que cela se passe mal (pertes plus grandes que celle déterminée par la VaR). D'où l'intérêt d'avoir un taux de confiance le plus élevé possible.

Attachons-nous désormais au modèle d'allocation d'un portefeuille vérifiant les contraintes de Markovitz en minimisant la CVaR associée à ce portefeuille.

II. Modèle du problème de minimisation de la CVaR

On a défini une fonction perte $f(x,y) = (b-y)^T x$ où x est un portefeuille, b est le vecteur des prix initiaux des actifs du portefeuille et y est un vecteur représentant les prix à la fin de la période de temps considérée pour une simulation donnée. Travaillant avec un programme informatique, les expressions continues deviennent discontinues. Ainsi, pour l'expression de la CVaR, on passe d'une intégrale à une somme discrète. De plus, on suppose légitimement que chaque scénario a la même probabilité de se produire. Si S est le nombre de scénarios possibles, la probabilité de survenue de chaque scénarios étant la même, on peut factoriser la somme par la probabilité de survenue de chaque scénario, en l'occurrence $\frac{1}{S}$. A partir de cela, on en déduit la CVaR discontinue associée à notre fonction perte :

$$CVaR_\alpha(x) = \frac{1}{S(1-\alpha)} \sum_{i\ si\ f(x,y)\geq VaR\alpha(x)} f(x,y)$$

Dans l'objectif de la minimiser via la résolution d'un système linéaire, on introduit la fonction auxiliaire suivante :

$$F_\alpha(x,\gamma) = \gamma + \frac{1}{S(1-\alpha)}\sum_{s=1}^{S}(f(x,y_s) - \gamma)^+$$

où $+$ signifie que les valeurs prises par $f(x,y_s) - \gamma$ ne sont prises en compte que si elles sont positives et valent 0 sinon.

Afin de travailler avec un système linéaire, on pose $z_s = (f(x, y_s) - \gamma)^+$. Cela ajoutera deux contraintes à notre système final. De plus, on précise que l'ensemble X représente l'ensemble des portefeuilles d'actifs à rendements aléatoires.

Alors le problème de minimisation de la CVaR, qui est équivalent à la minimisation de la fonction F_α introduite, est équivalent à :

$$\begin{cases} \text{Min } \gamma + \dfrac{1}{S(1-\alpha)} \displaystyle\sum_{s=1}^{S} z_s \\ z_s \geq 0, \forall s \in [\![1; S]\!] \\ z_s \geq f(x, y_s) - \gamma, \forall s \in [\![1; S]\!] \\ x \in X \\ \gamma \in \mathbb{R} \end{cases}$$

Notre fonction de perte étant linéaire en x, notre programme est donc un programme linéaire dont la fonction objectif minimise la CVaR.

Pour terminer, notre portefeuille doit vérifier les contraintes d'un portefeuille de Markovitz. Comme la fonction objectif minimise les pertes, les contraintes de ce type de portefeuille de Markovitz sont d'avoir un rendement minimum supérieure à une valeur notée R et la classique $\sum_{i=1}^{n} x_i = 1$.

Au final, notre modèle de minimisation de la CVaR sous les contraintes d'un portefeuille de Markovitz en remplaçant la fonction perte par sa définition est :

$$\begin{cases} \text{Min } \gamma + \dfrac{1}{S(1-\alpha)} \displaystyle\sum_{s=1}^{S} z_s \\ z_s \geq 0, \forall s \in [\![1; S]\!] \\ z_s \geq (b - y_s)^T x - \gamma, \forall s \in [\![1; S]\!] \\ \displaystyle\sum_{i=1}^{n} x_i = 1 \\ \displaystyle\sum_{i=1}^{n} \mu_i x_i \geq R \\ x \in X \\ \gamma \in \mathbb{R} \end{cases}$$

III. Résolution du modèle de minimisation de la CVaR

1 Hypothèses :

Dans cette partie, nous explicitons les hypothèses faites pour l'implémentation de la résolution du modèle :

- Nous supposons que chaque scénario peut se produire selon une même probabilité $\frac{1}{S}$ où S est le nombre de scénarios.
- Nous supposons que le vecteur des prix initiaux des actifs est généré selon une loi de probabilités uniforme. Cela modélise le fait qu'il est tout aussi probable que le prix d'un actif soit de 1 € que de 100 €.
- Nous supposons que le vecteur des prix futurs selon les scénarios futurs est généré selon une loi de probabilité normale. Cela modélise le fait qu'il est peu probable d'avoir un actif dont le prix initial est de 50 € et que son prix à l'issue de la période de temps considérée soit de 1 € ou, inversement, de 100 €. Au contraire, il est fortement probable que son prix à l'issue de la période de temps considérée soit proche de son prix initial de 50€.
- Nous supposons que l'allocation totale du portefeuille est constante. Cela signifie que l'argent du portefeuille est toujours entièrement investie dans les actifs le composant. C'est une hypothèse réaliste puisqu'un investisseur cherchera toujours à investir le maximum d'argent disponible, l'argent dormant ne rapportant rien.
- Nous supposons que les prix de nos actifs varient entre 0 et 100 €.
- Nous supposons que les rendements de chaque actif sont générés aléatoirement selon une loi uniforme pour la même raison que celle régissant la simulation des prix initiaux des actifs.
- Nous supposons que le rendement moyen d'un actif au terme de la période considérée est de 15%. Il s'agit d'un rendement assez proche de ce que l'on peut trouver sur des actifs risqués sur les marchés financiers.
- Nous supposons que les actifs répartis dans le portefeuille peuvent l'être à l'achat comme en vente à découvert :

$$\forall i \in [\![1; n]\!], x_i \in [-1; 1]$$

  En particulier, il est donc possible de profiter d'opportunités d'arbitrage de type A.
- Nous nous restreignons à des scénarios générant uniquement de la perte afin de focaliser notre analyse sur la minimisation de celles-ci. Cela implique donc que notre CVaR sera toujours positive. En conséquence, notre vecteur aléatoire des prix finaux est déterminé de telle sorte la CVaR soit toujours positive. Etant généré selon la loi normale $\mathcal{N}(0,1)$, nous appliquons la formule suivante :

$$\forall i, j \in [\![1, n]\!] * [\![1, S]\!], y[i][j] = \min\left(b[i], b[i] * (\mathcal{N}(0,1) + 0{,}5)\right)$$

2 Implémentation de la résolution

Afin de résoudre ce modèle linéaire, j'ai souhaité utiliser le puissant solveur de systèmes linéaires GLPK. Principalement pour deux raisons :

- Il est libre de droits, étant sous licence GNU v3 et n'est donc pas bridé contrairement à Cplex qui, en version gratuite, n'autorise pas la résolution de scénarios s'il y a plus de 1000 contraintes.
- Il m'a permit d'apprendre à utiliser une nouvelle librairie, ce qui, pour un développeur se destinant au monde de la finance, ne peut être qu'un atout supplémentaire.

Je relèverais tout de même de grosses difficultés à le prendre en mains. En effet, j'ai bien compris qu'il s'agissait initialement d'une librairie développée en C et ai mi pas moins d'une après-midi entière à implémenter mon premier algorithme de résolution. Mis à part ce défaut, cette librairie très puissante m'a donné entière satisfaction pour la résolution.

Connaissant les contraintes du système sur les z_s, j'ai ramené ces contraintes à la forme suivante :

$$0 \geq (b_i - y_{ij})x_i - \gamma - z_j$$

où $i, j \in [\![1; n]\!] * [\![1; S]\!]$.
Au final, en prenant en compte les deux contraintes sur le portefeuille de Markovitz ainsi que la contrainte sur l'allocation constante, j'obtiens une matrice de la forme suivante :

	γ	z1		…	zs	x1	x2	…	xn
objectif	γ	$\frac{1}{S}(1-\alpha)$			$\frac{1}{S}(1-\alpha)$	0	0		0
z1	-γ	-1	0		0	$(b_1 - y_{11})$	$(b_2 - y_{21})$		$(b_n - y_{n1})$
z2	-γ	0	-1			$(b_1 - y_{12})$	$(b_2 - y_{22})$		$(b_n - y_{n2})$
zs	-γ	0			-1	$(b_1 - y_{1S})$	$(b_2 - y_{2S})$		$(b_n - y_{nS})$
$\sum x_i$ = 1	0	0			0	1	1		1
μ*x >= R	0	0			0	μ_1	μ_2		μ_n
allocation constante	0	0			0	b_1	b_2		b_n

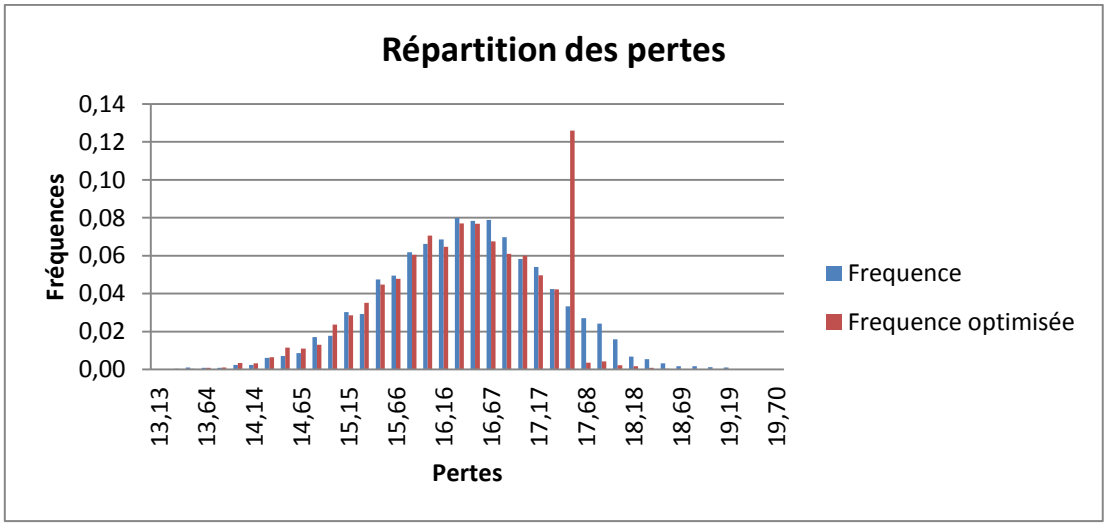
A partir de cette matrice et des contraintes sur les valeurs que peuvent prendre celles-ci, le programme résout alors le système linéaire et me donne un résultat avec la CVaR optimisée.

