

Projet d'Optimisation multi-critère: Optimisation de portefeuilles

Alex FOUGEROUX et Paul-Aimé ROBERT, ir4 ESAIP, 12/2025

Résumé:

Dans le cadre du cours de majeur optimisation multi-critère dispensé à l'ESAIP par M. Fabien LIONTI nous avons réalisé en tant que rendu final un projet d'optimisation multi critère. Pour le mener à bien, nous avons donc implémenté deux algorithmes (NSGA 2 ainsi qu'un algorithme Métaheuristique développé par nos soins). Toutes les informations complémentaires concernant la résolution de ce projet sont donc dans ce rapport.

Introduction au problème :

Nous nous attaquons ici à l'optimisation d'un portefeuille d'actifs selon plusieurs objectifs et sous contraintes. Comme dans la majorité des cas d'optimisation multi-critères, on cherche à minimiser les fonctions objectives. Ici notre variable est le portefeuille d'actif, on peut le voir comme l'ensemble des poids qui pondèrent l'importance de chaque actif. Ce qui correspond à un vecteur w de dimension n , le nombre d'actif disponible, appartenant à \mathbb{R}^n :

$$w \in \mathbb{R}^n | w_j \in [0, 1]$$

L'idée est donc d'obtenir un ensemble des solutions acceptables.

Puis de réduire cette ensemble à un front de pareto; afin de proposer à l'utilisateur un ensemble de solution pareto optimal.

Objectifs :

Nous cherchons donc à minimiser trois fonction :

fonction objectif 1 :

Première fonction à minimiser, le rendement attendu (négatif). Autrement dit, on cherche à maximiser le rendement attendu.

$$f_1(w) = -w^T \mu$$

Le rendement attendu est donc le vecteur des poids w multiplié par μ le rendement. Le rendement est lui-même calculé en nous basant sur les données passées sur l'évolution des actifs.

Le rendement attendu représente donc la croissance potentielle des actifs sur lesquels sont attribués des poids.

Implémenté comme fonction : algo.func_obj_1_yield.get_yield

fonction objectif 2 :

Seconde fonction à minimiser, le risque.

$$f_2(w) = w^T \Sigma w$$

Le risque est calculé à partir de notre vecteur de poids w , et de la matrice de covariance Σ , qui comme mu est calculée à partir des données passées de l'évolution des actifs.

La matrice de covariance permet de quantifier la variation de chaque actif en fonction des autres actifs et du temps.

Le risque représente donc la possibilité qu'une variation importante (qu'on peut donc interpréter comme imprévue) ne survienne sur les actifs sur lesquels sont attribués des poids.

Implémenté comme fonction : `algo.func_obj_2_risk.get_risk`

fonction objectif 3 :

Troisième fonction à optimiser, le coût de transaction.

$$f_3(w_{t-1}, w_t) = C \sum_{i=0}^n |w_{t,i} - w_{t-1,i}|$$

Le coût de transition se calcule à partir de la répartition courante de poids, c'est la somme de la différence des poids entre la répartition courante des poids et la nouvelle répartition des poids, le tout multiplié par C, le coût proportionnel (déterminé arbitrairement, à adapter pour correspondre au maximum à la réalité).

Ce coût représente donc les frais associés au mouvement de fonds.

Implémenté comme fonction : `algo.func_obj_3_trans_cost.get_trans_cost`

Contraintes :

La résolution de ce problème se fait sous contrainte, ici nous avons deux contraintes principales qui sont la contrainte d'égalité ainsi que la contrainte de cardinalité.

Mais ces contraintes sont aussi associées à une autre contrainte, les poids ne doivent pas être négatifs.

Aussi les contraintes sont représentées comme une égalité nulle par convention.

Elles sont implémentées au sein de la class problème (pour NGSA 2) : `algo.ngsa2.WalletProblem`

Et la contrainte d'égalité uniquement, est implémentée directement dans le code de l'algorithme génétique : `algo.genetic`.

