Cahier des charges : Simulation d'investissement en bourse

Table des matières

[Introduction 1](#_Toc145678205)

[Définition algorithme génétique 1](#_Toc145678206)

[1. Initialisation de la population : 1](#_Toc145678207)

[2. Évaluation de la population : 1](#_Toc145678208)

[3. Sélection des individus : 1](#_Toc145678209)

[4. Croisement (Crossover) : 2](#_Toc145678210)

[5. Mutation : 2](#_Toc145678211)

[6. Remplacement de la population : 2](#_Toc145678212)

[7. Critères d'arrêt : 2](#_Toc145678213)

[8. Renvoi du meilleur individu : 2](#_Toc145678214)

[Fonctions du programme 3](#_Toc145678215)

[Bloc Données Entrantes 3](#_Toc145678216)

[Bloc Algorithme genetique 3](#_Toc145678217)

[1. `simulate\_trading(actions, stop\_loss, take\_profit)` 3](#_Toc145678218)

[2. `evaluate\_individual(individual)` 3](#_Toc145678219)

[3. `create\_population(population\_size)` 3](#_Toc145678220)

[4. `select\_best\_individuals(population, num\_best)` 4](#_Toc145678221)

[5. `crossover(parent1, parent2)` 4](#_Toc145678222)

[6. `mutate(individual, mutation\_rate)` 4](#_Toc145678223)

[Bloc Simulation 5](#_Toc145678224)

[Bloc Affichage 5](#_Toc145678225)

[Utilisation du programme 6](#_Toc145678226)

# 

# Introduction

L’objectif du logiciel est de permettre d’analyser les courbes des actions en bourses tel que les cryptomonnaies, les matieres premieres ou encore les sociétés pour determiner quelle est la manière la plus adequate d’investir. Pour ce faire, nous allons mettre en place un algorithme génétique qui permettra de determiner les bons parametres à utiliser pour l’achat et la revente.

# Définition algorithme génétique

Un algorithme génétique est une technique d'optimisation inspirée du processus de sélection naturelle observé dans l'évolution des espèces. Il est souvent utilisé pour résoudre des problèmes d'optimisation et de recherche, où l'objectif est de trouver la meilleure solution possible parmi un ensemble de solutions candidates.

L'algorithme génétique suit plusieurs étapes clés pour évoluer vers une solution optimale :

## 1. Initialisation de la population :

- Création d'une population initiale d'individus, chaque individu représentant une solution potentielle du problème.

- Les individus sont souvent générés de manière aléatoire ou avec des règles spécifiques.

## 2. Évaluation de la population :

- Chaque individu est évalué en fonction de sa performance par rapport à l'objectif à atteindre (la fonction d'évaluation).

- Cette étape attribue une mesure de "fitness" à chaque individu, indiquant à quel point il est adapté à la résolution du problème.

## 3. Sélection des individus :

- Les individus sont sélectionnés pour la reproduction en fonction de leur performance (fitness).

- Les individus les plus performants ont plus de chances d'être sélectionnés, mais la sélection est généralement aléatoire pour éviter de rester bloqué dans un optimum local.

## 4. Croisement (Crossover) :

- Paires d'individus sélectionnés sont combinées pour créer de nouveaux individus (descendants) en échangeant une partie de leurs caractéristiques (gènes).

- Le croisement permet de créer une diversité génétique et d'explorer l'espace des solutions.

## 5. Mutation :

- Certains individus subissent des modifications aléatoires de leurs caractéristiques (gènes) pour introduire de la diversité dans la population.

- La mutation permet d'explorer de nouvelles solutions qui ne sont pas nécessairement dérivées de la reproduction.

## 6. Remplacement de la population :

- Les individus créés par le croisement et la mutation remplacent une partie de la population précédente.

- Certains individus peuvent être conservés d'une génération à l'autre pour maintenir une certaine stabilité.

## 7. Critères d'arrêt :

- L'algorithme génétique continue d'évoluer les générations jusqu'à ce qu'un critère d'arrêt soit atteint, comme un nombre maximal de générations ou un niveau de performance souhaité.

