Optimisation des Stratégies d'Investissement chez AlgoInvest&Trade

Analyse et Amélioration des Algorithmes d'Investissement

*Présentation réalisée par **Christophe RENARD***

***GitHub :** https://github.com/fox7725/P7_AlgoInvest_Trade*

Table des matières :

INTRODUCTION

Introduction à l'Optimisation des Stratégies d'Investissement

Contexte :

AlgoInvest&Trade est une société financière de premier plan qui se spécialise dans l'élaboration de stratégies d'investissement sophistiquées pour maximiser les retours sur investissement de ses clients.
Face à un marché en constante évolution, l'entreprise cherche à exploiter les avancées technologiques pour rester compétitive.

Défi :

Le principal défi réside dans la nécessité de développer un algorithme capable d'identifier les combinaisons d'investissements les plus rentables, en tenant compte d'un budget fixe et de la variabilité des retours sur investissement.
L'objectif est de fournir une solution qui maximise les bénéfices des clients sur une période de deux ans, tout en respectant les contraintes budgétaires.

Notre Mission :

En tant que développeur nouvellement intégré à l'équipe d'AlgoInvest&Trade, ma mission consiste à traduire les besoins commerciaux en solutions techniques efficaces.
Ce projet représente une opportunité unique de combiner expertise financière et compétences en programmation pour générer une valeur ajoutée significative pour nos clients.

Approche :

Nous aborderons le problème en deux phases : une analyse initiale via un algorithme de force brute pour comprendre la complexité du défi, suivie par le développement d'une solution optimisée capable de fournir des résultats rapides et précis.
L'accent sera mis sur l'efficacité, la précision et la capacité de l'algorithme à s'adapter à un grand volume de données.

Objectif de la Présentation :

Présenter notre approche méthodique pour relever ce défi, en détaillant les étapes clés du processus de développement, de l'analyse à l'optimisation.
Partager les résultats obtenus et discuter des perspectives d'amélioration continue de nos algorithmes d'investissement.

Fonctionnement de l'Algorithme de Force Brute

Comprendre l'Algorithme de Force Brute

Définition :

L'algorithme de force brute est une méthode de résolution de problèmes qui consiste à générer toutes les combinaisons possibles pour trouver la solution optimale. Pour notre cas d'investissement, cela signifie explorer toutes les combinaisons d'actions que nous pouvons acheter dans le cadre du budget alloué.

Exemple Simple :

Imaginons trois actions disponibles :
Action A à 100€ avec un profit de 10%
Action B à 200€ avec un profit de 20%
Action C à 150€ avec un profit de 15%
Avec un budget de 250€, l'algorithme évaluera toutes les combinaisons possibles (A, B, C, AB, AC, BC, ABC) pour identifier celle qui maximise le profit sans dépasser le budget.

Limites de la Force Brute : Complexité Exponentielle et Temps d'Exécution

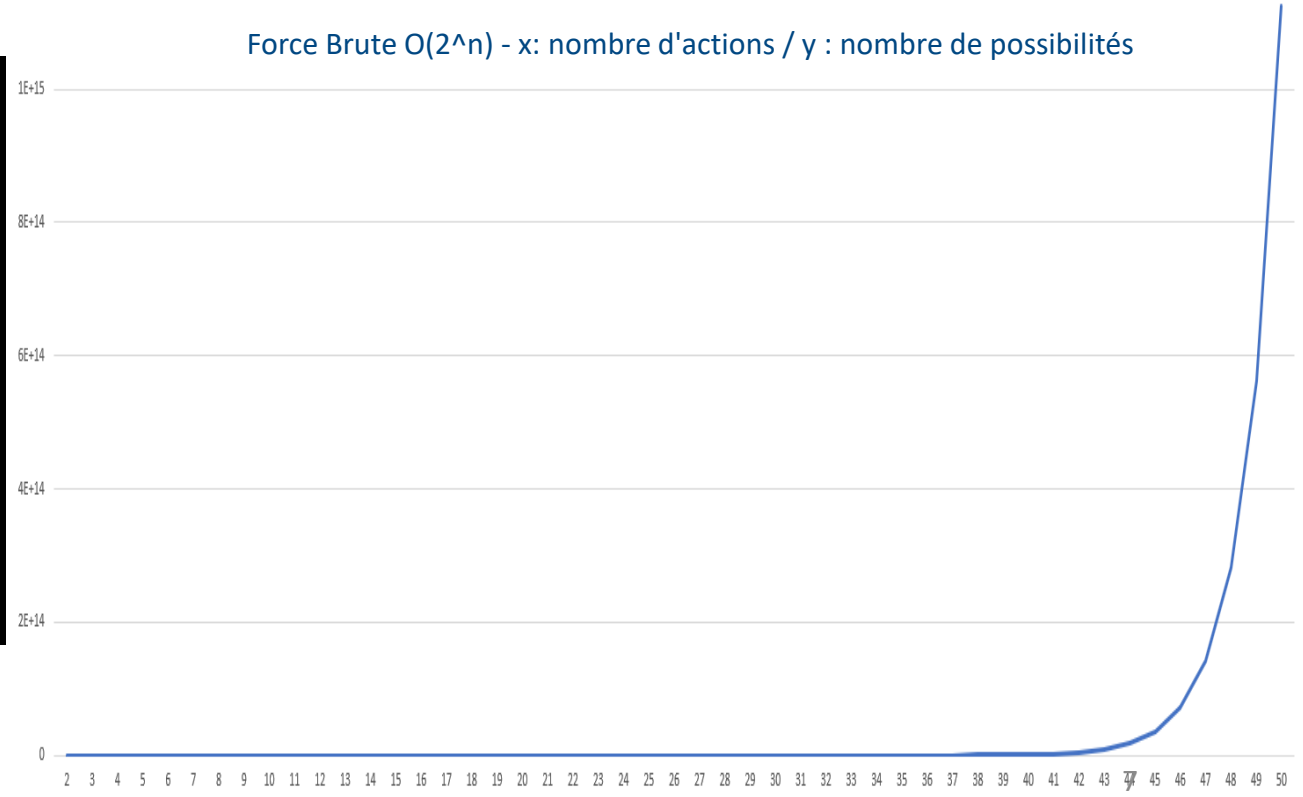
Complexité Exponentielle :

