Analyse d'algorithmes d'optimisation

OC Projet 7 : Résolvez des problèmes en utilisant des algorithmes en Python

Table des matières :

Algorithme : Analyse de l'algorithme dit de "Force Brute"

Comment fonctionne t-il ?

- Il compose toutes les combinaisons possibles en fonction du nombre d'actions du Dataset
- Il calcul, pour chaque combinaison, son coût total et le compare au portefeuille du client (fixé à 500€) :
  - Si le coût est trop élevé, il supprime la combinaison
  - Si le coût est égal ou en dessous du montant du porte-feuilles du client, il l'ajoute dans une liste
- Il consulte la liste finale contenant les combinaisons valables et calcul, pour chacunes de celles-ci, le bénéfice après 2 ans.
- Il ne conserve que le meilleur bénéfice et l'affiche en console

Avantage(s) de cette méthode :

- **L'utilisateur a la garantie que l'algorithme va explorer toutes les combinaisons possibles**
- **L'utilisateur a la garantie de posséder, in fine, la combinaison la plus rémunératrice par rapport à son investissement**

Inconvénient(s) de cette méthode :

- **L'algorithme est lent (environ 12s de temps d'attente pour un dataset de 20 actions seulement)**
- **Suivant la taille du dataset, l'algorithme peut-être amené à crasher suite à un temps et un nombre de calcul trop long**

Algorithme : Processus de réflexion pour la solution optimisée

Comment devrait se comporter l'algorithme de manière idéal ?

- Il devrait pouvoir donner la meilleure combinaison possible, peut-importe le nombre d'actions référencé
- Il devrait pouvoir s'exécuter sans aucun crash
- Il devrait pouvoir s'exécuter instantanément (-de 1s dans le cahier des charges pour l'exemple)

De quoi est constituée une actions ?

- Une référence
- Un prix à l'achat
- Un bénéfice sur 2 ans exprimé en % en relation avec le prix d'achat => prix d'achat de l'action X (bénéfice(%) / 100)

Les actions les plus intéressantes sont donc logiquement celles qui ont le meilleur bénéfice sur 2 ans.

Algorithme : Description de l'algorithme "optimisé" (avantages / limites)

L'algorithme utilisé résout le problème dit du "rendu de monnaie". La problématique de ce problème est d'arriver à rendre la monnaie sur un certain montant de la manière la plus efficace possible mais en limitant le nombre de pièces et billets à fournir.

Ici le montant représente le portefeuille du client, la monnaie à rendre va représenter le bénéfice sur 2 ans que cela peut lui apporter, et les pièces et billets vont quant à elles représenter les actions de nos dataset.

Comment fonctionne t-il ?

- Il va d'abord extraire toutes les actions une par unes afin de calculer le bénéfice sur 2 ans maximum en euros (tout en triant les actions qui semblent incohérentes)
- Il va ensuite classer ses actions de la plus rentable à la moins rentable
- Il va ajouter une par une les actions qui peuvent "rentrer" dans le portefeuilles du client et additionner les bénéfices au fur et à mesure.

Algorithme : Description de l'algorithme "optimisé" (avantages / limites)

Avantages et Limites de cette méthode

Avantage(s) de cette méthode :

- **L'utilisateur a la garantie que l'algorithme va inclure les actions à fort rendement**
- **L'algorithme sera beaucoup plus rapide (- de 1s de temps d'exécution)**
- **Le bénéfice final peut s'avérer être plus intéressant pour le client (voir comparaison avec Sienna)**

Inconvénient(s) de cette méthode :

- **La combinaison proposée ne sera peut-être pas la plus optimale**
- **Suivant la taille du dataset, les résultats peuvent être assez différents d'une méthode à l'autre**

Algorithme : Comparaison de l'efficacité "Force Brute" VS "Optimisé"

En terme de bénéfices :

```
Le coût de cette combinaison est de 498.00€
Le bénéfice de cette combinaison est de 99.08€ sur 2 ans

real    0m11.364s
```

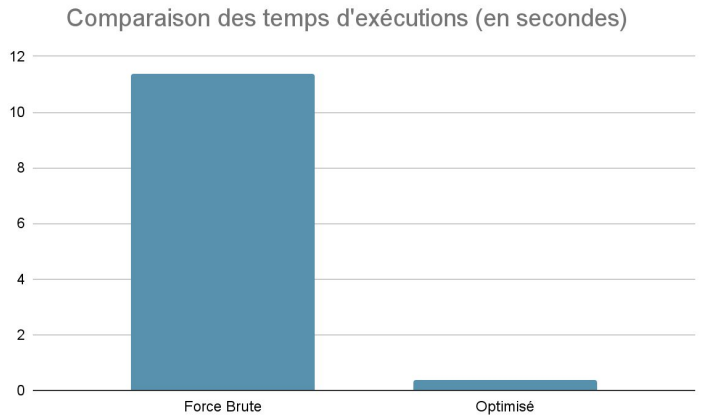
"Force Brute" : 498€ investi - 99,08€ bénéfice - soit un bénéfice de 19,89%

```
Le coût de cette combinaison est de 498.00€
Le bénéfice de cette combinaison est de 97.48€ sur 2 ans

real    0m0.394s
```