Contrainte d'égalité :

La contrainte d'égalité concerne la valeur des poids, la somme de l'ensemble des poids doit être égale à 1.

$$g_1(w) = \sum_{i=0}^n (w_j) - 1$$

$$C_1 : \{g_1(w) = 0\}$$

Cette contrainte est nécessaire si on souhaite que w soit bien une répartition et reste bien normé de la même manière pour tout l'ensemble des solution. Quelque soit l'algorithme utilisé.

Contrainte de cardinalité :

La contrainte de cardinalité concerne le nombre de poids non nul dans un vecteur w . On choisit donc un nombre K arbitraire.

$$g_2(w) = \sum_{i=0}^n (\mathbb{I}(w_i \neq 0)) - K$$

Cependant, pour des raisons techniques, il arrive que des poids soit très proche de zéro. Ils serait donc considéré comme compté dans la cardinalité, on implémente donc une tolérance δ_{tol} (valeur arbitraire).

$$g_2(w) = \sum_{i=0}^n (\mathbb{I}(w_j > \delta_{tol})) - K$$

$$C_2 : \{g_2(w) = 0\}$$

Dans notre implémentation nous avons opté pour une cardinalité de 5 et une tolérance de 0,0001.

Résolution :

Ce problème n'est pas convexe, une solution peut représenter un minimum local sans être la solution optimale (minimum globale). Cette non-convexité est principalement apportée par les contraintes, notamment la contrainte des coûts de transaction (avec une valeur absolue). Par ailleurs, la valeur absolue implique aussi une probable discontinuité.

Pour l'ensemble de ces raisons nous avons choisi d'opter uniquement pour des algorithmes génétiques qui permettent de résoudre le problème sans tenir compte de potentiel discontinuité et d'approcher le front de pareto idéal au plus proche malgré la non-convexité du problème.

Par le fait que nous ayons plusieurs fonctions à optimiser, il est très probable qu'une solution ne domine pas l'ensemble des autres. Nous aurons donc à l'issue de la résolution un front de pareto dominant l'ensemble des solutions proposées.

Optimisation bi-objectif :

Tout d'abord nous résolvons une version simplifiée du problème avec seulement deux fonctions objectives, le rendement attendu et le risque. Et sans la contrainte de cardinalité.

Ce qui se traduit par une optimisation du risque et du rendement attendu sans prendre en compte le nombre de poids à faire varier.

Approche 1, algorithme métaheuristique :

Pour résoudre ce problème nous avons implémenté un algorithme métaheuristique conçu nous même.

Voici quelques détails concernant sa conception.:

Initialisation : L'ensemble des solutions est initialisé de manière aléatoire (et indépendante).

Sélection : Nous utilisons une sélection nominale, nous sélectionnons donc les k meilleurs solutions (k valeur arbitraire) en nous basant sur la distance à l'origine (l'utopie soit $f_1 = 0, f_2 = 0$). Cette sélection permet de choisir les solutions plus proches du front de Lorenz (solutions favorisant les valeurs de compromis entre les deux fonctions objectives), et ultimement fait converger les solutions vers un point unique.

Reproduction, mutation et remplacement : Nous basons donc la population de l'itération suivante sur les k solutions sélectionnées. Ces solutions représentent donc les k premiers éléments de la population pour l'itération suivante. Chaque autre solution est une combinaison de deux parents issue des k solutions sélectionnées. Où chaque poids est une synthèse de la combinaison des poids des parents (par une opération choisie aléatoirement parmi deux possibilités, soit la moyenne, soit la norme). Puis, nous faisons muter aléatoirement ces poids afin de favoriser l'exploration. Cette mutation est primordiale, puisqu'elle permet de compenser la forte favorisation de l'exploitation, due à la sélection nominale.

Lorsque les solutions stagnent ou lorsque le maximum d'itération est atteint, nous reprenons la population sélectionnée et calculons le front de pareto de cet ensemble de solution.

Approche 2, NGSA 2 :

Nous avons aussi implémenté un algorithme NGSA 2. À la différence de notre algorithme, NGSA 2 essaie d'approcher au mieux le front de pareto dans son ensemble.