La complexité de cet algorithme est exponentielle, notée **O(2^n)**, où n est le nombre d'actions disponibles. Cela signifie que le temps nécessaire pour exécuter l'algorithme double avec chaque action ajoutée à notre ensemble de données.

Impact sur le Temps d'Exécution :

Pour un petit nombre d'actions, l'algorithme peut fonctionner rapidement. Cependant, à mesure que le nombre d'actions augmente, le temps d'exécution devient rapidement impraticable. Par exemple, pour 20 actions, il y a 1,048,576 combinaisons possibles à évaluer (2^{20} => Pour chaque action, on a deux choix : soit on inclue l'action dans le portefeuille, soit on ne l'inclue pas. Cela crée une situation binaire - oui ou non - pour chaque action.).

```
Variables globales
    portefeuille <- 500
    actions <- depuis Actions.csv : actions avec un nom, un prix et un profit en pourcentage du prix

Fonction brute_force(actions, budget_max)
    meilleure_combinaison <- []
    meilleure_combinaison <- 0
    Pour r allant de 1 à la longueur de actions (inclus)
        Pour chaque combinaison de r actions dans actions
            Calculer cout_total de la combinaison
                Si cout_total est inférieur ou égal à budget_max
                    Calculer profit_total de la combinaison
                    Si profit_total est supérieur à meilleur_profit
                        Mettre à jour meilleur_profit et meilleure_combinaison

Retourner meilleure_combinaison
```



Résultats obtenus avec la Force Brute

Sienna nous a donné un jeu de 20 actions à traiter en Force Brute, le but étant de trouver le meilleur profit sans dépasser le budget de 500€ voici le résultat après 1,37 secondes de travail :

```
Meilleur ensemble d'actions à acheter :
Nom : Action-4 - coût : 70 - profit : 14.0
Nom : Action-5 - coût : 60 - profit : 10.2
Nom : Action-6 - coût : 80 - profit : 20.0
Nom : Action-8 - coût : 26 - profit : 2.86
Nom : Action-10 - coût : 34 - profit : 9.18
Nom : Action-11 - coût : 42 - profit : 7.14
Nom : Action-13 - coût : 38 - profit : 8.74
Nom : Action-18 - coût : 10 - profit : 1.4
Nom : Action-19 - coût : 24 - profit : 5.04
Nom : Action-20 - coût : 114 - profit : 20.52
Pour un total de 10 actions, nous avons un coût total de 498 € et un profit total de 99.08 €
```

Algorithmes optimisés :

- Approche basée sur le tri et la sélection directe
-Approche basée sur la méthode dynamique du sac à dos

Présentation de deux algorithmes optimisés

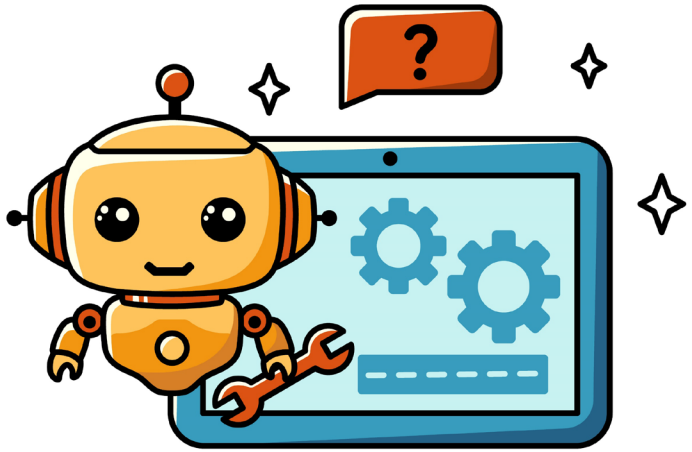
Nous avons exploré deux méthodes d'optimisation pour améliorer notre stratégie d'investissement : l'approche basée sur le tri et la sélection directe, et la méthode dynamique du sac à dos.