Les contraintes sur les valeurs (qui résultent en partie de nos hypothèses) sont les suivantes :

$$\begin{cases} Var \geq 0 \text{ (découle de se restreindre à des scénarios de perte)} \\ \forall s \in [\![1; S]\!], z_s \geq 0 \\ \forall i \in [\![1; n]\!], x_i \in [-1; 1] \\ \sum_{i=1}^{n} x_i = 1 \\ \sum_{i=1}^{n} \mu_i x_i \geq R \\ \sum_{i=1}^{n} b_i x_i = cte \end{cases}$$

3 Exemple de résolutions et explication de celles-ci

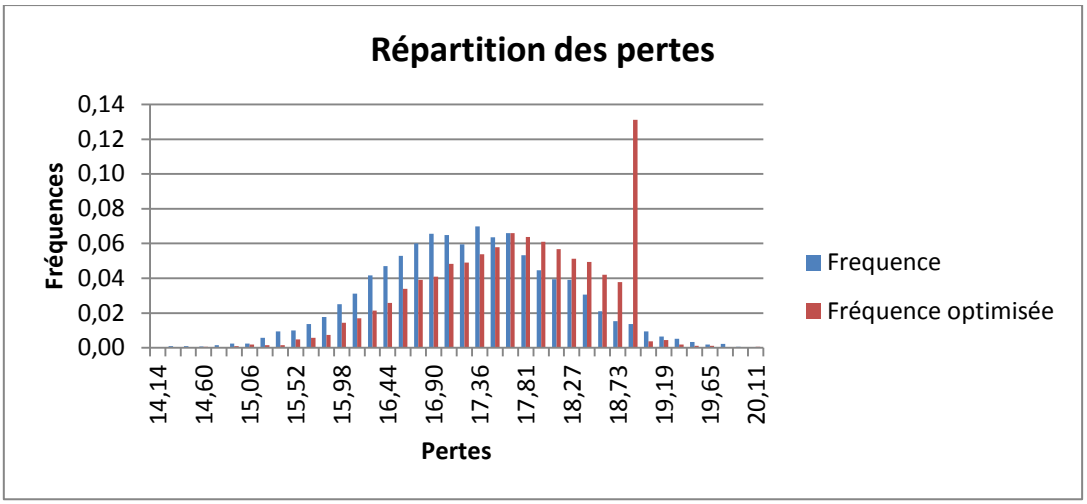
Dans mon programme ont été rajoutées des consignes permettant d'exporter les données générées par celui-ci au format .csv afin de les traiter ensuite sur un tableur type Microsoft Excel. J'y exporte les prix initiaux, les rentabilités de chaque actif, la répartition des actifs dans le portefeuille optimisé, la VaR optimisée ainsi que la CVaR optimisée. A partir de ces données, j'en déduis la perte moyenne du portefeuille pour chaque scénario, la VaR et la CVaR non optimisées, je vérifie que l'allocation totale du portefeuille avec les prix initiaux est bien la même avant et après optimisation. Et finalement, avec un tableau de fréquence des pertes, j'en déduit leur répartition et obtiens ce type de graphes :



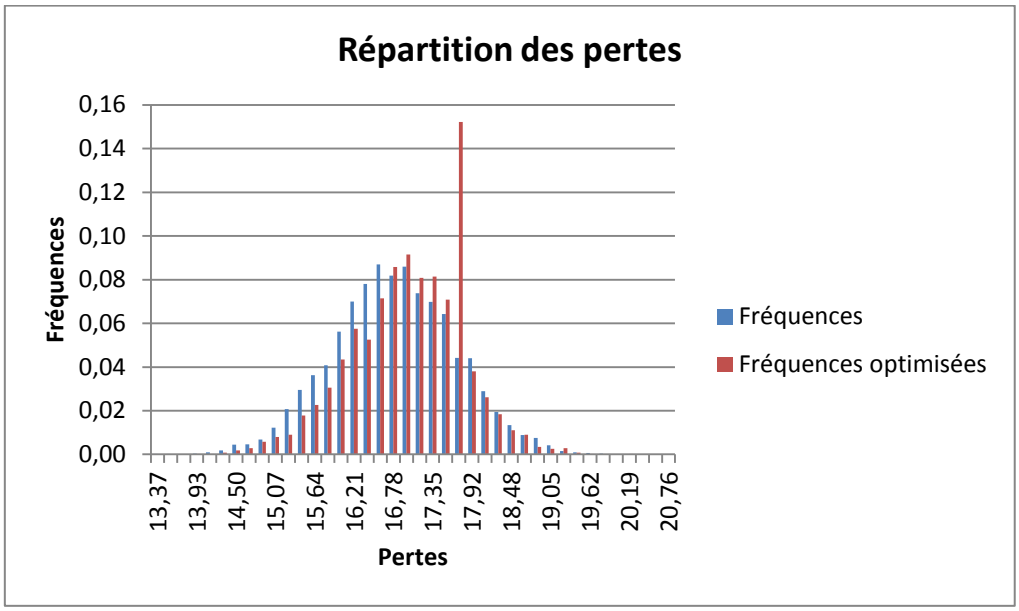
De ce travail résulte ensuite une analyse : la minimisation de la CVaR résulte aussi en la minimisation de la VaR puisque $CVaR \geq VaR$ par propriété. Sur ce scénario en particulier, on remarque que la répartition des pertes sont similaires en tous points (et respectent une gaussienne due à la loi normale de y) avant un montant de perte bien particulier où la fréquence optimisée de perte y est très importante et après laquelle les fréquences de pertes dans le portefeuille optimisé sont inférieures (voire quasi nulles) à celles possibles dans le portefeuille non optimisé. L'optimisation a en effet consisté à minimiser la CVaR et donc les pertes supérieures à la VaR. Elle ne cherche donc pas à minimiser toutes les pertes, mais à minimiser les scénarios de pertes très importantes, en particulier supérieures à la VaR, la CVaR ne prenant en compte que les pertes supérieures à la VaR. Cette fonction de perte est d'autant plus utile que légalement, la VaR représente le montant obligatoirement provisionné par les banques en cas d'un scénario de pertes très importantes comme des crises sur les marchés financiers. Minimiser les pertes supérieures à la VaR minimise donc les cas où l'argent mis de côté pour encaisser les pertes ne serait pas suffisante et où l'institution financière pourrait donc se retrouver dans une situation délicate. Les pertes conséquentes se retrouvent donc optimisées au même niveau de perte situé entre la VaR optimisée et la CVaR optimisée, d'où un pic de fréquences de pertes possibles à cet endroit puis beaucoup moins de pertes plus conséquentes.

Par analogie, pour un agent, il est plus confortable de savoir qu'il a de grandes chances de perdre 50 € et quasiment aucune de perdre plus que 50 € auquel cas il provisionnera donc 50 €, que d'avoir plus de chance de perdre 50 € que 60 € mais d'avoir tout de même des probabilités non négligeables (même si peu élevées) de perdre 60, voire 70 €.

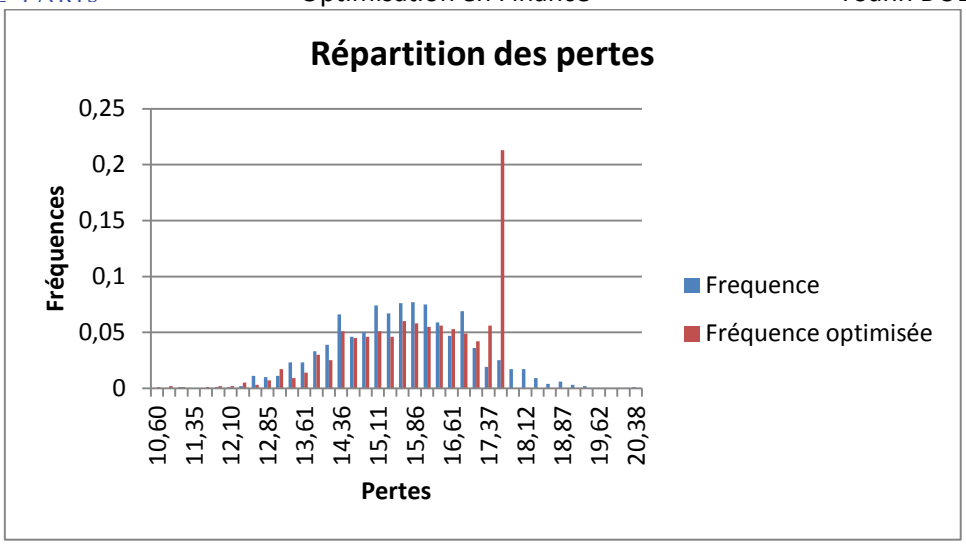
J'ai répété cette procédure pour créer des graphes de fréquences de pertes avec des entrées différentes, notamment sur notre degré de confiance alpha. En voici ci-dessous :



Celui-ci a les mêmes paramètres d'entrée que le précédent et nous montre à quel point les données (générées aléatoirement) sont tout de même importantes. En effet, ce graphe nous apprend qu'il est aussi possible, pour minimiser les pertes maximales, d'avoir globalement plus de probabilités d'avoir des pertes importantes et moins de pertes peu importantes.



Celui-ci a été généré à partir d'un degré de confiance de 80 %. On y repère ainsi que les pertes maximales n'ont pas été tant minimisées que cela bien qu'il y ait un pic de pertes comme sur tous les autres graphes. De ceci on déduit l'importance d'avoir un degré de confiance le plus proche possible de 1.



Enfin, celui-ci montre justement un calcul effectué avec un degré de confiance de 99%, bien que le portefeuille considéré ait moins d'actifs et de scénarios.

On voit bien qu'avec un degré de confiance très élevé, on peut aller jusqu'à avoir

$$VaR_{optimisée} = CVaR_{optimisée}$$

ce qui annule toutes les possibilités de pertes importantes supérieures à la VaR. Autant dire une situation idéale pour un agent financier.

III. Analyse statistique de la minimisation de la CVaR

1 Démarche

Ma démarche a consisté à analyser la variation de la VaR, la CVaR et la durée d'exécution en fonction des paramètres d'entrée du programme de résolution que sont :

- le nombre d'actifs composant le portefeuille
- le nombre de scénarios considérés pendant la simulation
- le rendement minimum espéré associé à la contrainte du portefeuille de Markovitz
- le degré de confiance dans le calcul de la CVaR optimisée

Pendant que je faisais varier un paramètre, je bloquais les autres à des valeurs prédéfinies par défaut. Ces valeurs sont :

- nombre d'actifs : 200
- nombre de scénarios : 2000
- rendement minimum espéré : 10 %
- degré de confiance minimum : 95 %

Puis pour une valeur de paramètre particulière, je lançais au moins 50 fois le programme afin d'avoir des échantillons de données suffisamment représentatifs.

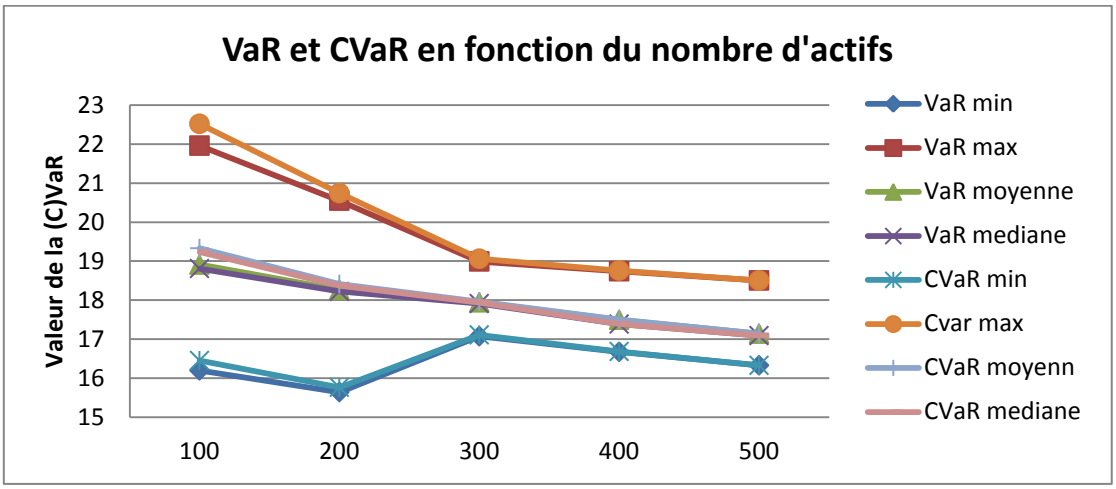
Enfin, à partir des données brutes, j'en ai extrait le minimum, le maximum, la valeur moyenne et la valeur médiane.