## 8. Renvoi du meilleur individu :

- Une fois l'algorithme terminé, le meilleur individu de la dernière génération est renvoyé comme la meilleure solution trouvée.

En répétant ces étapes de manière itérative, l'algorithme génétique converge vers des solutions de plus en plus performantes. L'approche s'inspire de l'évolution biologique, où les individus les mieux adaptés ont plus de chances de survivre et de transmettre leurs caractéristiques à la génération suivante.

# 

# Fonctions du programme

## Bloc Données Entrantes

## Bloc Algorithme genetique

### 1. `simulate\_trading(actions, stop\_loss, take\_profit)`

Cette fonction simule le processus d'achat et de revente d'actions en bourse en utilisant les paramètres StopLoss et TakeProfit.

Paramètres

- `actions` : Une liste d'entiers représentant les valeurs des actions à chaque période.

- `stop\_loss` : Le pourcentage de perte maximale avant de vendre les actions.

- `take\_profit` : Le pourcentage de gain avant de vendre les actions.

Résultat

La fonction renvoie le solde final après avoir effectué les transactions.

### 2. `evaluate\_individual(individual)`

Cette fonction évalue la performance d'un individu dans le contexte d'un algorithme génétique. L'individu est caractérisé par les paramètres StopLoss et TakeProfit.

Paramètres

- `individual` : Un tuple représentant les paramètres StopLoss et TakeProfit.

Résultat

La fonction renvoie une mesure de performance basée sur la simulation d'investissement.

### 3. `create\_population(population\_size)`

Cette fonction crée une population initiale d'individus avec des paramètres StopLoss et TakeProfit aléatoires.

Paramètres

- `population\_size` : La taille de la population.

Résultat

La fonction renvoie une liste d'individus.

### 4. `select\_best\_individuals(population, num\_best)`

Cette fonction sélectionne les meilleurs individus d'une population en fonction de leur performance.

Paramètres

- `population` : La liste des individus à évaluer.

- `num\_best` : Le nombre d'individus à sélectionner comme les meilleurs.

Résultat

La fonction renvoie une liste des meilleurs individus.

### 5. `crossover(parent1, parent2)`

Cette fonction effectue un croisement (crossover) entre deux individus pour créer deux descendants.

Paramètres

- `parent1` : Le premier parent (tuple représentant les paramètres StopLoss et TakeProfit).

- `parent2` : Le deuxième parent (tuple représentant les paramètres StopLoss et TakeProfit).

Résultat

La fonction renvoie deux descendants obtenus par croisement.

### 6. `mutate(individual, mutation\_rate)`

Cette fonction effectue une mutation sur un individu en modifiant légèrement ses paramètres StopLoss et TakeProfit.

Paramètres

- `individual` : L'individu à muter (tuple représentant les paramètres StopLoss et TakeProfit).

- `mutation\_rate` : Le taux de mutation, représentant la probabilité qu'un paramètre soit modifié.

Résultat

La fonction renvoie l'individu muté.

## Bloc Simulation

## Bloc Affichage

# Utilisation du programme

Pour utiliser le programme de simulation d'investissement en bourse, suivez ces étapes :

1. Importez le module contenant les fonctions ou copiez le code dans votre fichier Python.

2. Utilisez la fonction `simulate\_trading` en fournissant une liste d'actions, le pourcentage de perte maximale (StopLoss) et le pourcentage de gain avant revente (TakeProfit) pour simuler les transactions.

Exemple d'utilisation :

```python

actions = [3, 4, 5, 4, 3, 2, 1, 2, 3, 2, 1, 2, 3, 4, 3]

stop\_loss = 0.1 10% de perte maximale

take\_profit = 0.15 15% de profit avant revente

solde\_final = simulate\_trading(actions, stop\_loss, take\_profit)

print(f"Solde final : {solde\_final}")

```

3. Utilisez les autres fonctions du programme pour implémenter des algorithmes génétiques permettant d'optimiser les paramètres StopLoss et TakeProfit pour des simulations d'investissement plus avancées.

Conclusion

Ce cahier des charges définit les objectifs et les spécifications du projet de simulation d'investissement en bourse. Il fournit des indications sur l'utilisation des différentes fonctions du programme pour atteindre ces objectifs.