Approche basée sur le tri et la sélection

Cette méthode simplifiée effectue un tri décroissant des actions en fonction de leur rentabilité avant de les sélectionner jusqu'à l'épuisement du budget. Elle est intuitive et rapide pour des ensembles de taille modérée.

Méthode dynamique du Sac à Dos *(solution fournie par ChatGPT)*

Une technique avancée qui résout de manière optimale le problème d'allocation de budget pour maximiser le profit, adaptée pour gérer de grands volumes de données avec précision : Dans un tableau dont chaque colonne représente 1€ supplémentaire de budget et chaque ligne représente une action, on estime le choix d'action le plus rentable pour chaque portion du budget. Le calcul par tranche de budget permet d'éviter de refaire le calcul plusieurs fois pour chaque action. Je prends l'action dont le coût est inférieur au budget total et dont le profit est le plus élevé, je déduis son coût du budget, et je recommence avec le budget restant, jusqu'à ce que le budget soit épuisé. La dernière case du tableau contiendra donc la liste des actions offrant la rentabilité maximum pour le budget donné.



Pseudocodes

Approche basée sur la méthode dynamique du sac à dos

```
Variables globales
   n <- nombre d'actions
   budget_max <- 500
   actions  <- depuis dataset.csv : actions avec un nom, un prix et un profit en pourcentage du prix

Initialiser un tableau DP de taille (n+1) par (budget+1) avec des zéros
Pour i allant de 1 à n
   Pour chaque capacité de budget de 1 à budget_max
      Si le coût de l'action[i] <= capacité
         DP[i][capacité] = max(DP[i-1][capacité], profit de l'action[i] + DP[i-1][capacité - coût de l'action[i]])
       Sinon
          DP[i][capacité] = DP[i-1][capacité]
Reconstituer la solution pour trouver les actions sélectionnées
```

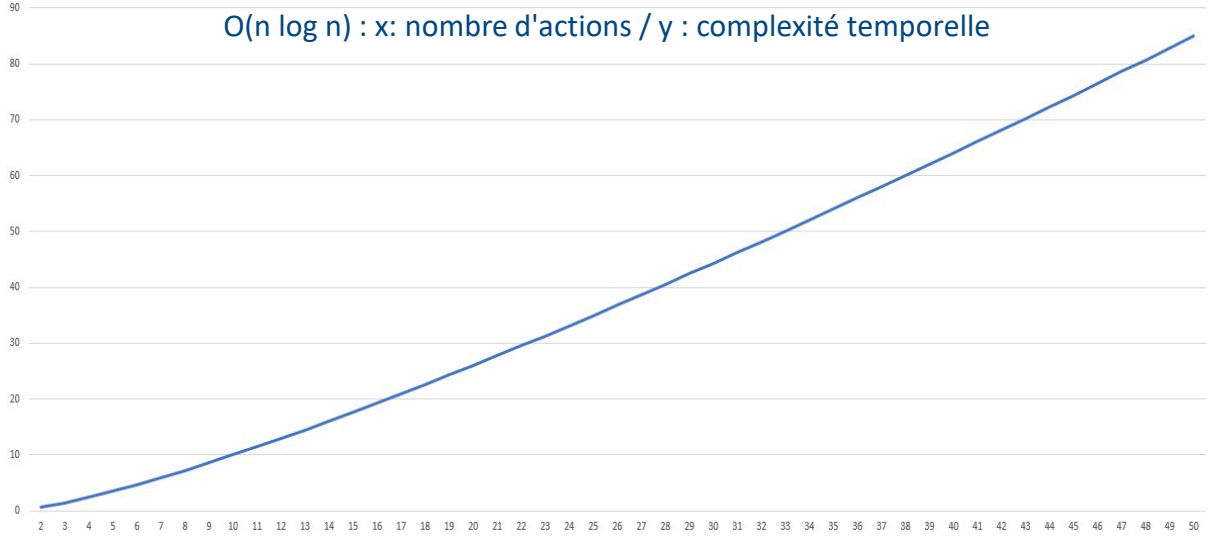
Approche basée sur le tri et la sélection directe

```
Variables globales
   portefeuille <- 500
   actions  <- depuis dataset.csv : actions avec un nom, un prix et un profit en pourcentage du prix
   choix_actions <- []

Convertir le pourcentage de profit en valeur absolue
Convertir les sommes en centimes pour plus de précision
Trier les actions valeur de profit décroissant
Pour chaque action dans les actions triées :
   Si budget >= coût de l'action :
      choix_action += action
      portefeuille -= coût de l'action
Retourner choix_actions
```