Remarque : On remarquera dans la section suivante que pour la moyenne et la médiane sont toujours très proches l'une de l'autre. Cela montre que les échantillons choisis (50 tests à chaque fois) sont suffisamment conséquents pour ne pas se retrouver avec des effets de seuil comme on peut en avoir sur de petits échantillons.

Nota : Ces simulations ont été réalisés sur un processeur Intel(R) Core(TM) i5-3210M CPU cadencé à 2,50 GHz avec 6,00 Go de mémoire vive sans autre processus tournant au premier plan.

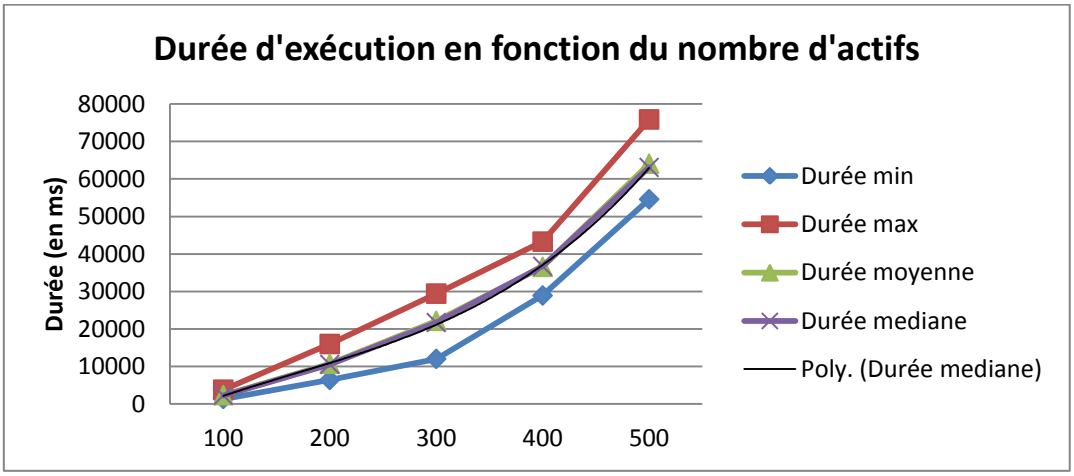
2 Analyse

Analyse en fonction du nombre d'actifs



De la variation du nombre d'actifs (tout autre paramètre restant constant), l'analyse de la VaR et de la CVaR montre deux choses :

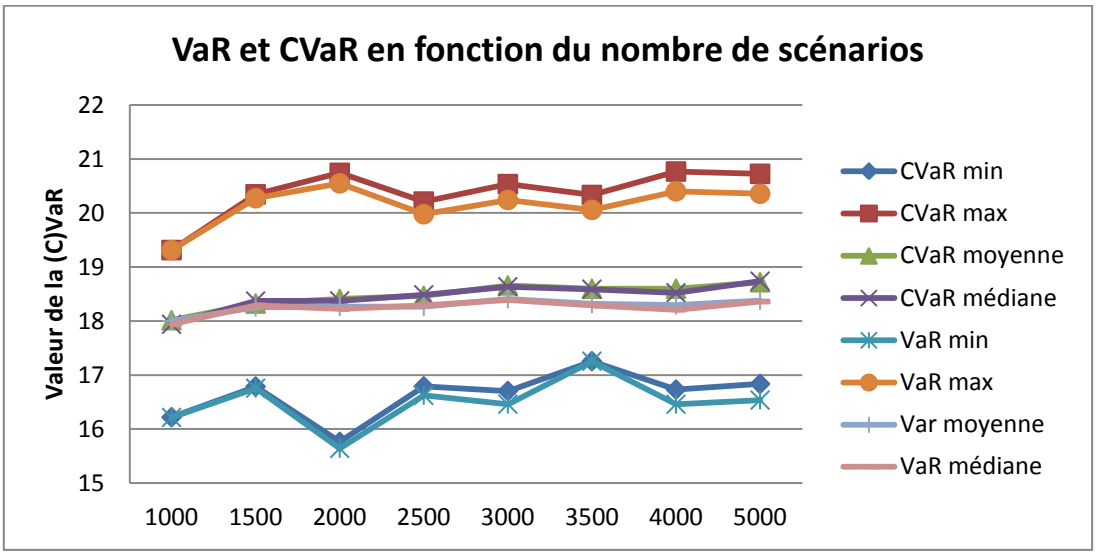
- Plus l'on ajoute d'actifs aux portefeuille, plus la VaR et la CVaR diminuent. Cela vient justifier l'idée que plus l'on diversifie son portefeuille, plus l'on diminue le risque de celui-ci.
- Plus l'on ajoute d'actifs au portefeuille, plus la VaR et la CVaR sont proches l'une de l'autre. De même, cela vient justifier l'idée que plus l'on diversifie son portefeuille, plus l'on diminue le risque de celui-ci car moins les pires scénarios ont une probabilité de se produire.



De la variation du nombre d'actifs, l'analyse de la durée d'exécution nous montre que la durée de calcul est polynomiale (une régression nous indiquant une degré d'ordre 3), variant en moyenne de 2,3 secondes pour un portefeuille avec 100 actifs à une minute pour un portefeuille avec 500 actifs.

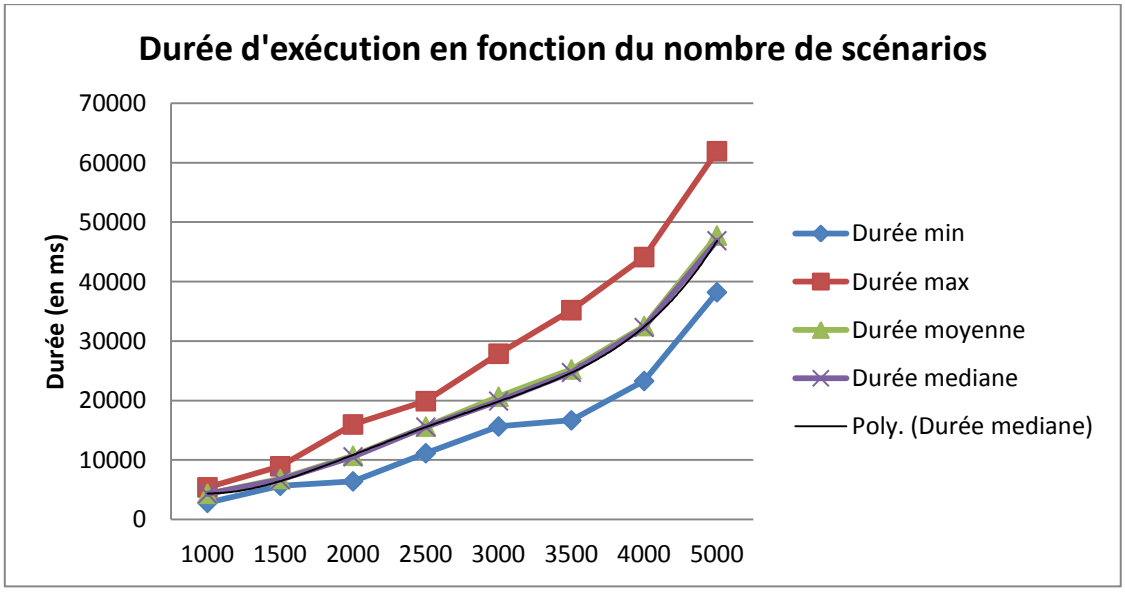
Finalement, si l'on veut minimiser la volatilité associée à un portefeuille, il faut en augmenter le nombre d'actifs mais les calculs de pertes en entreprise, normalement préalables à toute décision du trader, seront plus long et pourraient risquer de faire perdre une opportunité.

Analyse en fonction du nombre de scénarios



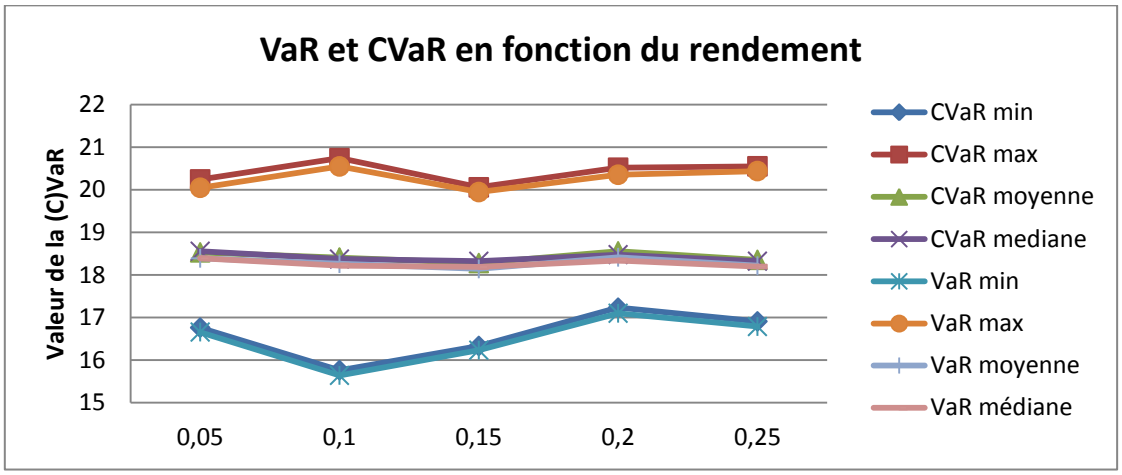
De ce graphe nous déduisons deux choses :

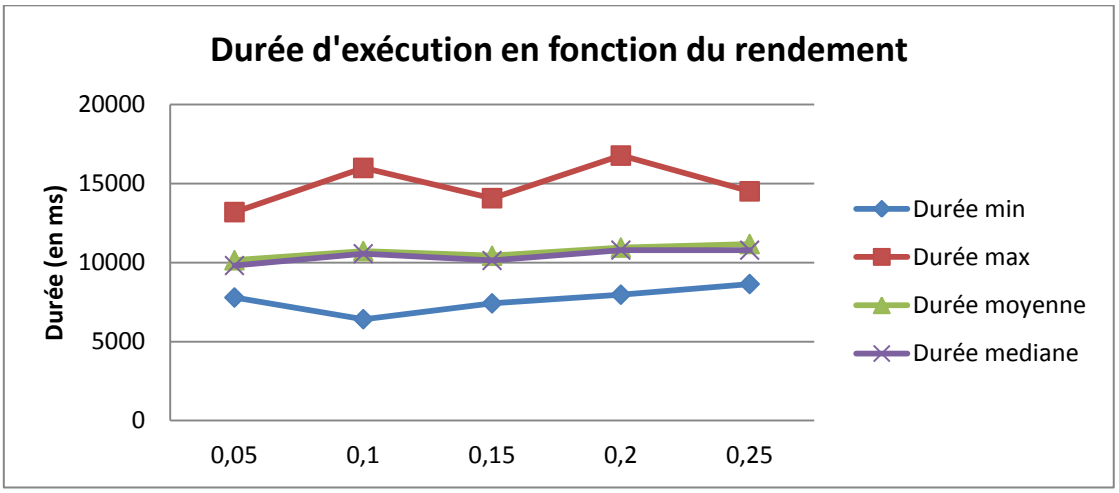
- La VaR et la CVaR ne dépendent pas du nombre de scénarios
- Plus le nombre de scénarios est élevé, plus l'écart entre les deux est important. On comprend aisément la raison : plus l'on introduit de scénarios, plus la probabilité que des scénarios extrêmes se produisent est importante et donc plus la CVaR va être élevée par rapport à la VaR.



