

## Algorithme 'brute force' : résultat

action,	coût(€),	rendement(%)
Action-4,	70.0,	20.0
Action-5,	60.0,	17.0
Action-6,	80.0,	25.0
Action-8,	26.0,	11.0
Action-10,	34.0,	27.0
Action-11,	42.0,	17.0
Action-13,	38.0,	23.0
Action-18,	10.0,	14.0
Action-19,	24.0,	21.0
Action-20,	114.0,	18.0

Coût : 498.0 €

Profit : 99.08 €

Calcul : 0.8848457336425781 s

Mémoire: +32,768 octets

## Algorithme 'brute force' : analyse algorithme

Le principe est de générer toutes les combinaisons sans remise d'actions possible avec 1 seule action, 2 actions, ..., N actions.

Retenir la combinaison d'actions (portfolio) dont le coût ne dépassent pas 500 € et avec le meilleur rendement.

La nombre de combinaisons de 1 à N éléments est égal à  $2^N$ .  
Le calcul peut se comprendre en considérant chaque action comme un bit dans une suite de N bits. Il suffit de calculer le nombre de combinaison possibles de cette suite :  $2^N$ .

$$O(N)=2^N$$

Le temps de calcul est ~1s comprenant la lecture du fichier de données + recherche du portfolio le plus performant.

Le temps de calcul est exponentiel. L'algorithme est à éviter pour de grandes valeurs de N.

## Algorithme optimisé: algorithme choisi

Le nombre élevé d'échantillons est pénalisant dans l'algorithme de 'brute force' où toutes les combinaisons sont étudiées.

Un algorithme où chaque action est examinée à la suite et en calculant le profit en tenant compte du résultat des actions préalablement calculé apporte une meilleure performance.

L'algorithme retenu est connu sous le nom : 0-1 knapsack  
[Knapsack problem - Wikipedia](#)

Il s'appuie sur l'utilisation d'une matrice  $(M+1) * (N+1)$  où  $N$  représente le nombre d'actions à évaluer et  $M$  le budget à dépenser \* 100.

La valeur des actions étant un nombre décimal avec 2 chiffres après la décimale, pour indexer les éléments dans la matrice, ceux-ci sont multipliés par 100, idem pour le budget.

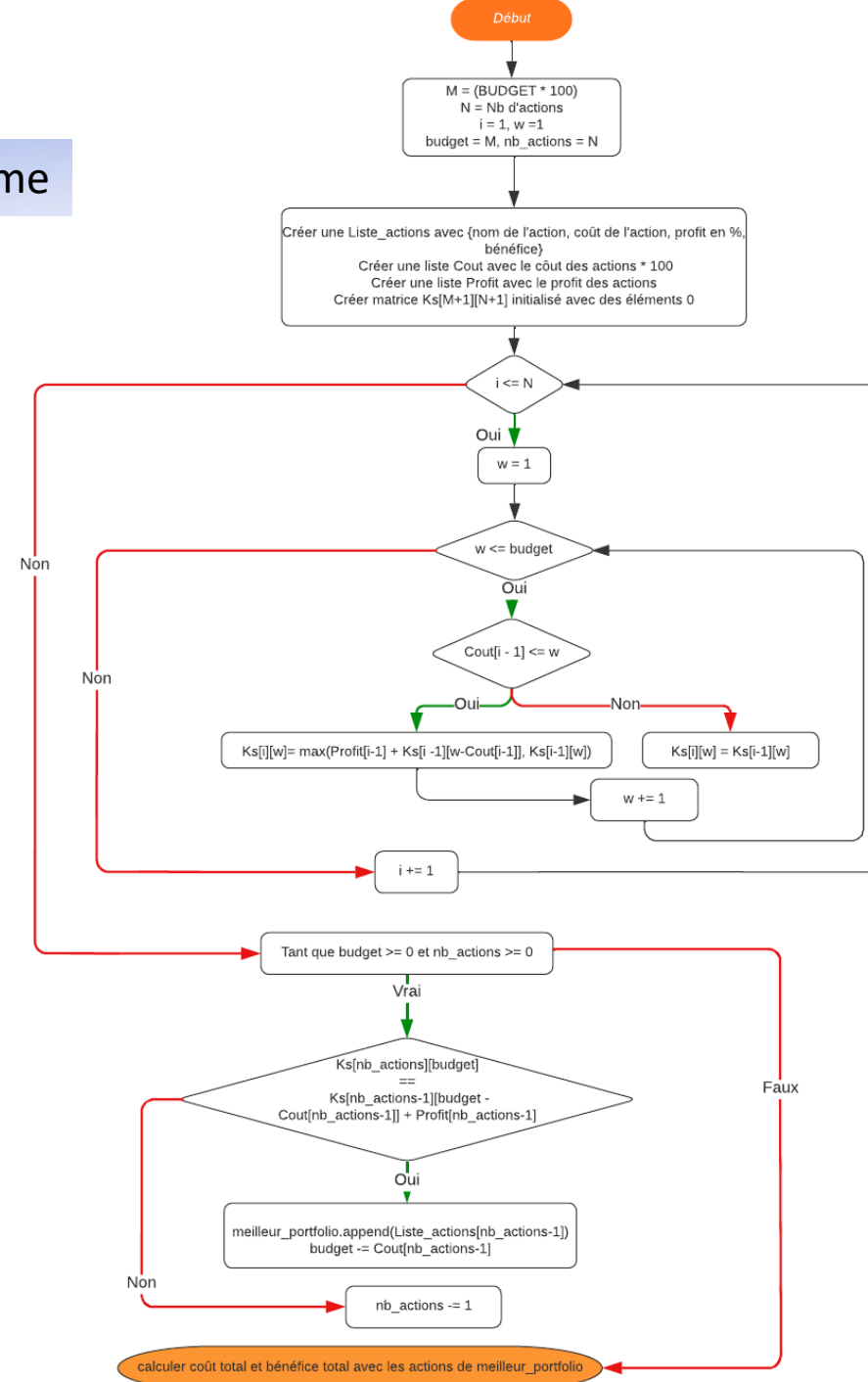
Ce facteur rajoute des traitements non utiles, une meilleure performance est possible si la valeur des actions est arrondie en entier.