Complexité des méthodes optimisées

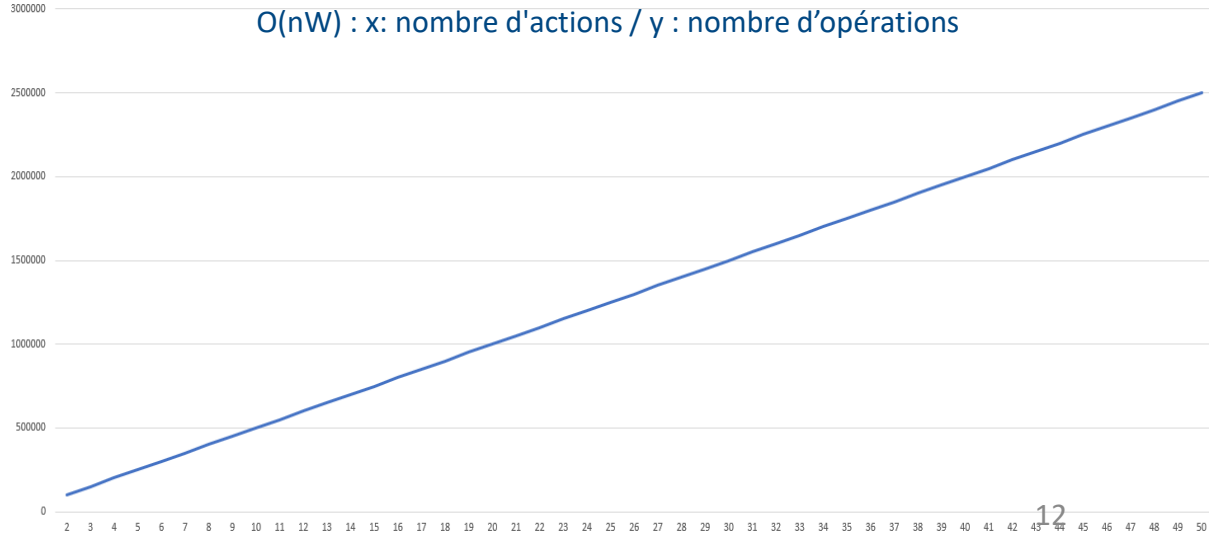
Approche basée sur le tri et la sélection directe

Complexité O(n log n) : Cette complexité vient du fait que l'algorithme doit d'abord trier les actions avant de les sélectionner, ce qui est généralement réalisé par des algorithmes de tri comme le tri rapide (quicksort) ou le tri par fusion (mergesort). Le "log n" indique que chaque division de la liste double le nombre de segments que nous pouvons trier en parallèle, et "n" est le travail nécessaire pour chaque segment.



Approche basée sur la méthode dynamique du sac à dos

Complexité O(nW) où n=nombre d'actions et W= budget en centimes : elle provient de deux boucles imbriquées qui itèrent sur chaque action (n actions) et chaque valeur de budget possible jusqu'à W. Pour chaque paire (action, budget), l'algorithme effectue un calcul constant pour déterminer le profit maximal. Donc, le nombre total d'opérations effectuées par l'algorithme est proportionnel au produit du nombre d'actions (n) et de la capacité maximale du budget (W)



Résultats :

Sienna nous a donné deux jeux de données représentant un total de 1498 actions. Pour faciliter notre étude, nous allons comparer les résultats des deux algorithmes optimisés sur l'ensemble des actions. L'objectif reste le même que pour la Brute Force : Obtenir le meilleur profit sans dépasser le budget de 500€.

Approche basée sur le tri et la sélection directe

Temps de traitement : 0,0045 seconde

```
Valeurs sélectionnées sur l'ensemble des deux fichiers :
Nom: Share-GRUT, Coût: 498.76€, Valeur après 2 ans: 196.61€
Nom: Share-CBNY, Coût: 1.22€, Valeur après 2 ans: 0.48€
Nom: Share-MLGM, Coût: 0.01€, Valeur après 2 ans: 0.0€
totaux : achat 499.99 - profit 197.09
Les deux fichiers ensemble contenaient 1498 actions, nous avons sélectionné 3 actions dans cet ensemble
```