On déduit de ce graphe que la durée d'exécution est polynomiale en fonction du nombre de scénarios. Une régression nous permet d'apprécier une durée d'exécution de l'ordre $\theta(S^4)$. Ainsi, il n'est peut-être pas pertinent de faire trop de simulations (2000 semble un bon compromis) puisque le temps de calcul augmente beaucoup par rapport à une garantie qui pourrait nous être apportée quant à la diminution de la probabilité de fortes pertes sur le portefeuille.

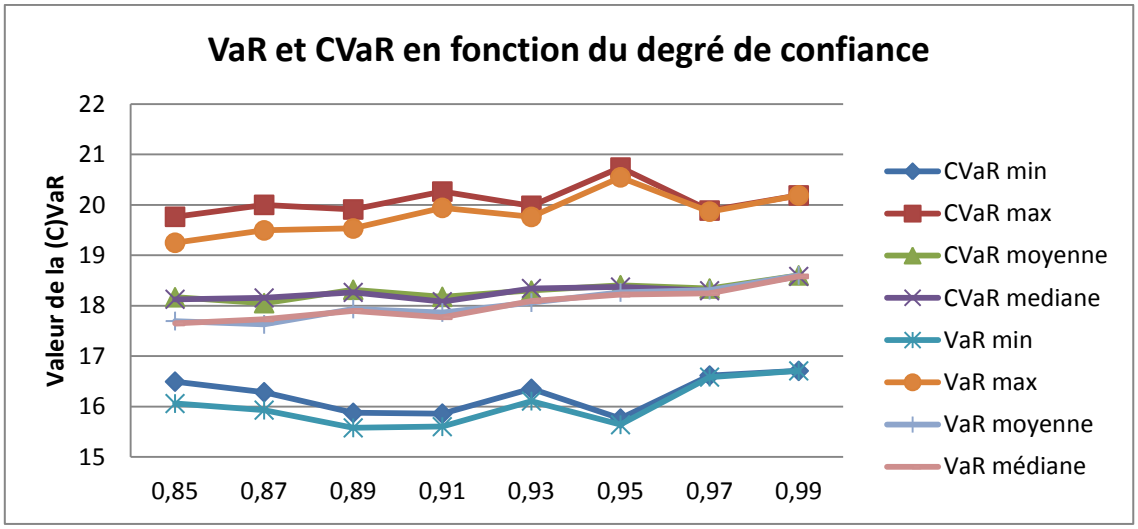
Analyse en fonction du nombre du rendement

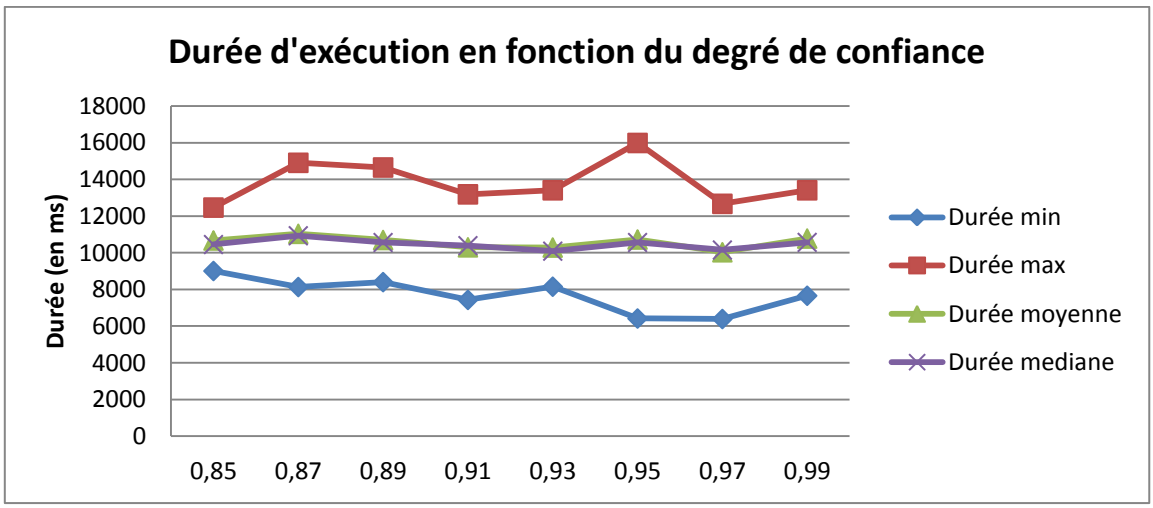




De ces deux graphes, nous déduisons que l'exécution ne dépend aucunement du rendement minimum attendu pour le portefeuille de Markovitz, que ce soit par rapport à la VaR/CVaR, ou par rapport à la durée d'exécution. Cela vient conforter le fait que notre portefeuille est construit indépendamment du rendement minimal exigé, contrairement au modèle que l'on simulera dans le prochain chapitre.

Analyse en fonction du nombre du degré de confiance





De ces deux graphes nous déduisons trois choses :

- La durée d'exécution ne dépend aucunement du degré de confiance
- La valeur de VaR et de la CVaR ne dépendent aucunement du degré de confiance
- Plus le degré de confiance est important, plus la VaR et la CVaR se rapprochent en valeur et donc moins les scénarios de pertes importantes se produiront.

Il est donc de tout intérêt d'avoir un degré de confiance maximal.

V. Résolution du modèle de Markovitz sous contrainte de CVaR

Le modèle à considérer dans cette partie est le suivant :

$$\begin{cases} \text{Max} \sum_{i=1}^{n} \mu_i x_i \\ s.c. \\ z_s \geq 0, \forall s \in [\![1; S]\!] \\ z_s \geq (b - y_s)^T x - \gamma, \forall s \in [\![1; S]\!] \\ \sum_{i=1}^{n} x_i = 1 \\ \gamma + \frac{1}{S(1-\alpha)} \sum_{s=1}^{S} z_s \leq CVaRMax \\ x \in X \\ \gamma \in \mathbb{R} \end{cases}$$

En quelques mots, ce modèle consiste à maximiser le rendement conformément au modèle de Markovitz tout en ayant une contrainte sur le risque maximum pris par les acteurs du marché.

Par rapport au modèle précédent, peu de choses changent dans l'implémentation. Nous les listons ici :
- La fonction objectif ne consiste plus à minimiser la CVaR mais à maximiser le rendement
- La contrainte obligeant un rendement supérieur à R disparaît et est remplacée par une contrainte obligeant une CVaR inférieure à une $CVaRMax$ définie au début de l'exécution du programme.

VI. Analyse statistique de la maximisation du rendement

1 Démarche

Ma démarche a consisté à analyser la variation de la VaR, le rendement et la durée d'exécution en fonction des paramètres d'entrée du programme de résolution que sont :

- le nombre d'actifs composant le portefeuille
- le nombre de scénarios considérés pendant la simulation
- la CVaR maximale espérée
- le degré de confiance dans l'optimisation du rendement du portefeuille.

Pendant que je faisais varier un paramètre, je bloquais les autres à des valeurs prédéfinies par défaut. Ces valeurs sont :

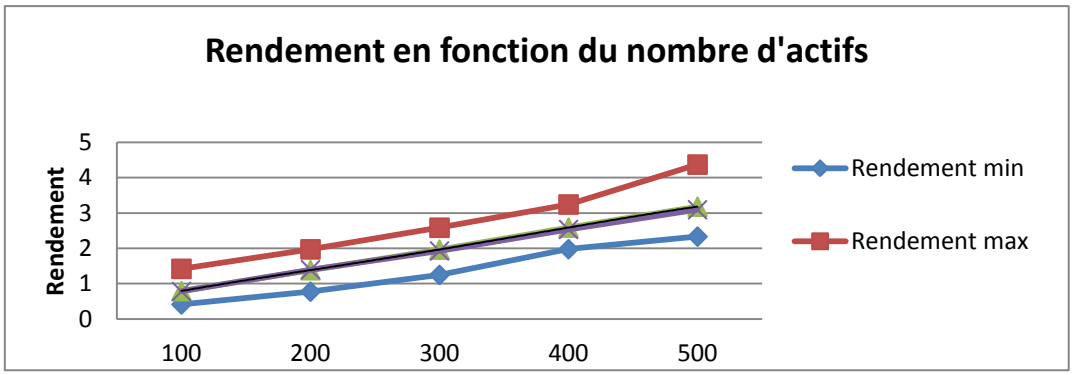
- nombre d'actifs : 200
- nombre de scénarios : 2000
- CVaR maximale espérée : 25
- degré de confiance minimum : 95 %

Remarque : Les procédures de simulations et de tests massifs sont les mêmes que dans la simulation de la minimisation de la CVaR sous contrainte de rendement.

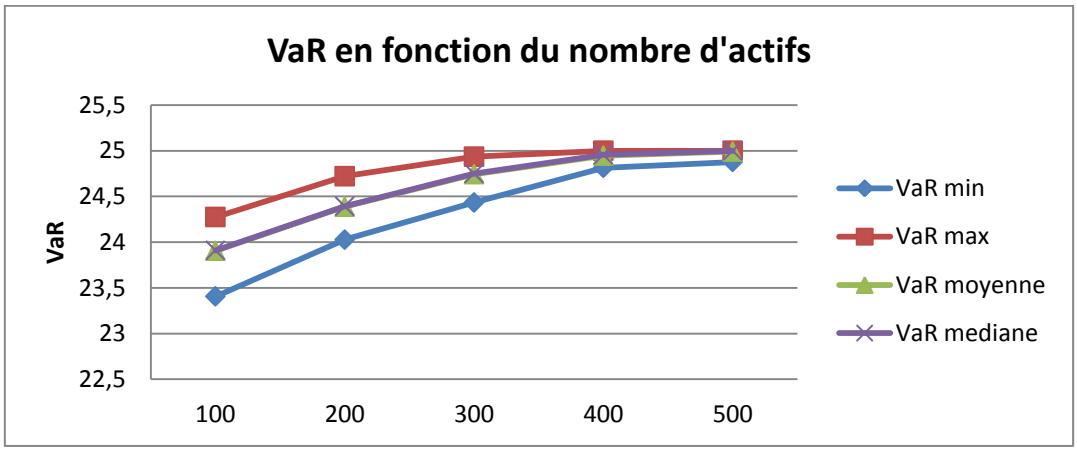
Dans les analyses suivantes, ce qui change par rapport à précédemment est que désormais, le rendement obtenu est variable et que dès lors qu'une solution optimale est trouvée, la CVaR est toujours égale à $CVaRMax$. Enfin, généralement, la VaR optimisée trouvée par GLPK est souvent très proche en valeur de $CVaRMax$.