"Optimisé" : 498€ investi - 97,48€ bénéfice - soit un bénéfice de 19,57%

En terme de rapidité :



Algorithme : Comparaison de l'efficacité "Force Brute" VS "Optimisé"

Pourquoi une si grande différence de vitesse ?

1. La méthode employé pour l'algorithme de Force Brute demande à l'ordinateur d'effectuer un nombre d'opérations et de combinaisons très élevés, dans le cas du dataset de 20 actions, l'algorithme dénombre un total de 1 048 576 combinaisons dont 813 347 font moins de 500€ à l'achat (soit environ 77% de toutes les combinaisons).

   De plus l'ordinateur va d'abord créer les combinaisons (donc prendre beaucoup de temps pour cette action) puis va à nouveau parcourir une seconde liste pour effectuer les calculs et comparaisons ce qui va prendre encore beaucoup de temps.

   Ce temps sera d'autant plus important en fonction de la taille du dataset, puisque la vitesse d'exécution dépend directement du nombre d'actions à effectuer.

2. A contrario l'algorithme "Optimisé" ne va pas créer de combinaisons mais va simplement trier les actions de la plus rentable à la moins rentable en terme de bénéfices

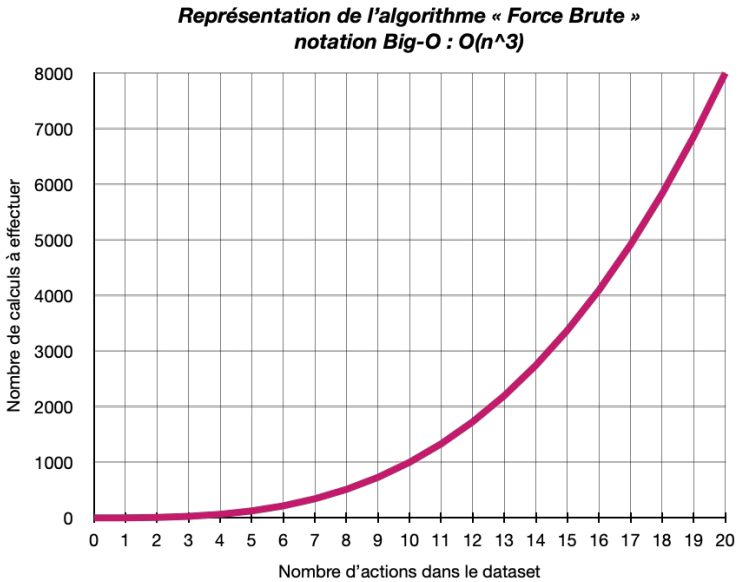
   Il va ensuite parcourir chacune de ses actions une par une et comparé son coût au portefeuille du client. Si elle peut être achetée alors le programme l'ajoute dans la liste finale

   Il y a donc beaucoup moins d'actions à faire avec cette méthode.

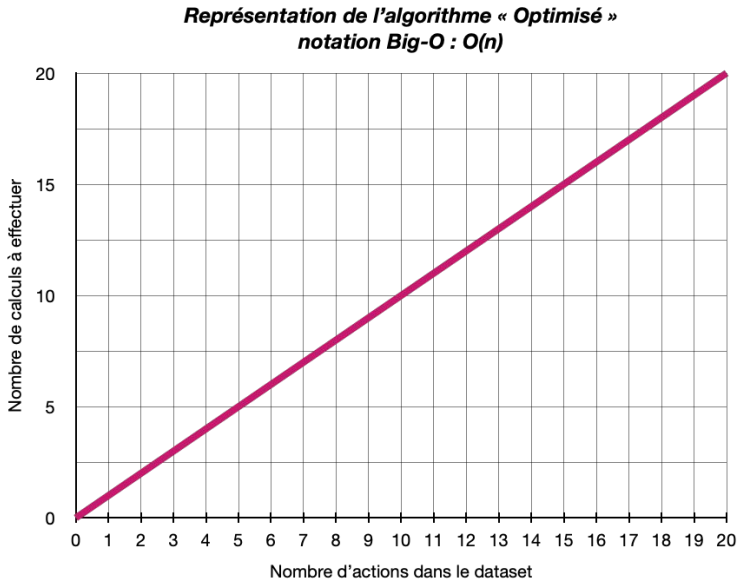
Algorithme : Comparaison de l'efficacité "Force Brute" VS "Optimisé"