Approche basée sur la méthode dynamique du sac à dos

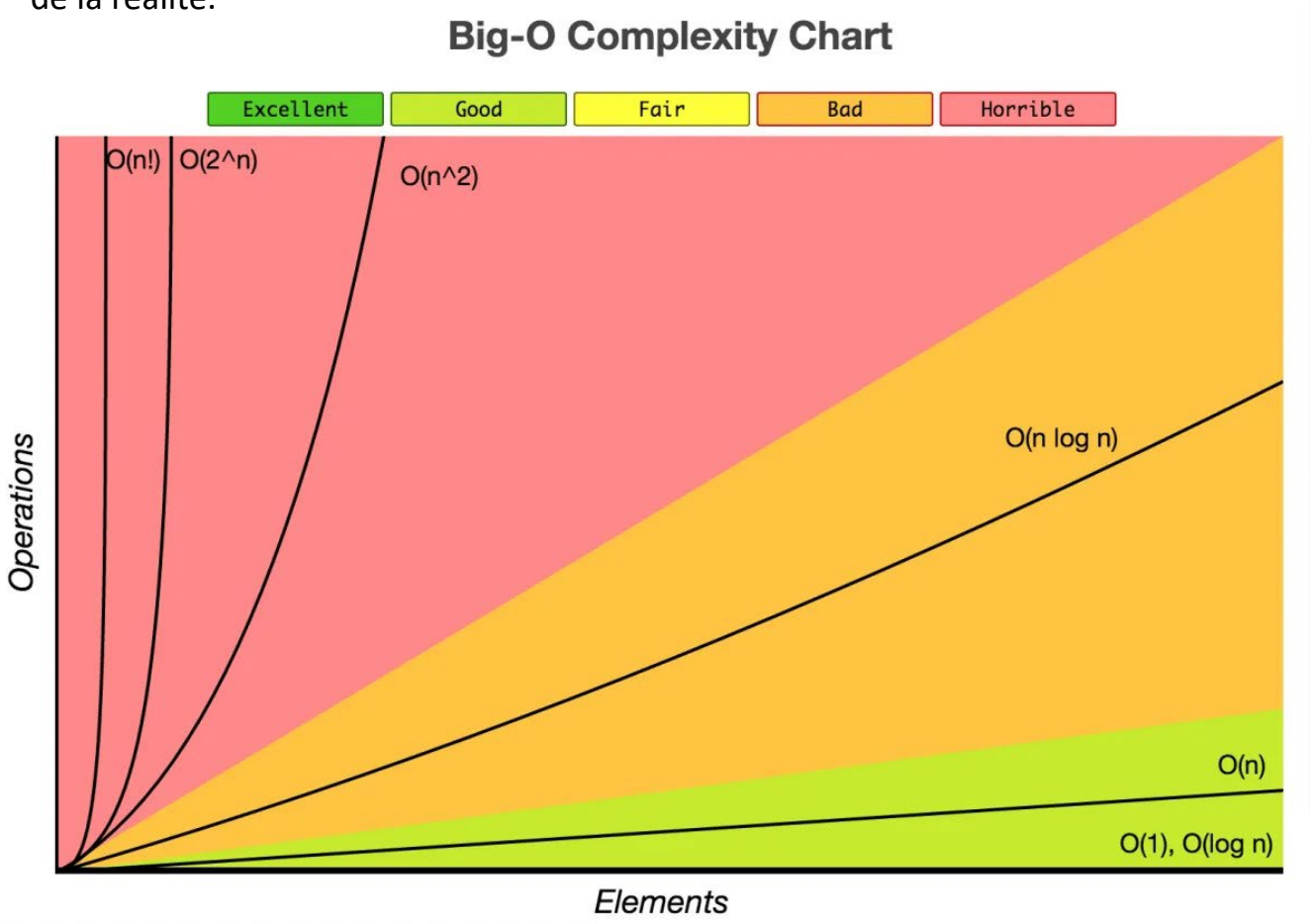
Temps de traitement : 27,38 secondes

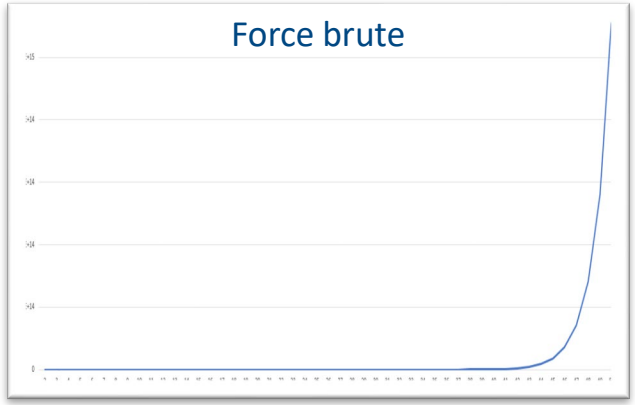
```
Actions sélectionnées :
Nom: Share-FWBE, Coût: 18.3€, Valeur après 2 ans: 7.29€
Nom: Share-ZOFA, Coût: 25.32€, Valeur après 2 ans: 10.07€
Nom: Share-PLLK, Coût: 19.94€, Valeur après 2 ans: 7.96€
Nom: Share-LXZU, Coût: 4.24€, Valeur après 2 ans: 1.68€
Nom: Share-PATS, Coût: 27.7€, Valeur après 2 ans: 11.07€
Nom: Share-NDKR, Coût: 33.06€, Valeur après 2 ans: 13.19€
Nom: Share-ALIY, Coût: 29.08€, Valeur après 2 ans: 11.61€
Nom: Share-JWGF, Coût: 48.69€, Valeur après 2 ans: 19.44€
Nom: Share-LFXB, Coût: 14.83€, Valeur après 2 ans: 5.9€
Nom: Share-XQII, Coût: 13.42€, Valeur après 2 ans: 5.3€
Nom: Share-KMTG, Coût: 23.21€, Valeur après 2 ans: 9.28€
Nom: Share-GHIZ, Coût: 28.0€, Valeur après 2 ans: 11.17€
Nom: Share-NHWA, Coût: 29.18€, Valeur après 2 ans: 11.6€
Nom: Share-MTLR, Coût: 16.48€, Valeur après 2 ans: 6.59€
Nom: Share-GTQK, Coût: 15.4€, Valeur après 2 ans: 6.15€
Nom: Share-FKJW, Coût: 21.08€, Valeur après 2 ans: 8.39€
Nom: Share-MLGM, Coût: 0.01€, Valeur après 2 ans: 0.0€
Nom: Share-WPLI, Coût: 34.64€, Valeur après 2 ans: 13.82€
Nom: Share-ZSDE, Coût: 15.11€, Valeur après 2 ans: 6.03€
Nom: Share-GIAJ, Coût: 10.75€, Valeur après 2 ans: 4.29€