2 Analyse

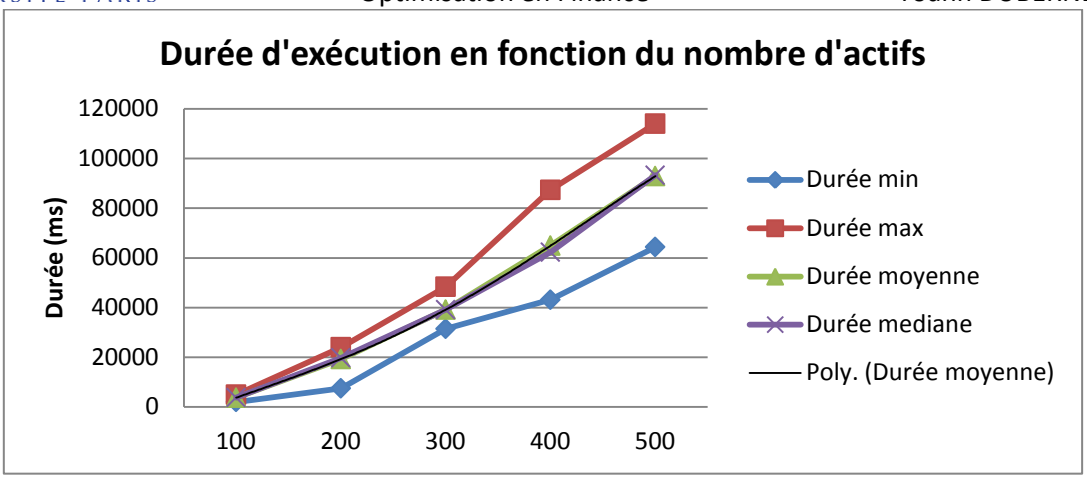
Analyse en fonction du nombre d'actifs



On observe dans ce graphique que plus l'on ajoute d'actifs au portefeuille sous contrainte de CVaR maximale, plus le rendement sera conséquent. On remarque aussi des rendements largement supérieurs à 100 %. Déjà pour 100 actifs, le rendement moyen est de 80 % ! Des rendements aussi conséquents sont rendus possibles par la possibilité d'utiliser les opportunités d'arbitrage. GLPK les utilise massivement durant toutes les résolutions de ce modèle de maximisation du rendement.

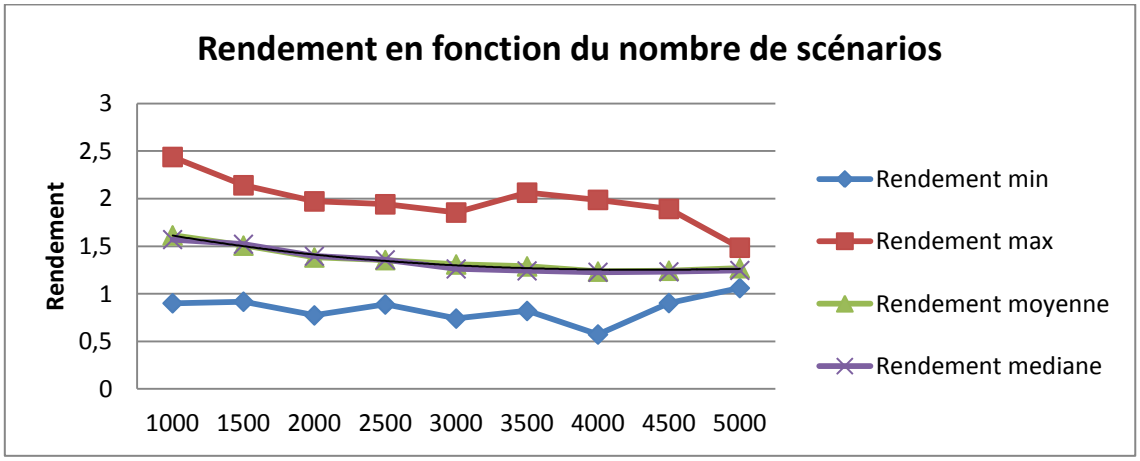


Dans ce graphique, on observe que plus l'on ajoute d'actifs, plus la VaR vient se plaquer sur la CVaRMax valant 25. Ce constat vient confirmer que la VaR n'est pas une mesure du risque parfaite puisque plus l'on diversifie son portefeuille, moins le risque pris est important, ce qui est l'inverse avec la VaR. Celle-ci vient se plaquer en dessous de la CVaR qui, elle, vaut toujours $CVaRMax$.

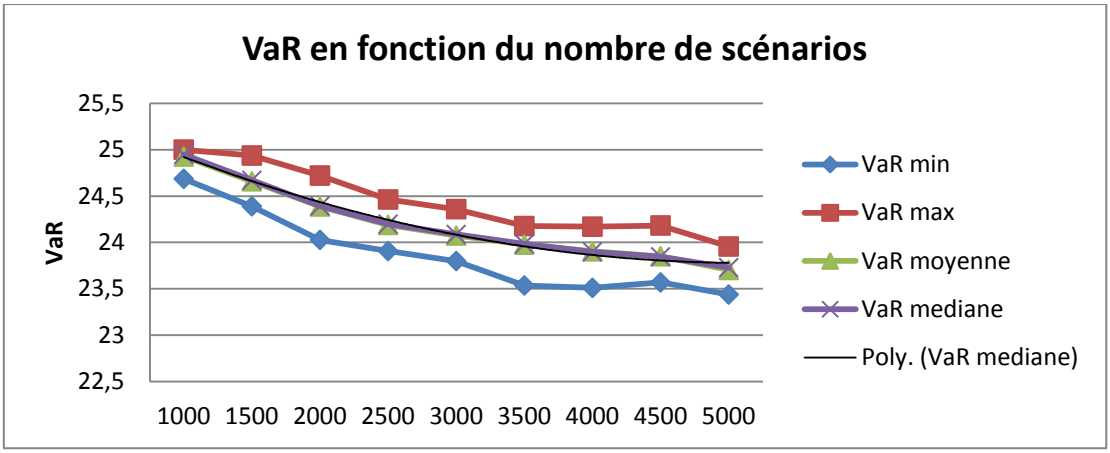


On observe par ce graphique que plus l'on a d'actifs dans notre portefeuille, plus la durée de traitement est longue. Cela paraît logique compte tenu de la taille de la matrice en mémoire en taille $O((n+s)*s)$. Cette explication est d'ailleurs valable aussi pour le problème précédent : la durée de résolution du programme dépend principalement de la taille de la matrice en entrée. On observe bien une durée de résolution linéaire en fonction du nombre d'actifs.

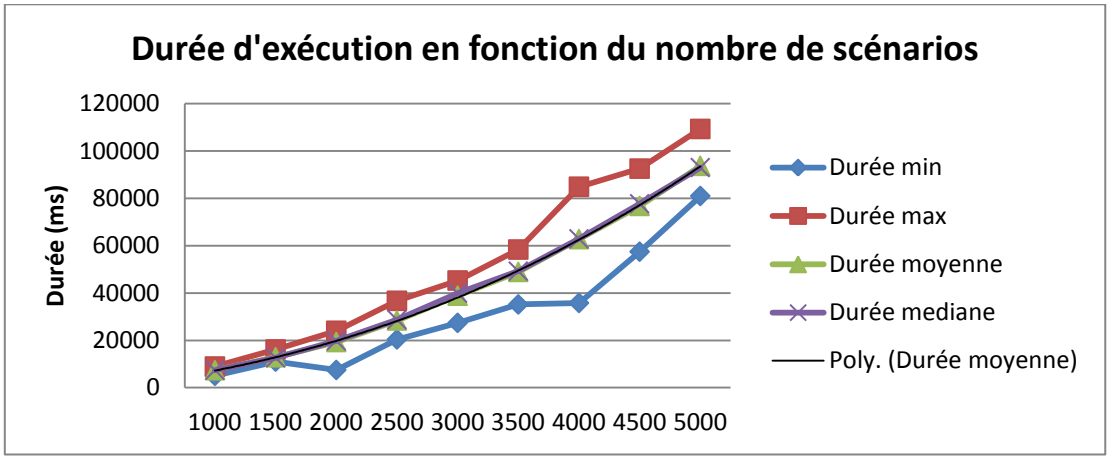
Analyse en fonction du nombre de scénarios



On note une sensible diminution du rendement lorsque l'on augmente le nombre de simulations. Cela s'explique car plus l'on simule de scénarios, plus la possibilité de scénarios contenant de aboutissant à de fortes pertes est importante.

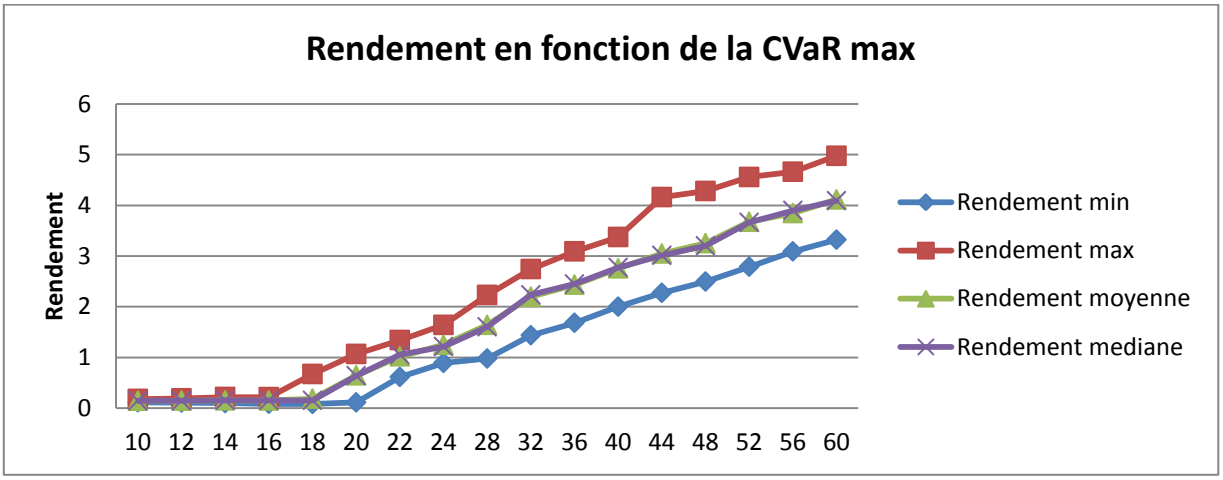


Contrairement au modèle précédent où la VaR était indépendante du nombre de scénarios, ici, nous cherchons à optimiser le rendement. Ainsi, comme le rendement a tendance à diminuer pour un nombre d'actifs fixés en fonction du nombre de scénarios, il est clair que moins l'on est rémunérés, moins l'on aura pris de risques, les marchés rémunérant les risques pris.