Comparaison des performances (méthode de notation Big-O)

Algorithme de "Force Brute"

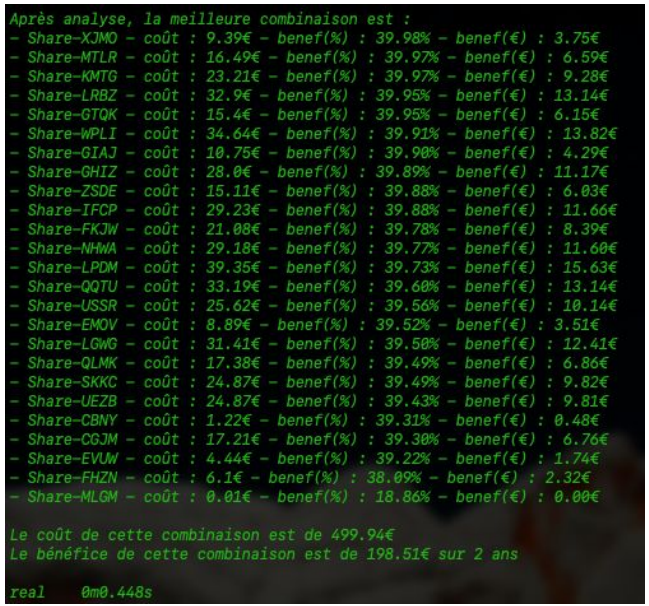


Algorithme "Optimisé"



Comparaison côte-à-côte avec les résultats de Sienna (Dataset 1 et 2)

Pour le dataset 1




```
Sienna bought:

Share-GRUT

Total cost: 498.76â,¬
Total return: 196.61â,¬
```

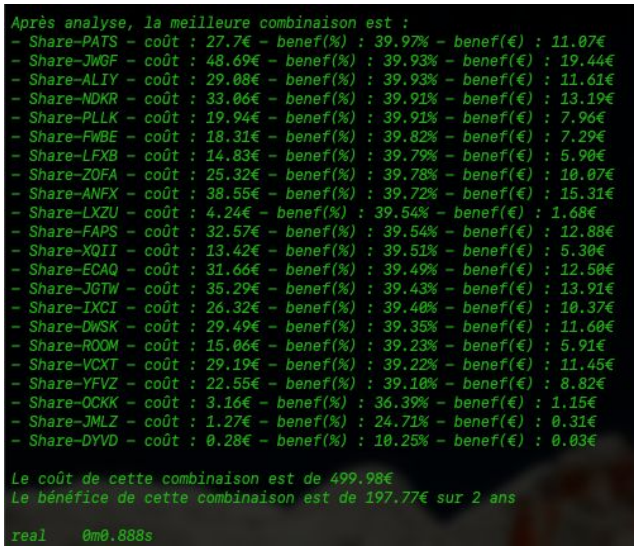
Les résultats ne correspondent visiblement pas avec l'algorithme de Sienna.
Cependant nous pouvons ici remarquer que le bénéfice sur 2 ans est plus important avec notre algorithme optimisé (+ 1,90€) mais que le coût de cette combinaison est aussi plus élevée (+1,18€).

La différence coût supplémentaire VS Bénéfice supplémentaire reste néanmoins meilleure avec l'algorithme optimisé (+0,72€)

Le rendement de notre combinaison est de 39,70% contre 39,41% concernant l'algorithme de Sienna

Comparaison côte-à-côte avec les résultats de Sienna (Dataset 1 et 2)

Pour le dataset 1




```
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,¬
```

Ici les résultat semblent plus proche entre les deux algorithmes (22 actions contre 18 pour Sienna) puisque plusieurs actions se retrouvent dans les deux combinaisons (ECAQ - IXCI - FWBE - NDKR ...)

Cependant nous pouvons ici remarquer que le bénéfice sur 2 ans est plus important avec notre algorithme optimisé (+ 4,00 €) mais que le coût de cette combinaison est aussi plus élevée (+10,74€).

La différence coût supplémentaire VS Bénéfice supplémentaire est clairement déficitaire (-6,74€ par rapport à la combinaison de Sienna)

Malgré tout nous constatons que le rendement de notre algorithme est de 39,55% contre 39,60% pour celui de Sienna ce qui ne représente pas une différence flagrante