Nom: Share-XJMO, Coût: 9.39€, Valeur après 2 ans: 3.75€
Nom: Share-LRBZ, Coût: 32.9€, Valeur après 2 ans: 13.14€
Nom: Share-IFCP, Coût: 29.23€, Valeur après 2 ans: 11.66€

Total d'achat: 499.96€, Valeur totale après 2 ans: 199.39€
Nombre d'actions sélectionnées : 23
```

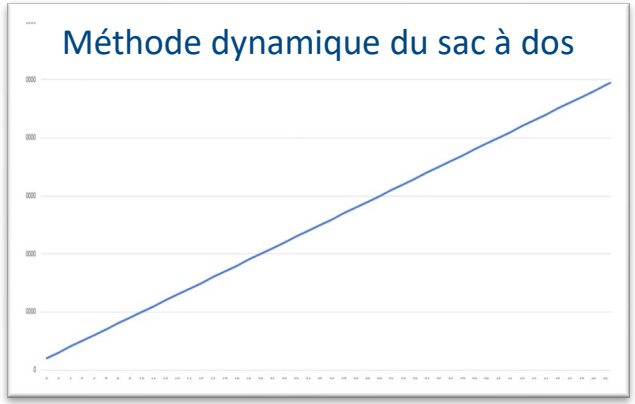
Comparatifs

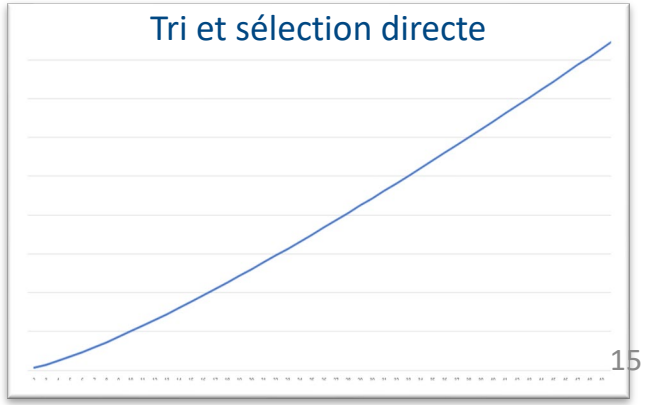
Comparaison des solutions algorithmiques

Nous avons donc comparé trois algorithmes pour la recherche de rentabilité avec un budget donné. L'avantage de la Force brute est la justesse et la précision des résultat, cependant cet algorithme n'est applicable qu'à un très faible nombre de données, la complexité du travail étant exponentielle (**O(2^n)**). On est d'ailleurs dans la zone rouge de la Charte de complexité Big-O. Pour les jeux de données très importants, la méthode dynamique du sac à dos sera la plus intéressant ayant une progression de la complexité linéaire (**O(nW)**), il allie justesse relative du résultat à rapidité de traitement. Pour un jeu moyen d'action, comme celui que nous a donné Sienna, l'algorithme de tri et de sélection directe est le meilleur compromis, puisque même s'il est plus rapide sur cette quantité de donnée (**O(n log n)**), le résultat reste le plus éloigné de la réalité.