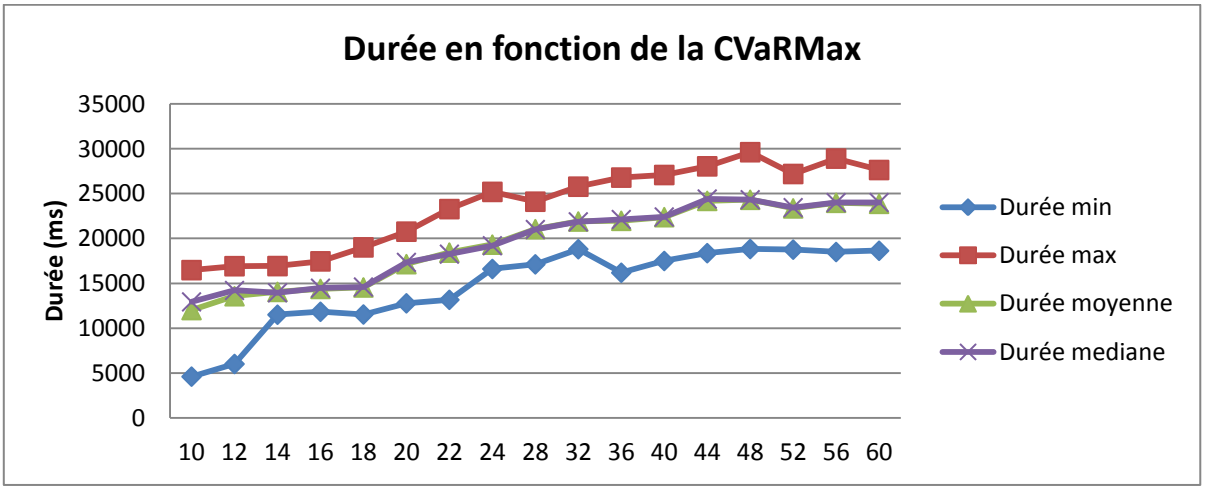
Enfin, dans ce graphique, on observe à nouveau un temps de traitement dépendant du nombre de scénarios. Dans le cadre d'une recherche à maximiser le rendement tout en prenant le moins de risques possibles, on augmentera le nombre de scénarios. Si l'on se refuse à perdre d'éventuelles opportunités sur les marchés (certains automates de trading doivent prendre des décisions sur les marchés sur des délais aux environs de 20 microsecondes), on fera beaucoup moins de simulations et l'on prendra donc plus de risque.

Analyse en fonction du maximum de CVaR acceptable



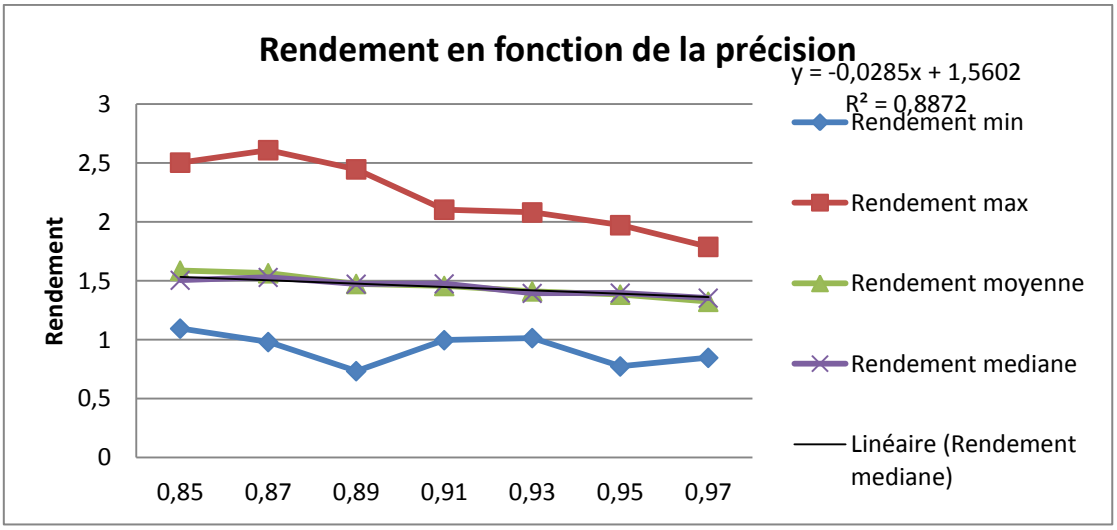
Ce graphe montre une certaine singularité pour des $CVaRMax$ entre 10 et 18 : par rapport aux autres données fixées du problème, tant que notre $CVaRMax$ n'a pas atteint une certaine valeur (18), on a des rendements très bas. En fait, notre problème n'a pas de solution optimale le résolvant. GLPK cherche alors une autre solution qui satisfasse le maximum de contraintes. Mais dans un tel cas, il ne procède pas à des arbitrages, ce qui donne des rendements en moyenne à 20 % au lieu de rendements conséquents

pouvant aller jusqu'à 400% pour un portefeuille dont nous avons fixé le risque maximum à une $CVaRMax$ de 60. Cela confirme l'idée selon laquelle plus l'on s'autorise à prendre de risque, plus le rendement engendré sera important.

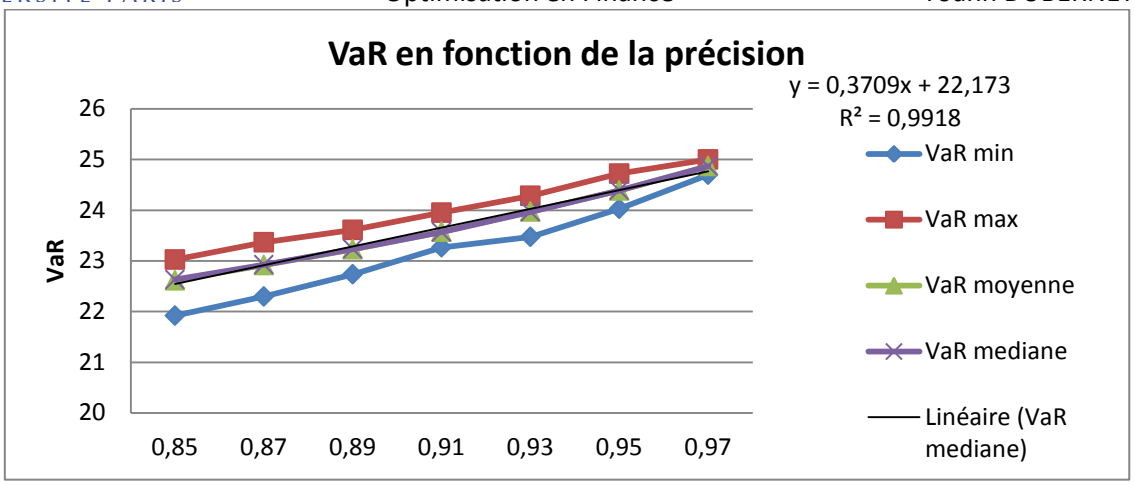


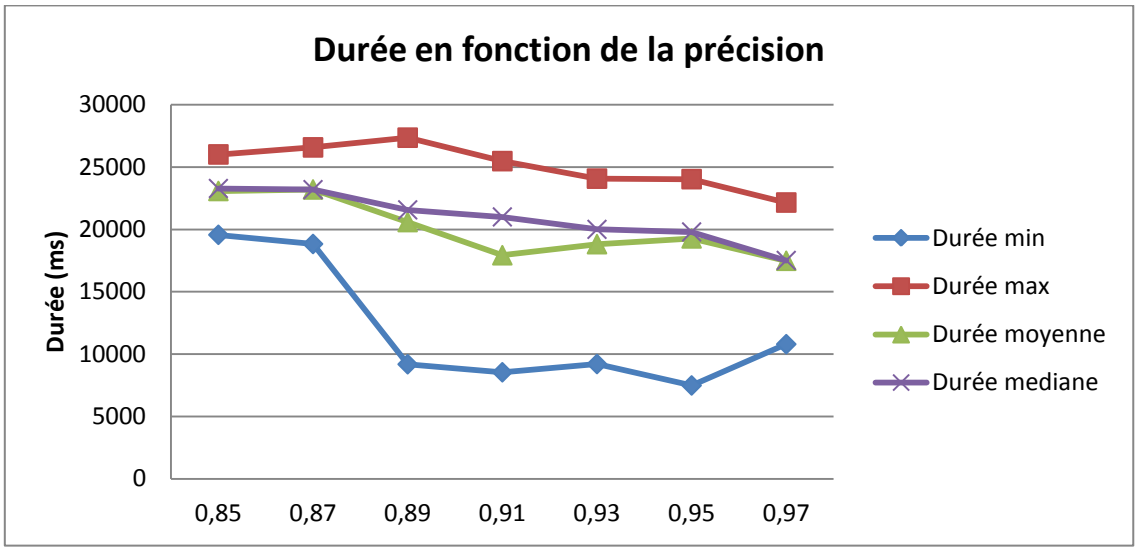
La variation de la durée d'exécution du programme a un comportement chaotique par rapport aux autres graphiques de durées, mais globalement, on remarque une hausse de la durée de traitement en fonction de la $CVaRMax$.

Analyse en fonction de la précision



On constate une sensible diminution du rendement lorsque l'on augmente la précision attendue. On explique ceci par le fait qu'une précision plus importante va naturellement augmenter la VaR optimisée (ce que l'on voit d'ailleurs dans le graphe suivant) car en augmentant la précision, on souhaite minimiser les cas très risqués. En revenant à l'idée que les marchés rémunèrent les risques, on comprend que plus l'on va chercher à minimiser le nombre de scénarios risqués, plus notre rendement que l'on peut espérer diminuera.





La variation de la durée d'exécution du programme en fonction du degré de précision souhaité indique une diminution de la durée si l'on souhaite une précision plus importante.

Optimisation d'un IndexFund

I. Introduction

Un indexfund est un portefeuille particulier composés de plusieurs actifs devant représenter le plus fidèlement possible celui-ci. Un exemple d'indexfund connu en France est le CAC-40 : il est composé des 40 entreprises réputées représenter au mieux l'évolution des cours des actions des entreprises françaises cotées sur le marché boursier européen Euronext.

Un indexfund est représenté par deux nombres essentiels qui nous suivront dans toute la suite :

- n qui représente le nombre d'actifs devant être représentés par celui-ci
- q qui représente le nombre d'actifs sélectionnés par l'algorithme pour représenter au mieux les n actifs devant être représentés

Remarque : il est possible de faire des indexfund d'indexfund, c'est d'ailleurs ce que l'on fera durant notre résolution en représentant le CAC 40 par un indexfund composé d'un nombre d'actifs strictement plus petit que les 40 initiaux.

Le problème de l'indexfund consiste finalement à choisir les actifs représentant le mieux le marché. Pour cela, sur une période considérée, on utilise la matrice des corrélations entre les n actifs de notre portefeuille. Par définition, la corrélation, un nombre entre -1 et 1, représente à quel point deux actifs considérés sont liés entre eux. Plus la corrélation est proche de 0, plus les deux actifs considérés ont peu de liens entre eux.