Comparaison :

Nous avons plusieurs points de comparaison, outre le temps d'exécution (beaucoup plus court chez NGSA 2). L'algorithme NGSA 2 est plus consistant que notre algorithme génétique. Puisqu'il donne systématiquement les mêmes solutions pour le même problème. En outre, on observe aussi que l'algorithme NGSA 2 donne un plus grand front de pareto que notre algorithme (pour 200 itération et une sélection de 100 solution, NGSA 2 donne en moyenne environ un front de pareto de 37 (plus ou moins 5), là où notre algorithme en donne un de 20 (plus ou moins 10). Ce qui est expliqué par la méthode de sélection (sur notre algorithme les solutions sont poussées à se regrouper autour des points favorisant le compromis).

Optimisation tri-Objectif :

Le problème :

Pour approcher au mieux le problème nous avons donc utilisé l'algorithme NGSA 2, mais en ajoutant le coût de transaction et la contrainte de cardinalité.

Critère de robustesse :

Cependant notre approche n'est pas parfaite et est limité par nos fonctions d'optimisation, puisque μ et Σ (le rendement et la matrice de covariance), ne sont que des estimations historiques. Ces paramètres impliquent donc une incertitude importante sur les prédictions.

Il serait envisageable de pallier à ça en utilisant la dominance stochastique de second ordre, pour comparer les solutions proposées (en comparant l'intégrale des fonctions de distributions cumulatives). Mais cela nous obligerait à générer et simuler des scénarios afin de comparer les performances des solutions et établir les dominances stochastiques.

Implémentation Technique :

Pour traduire la modélisation mathématique en un outil fonctionnel d'aide à la décision, nous avons opté pour une architecture logicielle modulaire basée sur l'écosystème python comme demandé.

La stack :

Le projet est structuré autour d'une séparation claire entre le backend et l'interface utilisateur:

- **Le backend** se situe dans le dossier scripts/ : il contient la logique métier pure, notamment les fonctions de récupération des données, les calculs matriciels de risque/rendement et les implémentations des algorithmes génétiques (NGSA2 et Génétique Standard)
- **Le frontend** se situe dans le dossier src/ : développé avec Streamlit, il assure l'interaction avec l'utilisateur. Ce framework à été choisi pour sa capacité à

prototyper rapidement des applications de data-science sans nécessiter de développement web complexe.

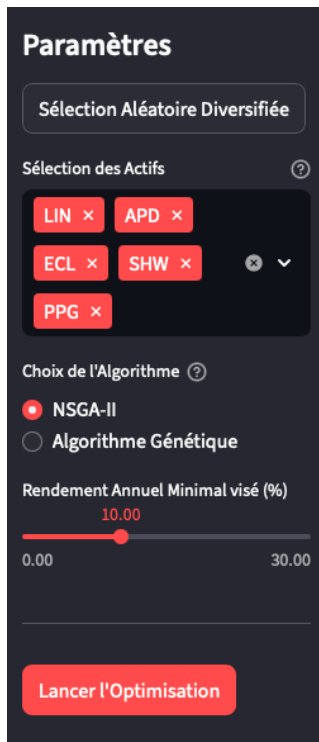
Pour la visualisation, nous avons utilisé Plotly. Cette bibliothèque permet de générer des graphiques dynamiques et interactifs, ce qui est indispensable pour explorer finement une frontière de Pareto.

Présentation de l'interface :

L'objectif de l'interface est de masquer la complexité algorithmique pour permettre à l'investisseur de se concentrer sur sa prise de décision. Le Dashboard est organisé en trois zones principales.

Paramétrage et sélection :

Le panneau latéral gauche permet de définir l'univers d'investissement et les contraintes.



On peut sélectionner les actifs manuellement ou utiliser notre fonctionnalité de “Génération Aléatoire Diversifiée”. Cette fonction assure une sélection d'actifs répartis équitablement entre différents secteurs GICS, évitant ainsi un biais sectoriel initial.

Ensuite, on a également le choix de l'algorithme utilisé soit NSGA2 ou une approche génétique standard.

Enfin, nous avons le choix d'une contrainte de rendement (rmin) que l'on peut gérer via un slider ce qui permettra de filtrer dynamiquement les solutions proposées.

Visualisation des résultats :

L'onglet principal “Portefeuille Optimal” présente les résultats une fois l'optimisation terminée.