Comparaison avec les résultats de Sienna

Affichage des résultats

Méthode dynamique du sac à dos ->

	Sienna	Tri et sélection directe
Dataset 1	Sienna bought: Share-GRUT Total cost: 498.76â,¬ Total return: 196.61â,¬	Valeurs sélectionnées pour le fichier 'optimized/dataset1_Python+P7.csv' : Nom: Share-GRUT, Coût: 498.76€, Profit après 2 ans: 196.61€ Nom: Share-CBNY, Coût: 1.22€, Profit après 2 ans: 0.48€ Nom: Share-MLGM, Coût: 0.01€, Profit après 2 ans: 0.0€ totaux : achat 499.99 - profit 197.09 Le fichier initial contenait 957 actions, nous avons sélectionné 3 actions dans le fichier.
Dataset 2	Sienna bought: Share-ECAQ 3166 Share-IXCI 2632 Share-FWBE 1830 Share-ZOFA 2532 Share-PLLK 1994 Share-YFVZ 2255 Share-ANFX 3854 Share-PATS 2770 Share-NDKR 3306 Share-ALIY 2908 Share-JWGF 4869 Share-JGTW 3529 Share-FAPS 3257 Share-VCAX 2742 Share-LFXB 1483 Share-DWSK 2949 Share-XQII 1342 Share-ROOM 1506 Total cost: 489.24â,¬ Profit: 193.78â,¬	Valeurs sélectionnées pour le fichier 'optimized/dataset2_Python+P7.csv' : Nom: Share-JWGF, Coût: 48.69€, Profit après 2 ans: 19.44€ Nom: Share-MBQU, Coût: 51.46€, Profit après 2 ans: 18.41€ Nom: Share-QEVK, Coût: 49.77€, Profit après 2 ans: 17.11€ Nom: Share-DLNE, Coût: 44.06€, Profit après 2 ans: 16.19€ Nom: Share-IJFT, Coût: 40.9€, Profit après 2 ans: 15.91€ Nom: Share-ANFX, Coût: 38.54€, Profit après 2 ans: 15.31€ Nom: Share-MALJ, Coût: 46.37€, Profit après 2 ans: 15.25€ Nom: Share-OPBR, Coût: 39.0€, Profit après 2 ans: 15.19€ Nom: Share-FWMV, Coût: 41.68€, Profit après 2 ans: 14.92€ Nom: Share-HATC, Coût: 43.45€, Profit après 2 ans: 14.83€ Nom: Share-XGNC, Coût: 41.86€, Profit après 2 ans: 14.71€ Nom: Share-XQII, Coût: 13.42€, Profit après 2 ans: 5.3€ Nom: Share-DYVD, Coût: 0.28€, Profit après 2 ans: 0.03€ Nom: Share-LKSD, Coût: 0.12€, Profit après 2 ans: 0.01€ totaux : achat 499.6 - profit 182.61 Le fichier initial contenait 541 actions, nous avons sélectionné 14 actions dans le fichier.

Nom: Share-KMTG, Coût: 23.21€, Valeur après 2 ans: 9.28€
Nom: Share-GHIZ, Coût: 28.00€, Valeur après 2 ans: 11.17€
Nom: Share-NHWA, Coût: 29.18€, Valeur après 2 ans: 11.60€
Nom: Share-UEZB, Coût: 24.87€, Valeur après 2 ans: 9.81€
Nom: Share-LPDM, Coût: 39.35€, Valeur après 2 ans: 15.63€
Nom: Share-MTLR, Coût: 16.48€, Valeur après 2 ans: 6.59€
Nom: Share-USSR, Coût: 25.62€, Valeur après 2 ans: 10.14€
Nom: Share-GTQK, Coût: 15.40€, Valeur après 2 ans: 6.15€
Nom: Share-FKJW, Coût: 21.08€, Valeur après 2 ans: 8.39€
Nom: Share-MLGM, Coût: 0.01€, Valeur après 2 ans: 0.00€
Nom: Share-QLMK, Coût: 17.38€, Valeur après 2 ans: 6.86€
Nom: Share-WPLI, Coût: 34.64€, Valeur après 2 ans: 13.82€
Nom: Share-LGWG, Coût: 31.41€, Valeur après 2 ans: 12.41€
Nom: Share-ZSDE, Coût: 15.11€, Valeur après 2 ans: 6.03€
Nom: Share-SKKC, Coût: 24.87€, Valeur après 2 ans: 9.82€
Nom: Share-QQTU, Coût: 33.19€, Valeur après 2 ans: 13.14€
Nom: Share-GIAJ, Coût: 10.75€, Valeur après 2 ans: 4.29€
Nom: Share-XJMO, Coût: 9.39€, Valeur après 2 ans: 3.75€
Nom: Share-LRBZ, Coût: 32.90€, Valeur après 2 ans: 13.14€
Nom: Share-KZBL, Coût: 28.99€, Valeur après 2 ans: 11.35€
Nom: Share-EMOV, Coût: 8.89€, Valeur après 2 ans: 3.51€
Nom: Share-IFCP, Coût: 29.23€, Valeur après 2 ans: 11.66€
Total d'achat: 499.95€, Valeur totale après 2 ans: 198.54€
Nombre d'actions sélectionnées: 22