## Algorithme optimisé: algorithme choisi

La ligne 0 et colonne 0 sert à initialiser le calcul.

L'algorithme pour calculer le meilleur revenu est en  $O(M*N)$ , cela correspond aux 2 boucles imbriquées dans le diagramme qui suit.

## Algorithme optimisé: diagramme



## Algorithme 'brute force' vs optimized : rapport d'exploration de l'ensemble des données

### brute force

action,	coût(€),	rendement(%)
Action-4,	70.0,	20.0
Action-5,	60.0,	17.0
Action-6,	80.0,	25.0
Action-8,	26.0,	11.0
Action-10,	34.0,	27.0
Action-11,	42.0,	17.0
Action-13,	38.0,	23.0
Action-18,	10.0,	14.0
Action-19,	24.0,	21.0
Action-20,	114.0,	18.0

Coût : 498.0 €

Profit : 99.08 €

Calcul : 0.8848457336425781 s

Mémoire: +32,768 octets

### optimized

action,	coût(€)	rendement(%)
Action-20,	114.0,	18.0
Action-19,	24.0,	21.0
Action-18,	10.0,	14.0
Action-13,	38.0,	23.0
Action-11,	42.0,	17.0
Action-10,	34.0,	27.0
Action-8,	26.0,	11.0
Action-6,	80.0,	25.0
Action-5,	60.0,	17.0
Action-4,	70.0,	20.0

Coût : 498.0 €

Profit : 99.08 €

Calcul : 0.23109769821166992 s

Mémoire: +1,269,760 octets

- Temps de calcul meilleur en optimized
  - Diff=0.65s
- Occupation de mémoire meilleur en brute force
  - Diff=1,236,992 octets

## Algorithme optimized - rapport d'exploration de l'ensemble des données : dataset1

action,	coût(€)	rendement(%)
Share-KMTG,	23.21,	39.97
Share-GHIZ,	28.0,	39.89
Share-NHWA,	29.18,	39.77
Share-UEZB,	24.87,	39.43
Share-LPDM,	39.35,	39.73
Share-MTLR,	16.48,	39.97
Share-USSR,	25.62,	39.56
Share-GTQK,	15.4,	39.95
Share-FKJW,	21.08,	39.78
Share-MLGM,	0.01,	18.86
Share-QLMK,	17.38,	39.49
Share-WPLI,	34.64,	39.91
Share-LGWG,	31.41,	39.5
Share-ZSDE,	15.11,	39.88
Share-SKKC,	24.87,	39.49
Share-QQTU,	33.19,	39.6
Share-GIAJ,	10.75,	39.9
Share-XJMO,	9.39,	39.98
Share-LRBZ,	32.9,	39.95
Share-KZBL,	28.99,	39.14
Share-EMOV,	8.89,	39.52
Share-IFCP,	29.23,	39.88

Coût : 499.95 €

Profit : 198.546521 €

Calcul : 13.948973655700684 s

Mémoire: +2,822,144 octets

Sienna bought:

Share-GRUT

Total cost: 498.76€,

Total return: 196.61€ ,

Meilleur résultat avec mon algo pour le coût et le profit.

L'algo de Sienna semble prendre les actions avec le le coût le plus élevé

## Algorithme optimized - rapport d'exploration de l'ensemble des données : dataset2

action,	coût(€)	rendement(%)
Share-ECAQ,	31.66,	39.49
Share-IXCI,	26.32,	39.4
Share-FWBE,	18.3,	39.82
Share-ZOFA,	25.32,	39.78
Share-PLLK,	19.94,	39.91
Share-LXZU,	4.24,	39.54
Share-YFVZ,	22.55,	39.1
Share-ANFX,	38.54,	39.72
Share-PATS,	27.7,	39.97
Share-SCWM,	6.42,	38.1
Share-NDKR,	33.06,	39.91
Share-ALIY,	29.08,	39.93
Share-JWGF,	48.69,	39.93
Share-JGTW,	35.29,	39.43
Share-FAPS,	32.57,	39.54
Share-VCAX,	27.42,	38.99
Share-LFXB,	14.83,	39.79
Share-DWSK,	29.49,	39.35
Share-XQII,	13.42,	39.51
Share-ROOM,	15.06,	39.23

Coût : 499.9 €

Profit : 197.96466400000003 €

Calcul : 8.64098334312439 s

Mémoire: +2,678,784 octets

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: 489.24€,-

Profit: 193.78€,-

Meilleur résultat avec mon algo pour le coût et le profit.

L'algo de Sienna semble ignorer les actions avec un coût < 10 €.

De plus le fichier de données ici(2) comporte des erreurs avec des actions de coût <= 0 €.