En haut de page, les métriques essentielles (Rendement espéré, volatilité, Ratio de Sharpe) sont affichées pour le portefeuille “élu”. Ce choix est effectué automatiquement par le système : parmi toutes les solutions respectant le rendement cible, l’algorithme sélectionne celle qui minimise le risque.

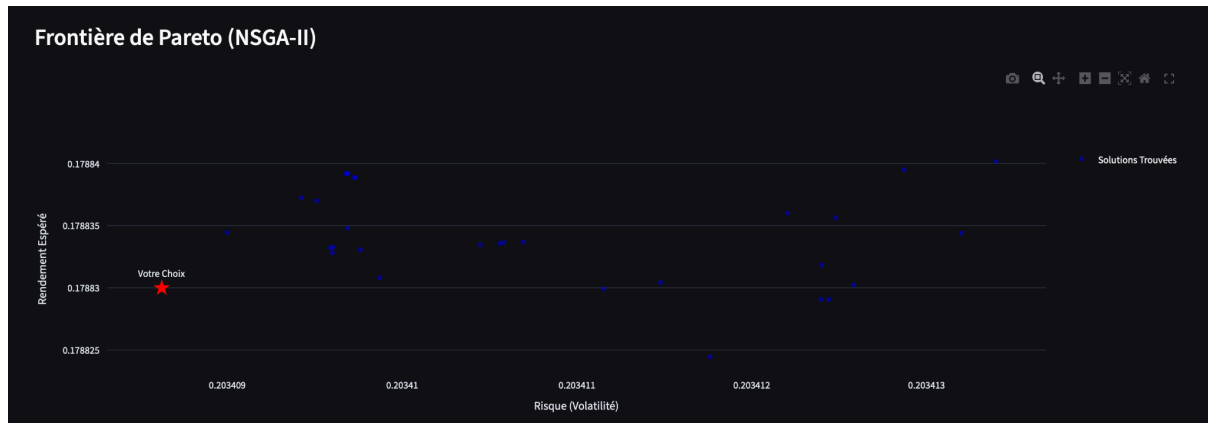
Ensuite, pour valider la diversification, un graphique “camembert” décompose le portefeuille par secteur macro-économique puis par actif. Cela permet de vérifier en un coup d’œil que le portefeuille ne repose pas sur une seule industrie.

Analyse des résultats:

Nous avons testé notre modèle sur un panier de 15 à 20 actifs, incluant des valeurs technologique (volatiles) et des valeurs de consommation (défensives), avec un objectif de rendement annuel de 10%.

Interprétation de la Frontière Efficiente

Le graphique ci-dessous représente l’ensemble des solutions viables trouvées par l’algorithme NSGA2



Le nuage de points (Frontière) nous permet d'observer la formation d'une courbe convexe caractéristique. Chaque point bleu représente un portefeuille optimal au sens de Pareto : pour un niveau de risque donné, il est impossible de trouver un meilleur rendement. L'algorithme a correctement éliminé les solutions dominées (en bas à droite du graphique). Ensuite, pour la solution (étoile rouge) le point mis en évidence correspond à la recommandation finale. Dans notre scénario, pour un rendement cible de 10%, l'algorithme a identifié une combinaison d'actifs permettant de contenir la volatilité.

Analyse de la composition :

L'analyse des poids montre la pertinence de l'optimisation sous contraintes. L'algorithme tend à allouer une part significative aux actifs performants (souvent Tech ou Energie) pour satisfaire la contrainte de rendement $f1(w)$. Pour contrebalancer ce risque l'algorithme choisit de petites positions dans des secteurs décorrélés (Santé, Utilities) pour minimiser la variance globale $f2(w)$.

Conclusion :

Pour résoudre ce problème nous avons essayé plusieurs algorithmes ce qui nous a amené à conserver NSGA 2, qui permet de représenter le plus grand éventail de solutions optimisées, tout en ayant la flexibilité pour performer sur ce problème. Cependant, en l'état toute l'incertitude n'est pas pleinement prise en compte, il pourrait être intéressant, avec une plus grande capacité de calcul, d'implémenter une SSD afin de sélectionner des solutions meilleurs encore.

Annexe :

Pour l'intégration des formules nous avons utilisé l'outil <https://latex2image.io/eraut.com/>.