Nom: Share-ECAQ, Coût: 31.66€, Valeur après 2 ans: 12.50€
Nom: Share-IXCI, Coût: 26.32€, Valeur après 2 ans: 10.37€
Nom: Share-FWBE, Coût: 18.30€, Valeur après 2 ans: 7.29€
Nom: Share-ZOFA, Coût: 25.32€, Valeur après 2 ans: 10.07€
Nom: Share-PLLK, Coût: 19.94€, Valeur après 2 ans: 7.96€
Nom: Share-LXZU, Coût: 4.24€, Valeur après 2 ans: 1.68€
Nom: Share-YFVZ, Coût: 22.55€, Valeur après 2 ans: 8.82€
Nom: Share-ANFX, Coût: 38.54€, Valeur après 2 ans: 15.31€
Nom: Share-PATS, Coût: 27.70€, Valeur après 2 ans: 11.07€
Nom: Share-SCWM, Coût: 6.42€, Valeur après 2 ans: 2.45€
Nom: Share-NDKR, Coût: 33.06€, Valeur après 2 ans: 13.19€
Nom: Share-ALIY, Coût: 29.08€, Valeur après 2 ans: 11.61€
Nom: Share-JWGF, Coût: 48.69€, Valeur après 2 ans: 19.44€
Nom: Share-JGTW, Coût: 35.29€, Valeur après 2 ans: 13.91€
Nom: Share-FAPS, Coût: 32.57€, Valeur après 2 ans: 12.88€
Nom: Share-VCAX, Coût: 27.42€, Valeur après 2 ans: 10.69€
Nom: Share-LFXB, Coût: 14.83€, Valeur après 2 ans: 5.90€
Nom: Share-DWSK, Coût: 29.49€, Valeur après 2 ans: 11.60€
Nom: Share-XQII, Coût: 13.42€, Valeur après 2 ans: 5.30€
Nom: Share-ROOM, Coût: 15.06€, Valeur après 2 ans: 5.91€
Total d'achat: 499.90€, Valeur totale après 2 ans: 197.96€
Nombre d'actions sélectionnées: 20

Analyse des résultats

Dataset 1 :

	Coût	Profit	Nombre d'actions	Profit en %
Sienna	498€76	196€61	1	39,42%
Tri et sélection	499€99	197€09	3	39,42%
Tableau dynamique	499€95	198€54	22	39,71%

Le tableau ci-dessus nous montre que les résultats obtenus avec le jeu d'actions numéro 1 sont sensiblement les mêmes. La différence se situe surtout au niveau du nombre d'action sélectionnées. En effet, alors que le tableau dynamique semble privilégier l'achat d'une multitude de petites actions, les deux autres méthodes semblent favoriser les grosses actions.

Dataset 2 :

	Coût	Profit	Nombre d'actions	Profit en %
Sienna	489€24	193€78	18	39,61%
Tri et sélection	499€60	182€61	14	36,55%
Tableau dynamique	499€90	197€96	20	39,60%

Le tableau ci-dessus nous montre que les résultats obtenus avec le jeu d'actions numéro 2 ont cette fois un écart plus important. Sienna et le tableau dynamique sélectionnent un nombre d'actions plus important offrant ainsi une rentabilité meilleure.

Conclusion :

La comparaison entre les résultats de Sienna et nos algorithmes est difficile ne sachant pas comment elle a réalisé sa sélection. On peut cependant constater que l'algorithme dynamique offre dans tous les cas un meilleur résultat. La méthode su tri et sélection n'est pas toujours la meilleure du fait même de sa conception, le fait de ne choisir que de gros rendements pourrait nous faire passer à côté de petites actions plus rentables situées plus bas dans la liste triée.

Ouverture et conclusion

Vers un Avenir Optimisé

Plusieurs pistes d’améliorations futures de nos algorithmes optimisés

Intégration et Tests Continus :
Backtesting : Utiliser l'algorithme sur des données historiques pour évaluer la performance sur le long terme.
A/B Testing : Comparer les résultats de l'algorithme avec les choix d'experts humains pour affiner les stratégies d'investissement.

Améliorations Techniques :
Apprentissage Automatique : Explorer l'intégration de techniques d'apprentissage automatique pour prédire les tendances des actions et affiner les sélections.
Optimisation Multi-objectifs : Adapter l'algorithme pour considérer d'autres objectifs, tels que le risque, en plus du profit.

Engagement Client :
Personnalisation : Adapter les sorties de l'algorithme aux préférences et aux objectifs spécifiques des clients.
Transparence : Améliorer la transparence en expliquant les recommandations de l'algorithme aux clients de manière compréhensible.

Conclusion :
L'algorithme actuel est un solide point de départ pour une optimisation continue, visant à améliorer non seulement la performance financière mais aussi la satisfaction et l'engagement des clients.