Le problème de base d'optimisation de l'indexfund consiste donc à maximiser la corrélation globale de notre indexfund en choisissant q actifs parmi les n disponibles. Il s'agit notamment d'un problème en nombres entiers : pour chaque actif considéré, il n'y a que deux solutions possibles :

- 0 : on ne sélectionne pas l'actif
- 1: on sélectionne l'actif pour le rajouter à notre indexfund

En raison de sa caractéristique de problème d'optimisation linéaire en nombres entiers, ce problème est dit NP-complet. Mathématiquement, cela signifie que bien qu'on puisse *vérifier* rapidement toute solution proposée du problème NP-complet, on ne sait pas en *trouver* efficacement. Or, notre objectif est pourtant de trouver une solution, et le plus vite possible.

Le problème d'optimisation de l'indexfund posé est donc le suivant :

$$\begin{cases} Max \sum_{i=1}^{n}\sum_{j=1}^{n} \rho_{ij}\, x_{ij} \\ \sum_{i=1}^{n} y_j = q \\ \sum_{j=1}^{n} x_{ij} = 1, \forall i \in [\![1;n]\!] \\ x_{ij} \leq y_j \ \forall i,j \in [\![1;n]\!]^2 \\ x_{ij}, y_j \in \{0;1\} \ \forall i,j \in [\![1;n]\!]^2 \end{cases}$$

où :

- ρ_{ij} représente la corrélation entre l'actif i et l'actif j
- x_{ij} vaut 1 si l'actif j est choisi pour représenter l'actif i, 0 sinon
- y_j représente le jème actif sélectionné pour l'indexfund

Le problème de ce problème est qu'il prend beaucoup de place en mémoire et met un temps conséquent à être résolu. Dans GLPK, la matrice qui est envoyée au solveur est une matrice de taille n² + n colonnes * n² + n + 1 lignes. Comme on l'avait vu dans l'exercice 1, le temps de résolution dans GLPK est fortement corrélé avec la taille de la matrice en entrée. Et comme précisé précédemment, nous avons affaire à un problème de résolution en nombres entiers avec une solution difficile à trouver. En effet, pour déjà 30 actifs, l'arbre de résolution de l'algorithme est déjà très conséquent. Afin de parer à ce problème, il nous est donc proposé de résoudre un algorithme équivalent par la méthode de la relaxation lagrangienne dont la mémoire utilisée par le processus n'est pas aussi conséquente et la résolution beaucoup plus rapide.

Dans la suite, on pose $u = (u1, \dots, un)$ un vecteur de n nombres, puis $\forall j \in [\![1;n]\!], C_j = \sum_{i=1}^{n} (\rho_{ij} - u_i)^+$

Alors on montre (ce n'est pas demandé dans cette résolution) que notre premier problème est équivalent au programme suivant :

$$\begin{cases} Max \sum_{j=1}^{n} C_j y_j + \sum_{i=1}^{n} u_i \\ \sum_{i=1}^{n} y_j = q \\ y_j \in \{0;1\} \ \forall j \in [\![1;n]\!] \end{cases}$$

Deux différences importantes avec le modèle précédent non relaxé :

- Le problème relaxé n'a plus qu'une seule contrainte
- Le problème relaxé dépend du vecteur u initial

Ainsi, il est beaucoup plus facile à résoudre mais dépend d'un nouveau paramètre d'entrée alors qu'une seule et unique itération du modèle non relaxé suffisait à déterminer le meilleur résultat possible en fonction des paramètres d'entrée. Les deux problèmes sont donc équivalents mais le deuxième, bien que plus rapide à être exécuté, va devoir être exécuté plusieurs fois en faisant varier u afin que la solution trouvée soit la plus proche possible de la réalité.

II. Implémentation de la résolution

1 Implémentations

L'implémentation de la méthode de la relaxation lagrangienne ne pose aucun soucis particulier avec GLPK, la matrice des contraintes coefficient simplement étant sous la forme suivante :

$$(C_1 \quad \cdots \quad C_n)$$

et la seule contrainte étant :

$$\sum_{i=1}^{n} y_i = q$$

Afin de simplifier l'algorithme de résolution, on remarque que la somme des u_i est constante et qu'elle peut donc être isolée du reste du problème, ne dépendant pas des y_i. Ainsi, l'appel à GLPK se fait en souhaitant maximiser non pas $\sum_{j=1}^{n} C_j y_j + \sum_{i=1}^{n} u_i$ mais simplement $\sum_{j=1}^{n} C_j y_j$, puis on rajoute à la valeur trouvée à la fin $\sum_{i=1}^{n} u_i$.

Il nous est donné comme information que pour tout jeu de données :

$$Z^* \leq L^*(u)$$

où Z^* représente la solution du problème non relaxé et $L^*(u)$ représente une solution du problème relaxé en fonction de u. On remarque en particulier que cette dernière dépend de notre vecteur d'entrée u. Tout l'objectif dans la suite est donc de minimiser la valeur trouvée par la fonction objectif du problème relaxé afin que celle-ci se rapproche le plus possible de la valeur exacte que l'on peut trouver avec l'algorithme de résolution non relaxé.

J'ai tenté trois algorithmes différents permettant de minimiser $L^*(u)$. Le premier était de tester avec un vecteur u dont chaque composante a la même valeur. L'avantage de cette méthode est le peu d'itérations qu'elle nécessite. En effet, l'algorithme consiste à commencer avec un vecteur u dont toutes les composantes sont à 1 puis de les diminuer 0,001 par 0,001 au fur et à mesure tant que la corrélation

trouvée diminue. Dès que celle-ci remonte, on arrête le programme et on retourne la corrélation minimale trouvée. On a donc au maximum 1000 appels à la résolution. C'est le net avantage de cette résolution : peu d'appels et donc un temps total de résolution très faible. Le problème est qu'on peut faire mieux en résultat final trouvé : pour un jeu de tests se basant sur les corrélations calculées à partir des valeurs journalières du CAC40 sur un an et avec 10 actifs sélectionnés, le résultat trouvé était une corrélation de 31,92 quand le résultat trouvé avec l'algorithme non relaxé était de 29,5. J'ai alors réfléchi à faire varier chaque composante du vecteur d'entrée indépendamment les unes des autres afin qu'une fois atteint un minimum avec chaque composante égale, on simule chaque composante du vecteur avec un nombre aléatoire choisi entre 0 et le minimum atteint avec chaque composante égale. Exemple : si le minimum trouvé avec chaque composante égale est 0,36, on choisissait un nombre entre 0 et 0,36. Cela était censé permettre d'économiser encore un certain nombre de calculs par rapport à ma dernière méthode expliquée ultérieurement. Seulement, l'inconvénient majeur de celle-ci est qu'afin de minimiser au maximum le résultat trouvé, j'ai du faire jusqu'à 10 000 itérations avec différentes composantes aléatoirement choisies pour ensuite en extraire le minimum trouvé et obtenir un résultat approximativement acceptable. Et finalement, j'ai eu deux gros problèmes :

- Le résultat trouvé n'est jamais strictement le même puisqu'à chaque nouveau lancement de l'algorithme, ce sont de nouvelles composantes qui sont choisies aléatoirement
- Le résultat trouvé n'est absolument pas satisfaisant par rapport au premier modèle puisqu'en moyenne, pour les mêmes entrées que ma première méthode, la valeur minimale trouvée tourne aux environs de 50, ce qui est bien loin de 31,92 que je trouvais déjà conséquent par rapport à la valeur théorique de 29,5.

Finalement, j'ai donc implémenté une troisième méthode et c'est sur celle ci que tous mes tests se sont basés dans la suite. Il s'agit d'une méthode très proche de l'algorithme glouton. Et celle-ci donne des résultats très satisfaisants.

2 Greedy Heuristic

L'algorithme glouton (ou greedy algorithm en anglais) est un algorithme qui suit le principe de faire, étape par étape, un choix optimum local dans l'espoir d'obtenir un résultat optimum global satisfaisant.
Ce type d'algorithme est souvent utilisé quand le coût associé à un algorithme de résolution exact est trop important, ce qui est notre cas ici. L'inconvénient majeur de ce type d'algorithme (NP-complet) est que sa durée de résolution augmente substantiellement lorsque la taille de l'entrée augmente. Ainsi, dans mon premier algorithme avec chaque composante égale, il n'y avait qu'une seule boucle faisant des appels successifs (complexité maximale $\frac{1}{pas}$) tandis que désormais, il va falloir imbriquer deux boucles pour faire varier chaque composante indépendamment et la complexité maximale sera de $n * \frac{1}{pas}$.

Nous décrivons maintenant l'algorithme utilisé dans le cas de la résolution de notre problème d'optimisation de l'indexfund.

```
Pour chaque composante de u, faire
       valeurTmp = calculerSolution(n, q, vecteurU);
       ajouter(valeurTmp, valeursTmp);
       Tant que valeurTmp <= min(valeursTmp), faire
              diminuerComposanteActuelleDeU(pas);
              valeurTmp = calculerSolution(n, q, vecteurU);
              ajouter(valeurTmp, valeursTmp);
       Fin tant que
       ajouter(mins, stocker(min(valeursTmp)));
Fin pour

retourner min(mins);
```

Le nombre d'appels dans le pire des cas de l'algorithme de calcul de la solution est de : 1 + n*(n* 1/pas).
C'est pour cela que contrairement aux algorithmes précédents, mon pas choisi n'était pas de 1/1000 mais de 1/100.
De plus, lorsque n passait de 40 pour le CAC40 à 100 pour le FTSE, dans le pire des cas, l'augmentation du nombre d'appels dans le pire des cas était de :

$$\frac{1 + 100 * (100 * 100)}{1 + 40 * (40 * 100)} = 6{,}25$$

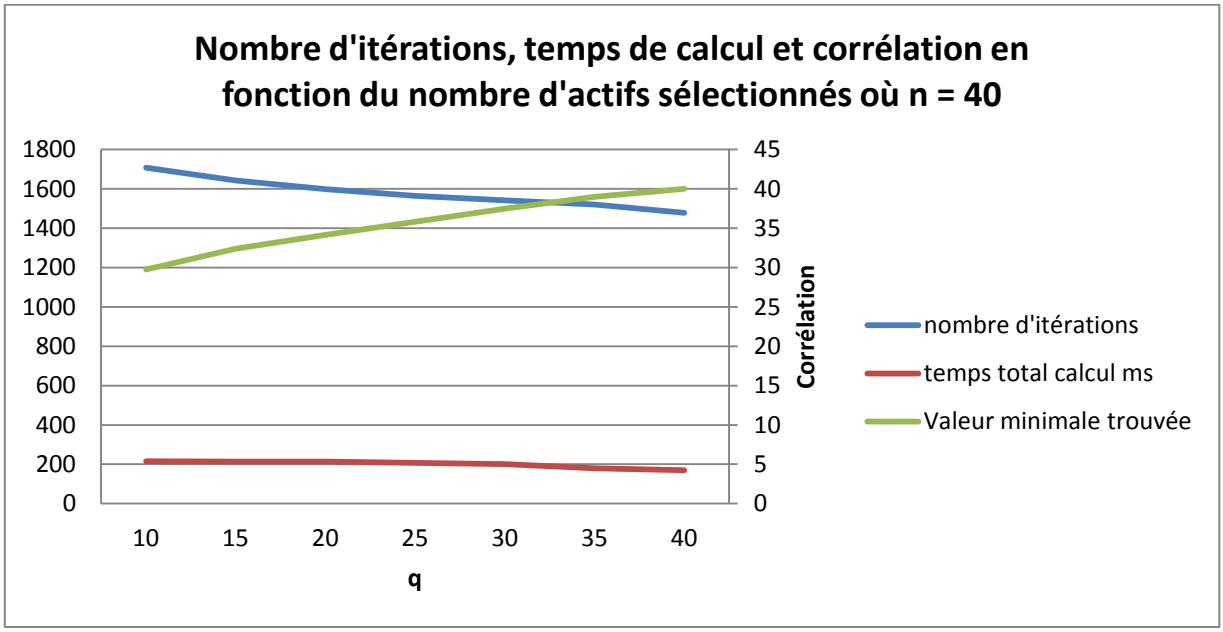
quand n est multiplié par 2,5.

Comparé au premier modèle où le nombre d'appels augmente linéairement en fonction de n, on est ici sur un algorithme en complexité temporelle en $O(n^2 * complexité(algorithme\_resolution))$. Sachant que la complexité de l'algorithme de résolution est fonction de la taille de la matrice d'entrée (ici $1 * n$), on a globalement une complexité en $O(n^3)$.

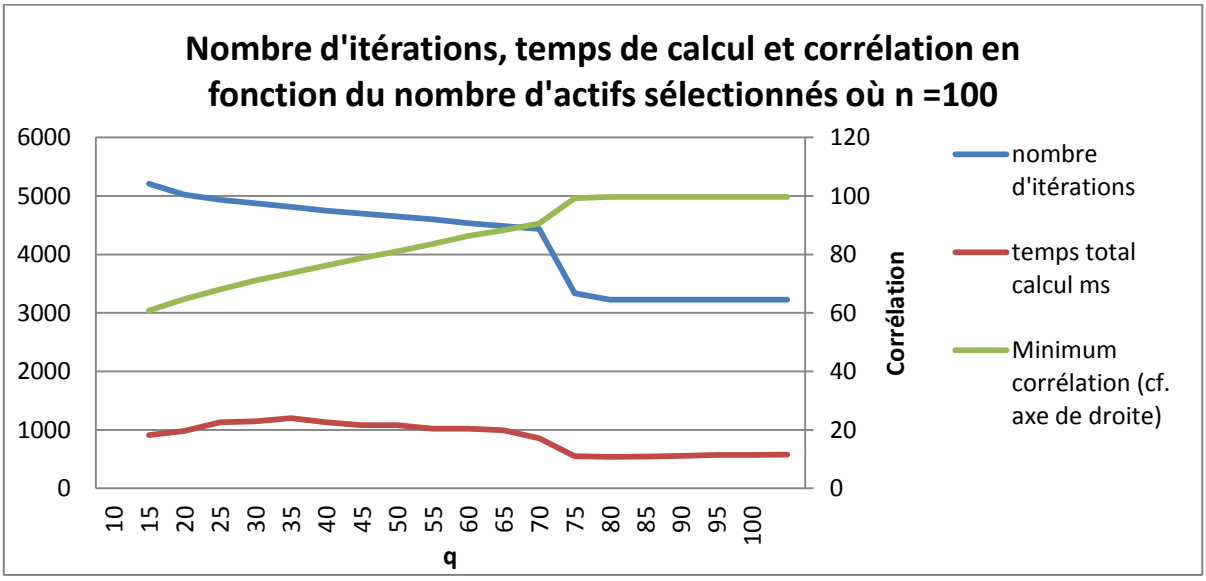
Ce dernier algorithme m'a alors apporté entière satisfaction puisque pour le cas du CAC40 avec 10 actifs sélectionnés, j'ai obtenu comme corrélation 29,71 pour une corrélation par le modèle non relaxé de 29,5.

3 Résolutions et explications

Avec ce dernier algorithme apportant des solutions très satisfaisantes, j'ai réalisé quelques statistiques pour n = 40 sur les données du CAC40 et n = 100 sur les données du FTSE100.

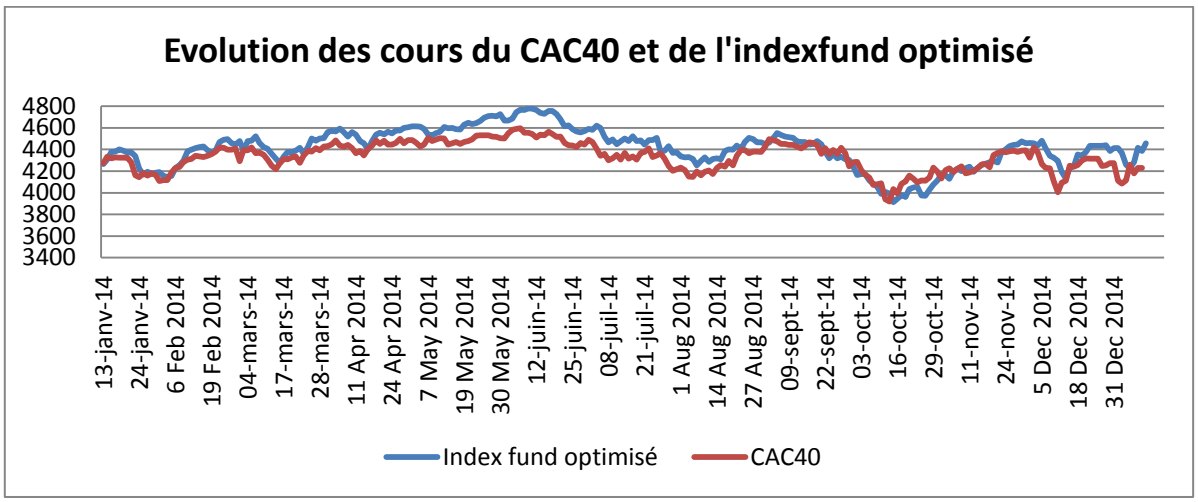


Comme on pourrait s'y attendre, plus l'on augmente le nombre d'actifs sélectionnés dans le portefeuille, plus l'on se rapproche du maximum de corrélation possible, égale au nombre d'actifs (ici : 40). Notre algorithme optimise très bien puisque déjà avec 10 actifs choisis, nous avons tout de même une corrélation proche de 30, ce qui, si nous n'optimisions pas notre portefeuille, reviendrait donc à devoir investir dans 30 actifs. De plus, on observe que plus l'on sélectionne d'actifs, plus le nombre d'itérations et le temps total des calculs diminuent.



En multipliant la taille de notre portefeuille par 2,5 (FTSE100), on observe bien un nombre d'appels plus conséquent, avec un facteur environ égal à 2,5, et de même pour la durée de calcul. On observe cependant une singularité pour n = 100 : à partir de n = 75, le nombre d'itérations chute brutalement et la corrélation maximale trouvée atteint d'un coup son maximum (100), pour tout nombre d'actifs sélectionnés supérieur ou égal à 75, cela reste identique. Seul le temps de calcul remonte un peu en raison d'une taille de matrice plus grande et donc d'une vitesse de résolution qui s'en trouve ralentie.

Pour conclure cette partie sur l'indexfund, j'ai recherché à voir ce que donnerait l'évolution de mon indexfund optimisé en sélectionnant 10 actifs parmi les 40 du CAC40 pour comparer celui-ci par rapport à l'évolution enregistrée du CAC40. En partant du principe que mon indice partait du même nombre de points que le CAC40 à la date initiale de mon calcul sur un an (13 janvier 2014 - 12 janvier 2015), après avoir récupéré les valeurs des 10 actifs sélectionnés par mon programme sur la période considérée ainsi que celles du CAC40, j'ai pu obtenir le graphe suivant :



Cela vient confirmer que l'indexfund optimisé à 10 actifs est un excellent représentant du portefeuille représenté par le CAC40 sur la période considérée, et même que mon indexfund avec les actifs sélectionnés est même meilleur (puisque vaut plus de points) que le CAC40 quasiment sur toute l'année 2014, sauf sur la période septembre 2014 - novembre 2014, sans forcément être très en dessous du CAC40, contrairement à celui-ci qui avait tout de même 200 points de moins que notre indexfund optimisé au mois de juin 2014.

Conclusion

Au travers de ces trois problèmes, nous avons pu mettre en pratique la résolution de problèmes linéaires et toucher à des problématiques concrètes en optimisation d'actifs au sein d'un portefeuille.

Que ce soit pour la réduction des scénarios pertes importantes dans un portefeuille, pour maximiser le rendement d'un portefeuille de Markovitz sous les contraintes d'une ValueAtRisk ne dépassant pas une certaine valeur, ou pour choisir au sein d'un ensemble d'actifs un certain nombre tel que les sélectionnés soient le plus représentatifs possibles de l'ensemble, les sujets m'ont permis de mieux comprendre l'intérêt de poser des systèmes linéaires en finance et de prendre conscience des problématiques de durée des résolutions, très importantes sur les marchés.

Quand on sait que certains logiciels doivent prendre des décisions en maximum 20 microsecondes souvent liées à des opportunités d'arbitrage le temps que l'information sur un cours donné s'actualise sur toutes les places financières du monde, on comprend l'importance d'optimiser au maximum nos algorithmes et de faire des choix stratégiques entre minimiser le risque (qui prend plus de temps de calcul) et minimiser le temps de calcul (qui augmente le risque pris).

Ainsi, à partir du travail réalisé dans la première partie, on pourrait en déduire les paramètres les mieux adaptés répondant à la fois aux exigences en temps de calcul et en minimisation du risque pris sur les marchés. A partir du travail réalisé dans la deuxième partie, on pourrait en déduire, pour un portefeuille donné, le nombre d'actifs nécessaires pour le représenter quasiment parfaitement. Par exemple, dans notre étude sur le fond indiciel FTSE100, on voyait clairement qu'il n'est pas nécessaire d'investir dans plus de 75 actifs pour le représenter parfaitement.

Bibliographie

Ci-dessous mes références principales classées par nom de domaine croissant :

http://www.abcbourse.com/apprendre/19_value_at_risk.html
http://alineasofts.free.fr/extern/GLPK_memo.txt
http://lists.gnu.org/archive/html/help-glpk/2009-11/msg00074.html
http://www.memoireonline.com/08/09/2572/m_Choix-des-portefeuilles-une-generalisation-de-lapproche-MV7.html
http://glpk-java.sourceforge.net/apidocs/org/gnu/glpk/GLPK.html
http://www.iecn.u-nancy.fr/~scheid/Enseignement/heuristiques.pdf
http://www.ise.ufl.edu/uryasev/files/2011/11/Credit_risk_optimization.pdf
http://www.uvm.edu/~pdodds/files/papers/others/1977/cornuejols1977.pdf
http://fr.wikipedia.org/wiki/Algorithme_glouton
http://fr.wikipedia.org/wiki/Fonds_indiciel

Ne figurent pas dans les détails mais je tiens à les citer pour toutes les données brutes que j'y ai récupéré :

http://www.abcbourse.com/download/historiques.aspx
http://uk.finance.yahoo.com