REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Larbi Ben M'Hidi Oum El Bouaghi

Cours Méthodes algorithmiques

1ere année master AD

Chapitre 5 : Les méthodes probabilistes

Dr.Belhouchette

01

INTRODUCTION

Introduction aux Méthodes Probabilistes

Les méthodes probabilistes sont des approches algorithmiques qui utilisent des techniques basées sur la probabilité pour résoudre des problèmes.

Ces méthodes exploitent l'incertitude ou les comportements aléatoires pour :

- **Simplifier les calculs complexes**
- **Optimiser les performances des algorithmes**
- **Résoudre des problèmes où les solutions déterministes sont inefficaces ou inexistantes**

02

Définitions

Définition des méthodes Probabilistes

Les méthodes probabilistes sont une classe d'approches algorithmique et analytique où la probabilité est utilisée pour résoudre des problèmes ou pour analyser des algorithmes.

Contrairement aux algorithmes déterministes, qui suivent une séquence fixe d'opérations, les algorithmes probabilistes exploitent des choix aléatoires dans leur exécution.

Définition des méthodes Probabilistes

Ils se repose principalement sur :

1. Utilisation de la **randomisation** comme outil de conception.
2. Modélisation de phénomènes complexes ou incertains par des variables aléatoires.
3. Analyse basée sur les notions d’espérance mathématique et de probabilité.

03

Contexte d’Utilisation des Méthodes Probabilistes

Contexte d'Utilisation des Méthodes Probabilistes

- Problème du voyageur de commerce.
- Problème de la couverture minimale.

Optimisation combinatoire

- Traitement de données incertaines.
- Analyse bayésienne pour gérer l'incertitude.

Apprentissage Automatique et IA

- Génération de nombres aléatoires pour :
  - Produire des clés cryptographiques.
  - Évaluer les risques d'attaques.
- Conception de systèmes sécurisés basés sur la probabilité.

Cryptographie et Sécurité

- Analyse de graphes complexes (ex. réseaux sociaux).
- Algorithmes comme les marches aléatoires pour explorer les relations et structures.

Traitement de Graphes

- Etude de systèmes complexes :
  - Marchés financiers.
  - Propagation des épidémies.
  - Systèmes physiques et biologiques.
- Utilisation pour prédire des comportements et tester des scénarios.

Simulation et Modélisation

04

Différence entre Algorithmes Déterministes et Probabilistes

Différence entre Algorithmes Déterministes et Probabilistes

Définition

Algorithmes Déterministes :
Un algorithme déterministe produit toujours le même résultat pour une entrée donnée. Toutes les étapes de l'algorithme sont définies et prévisibles.

Algorithmes Probabilistes :
Un algorithme probabiliste intègre des choix aléatoires dans son exécution. Il peut produire des résultats différents pour une même entrée, et son comportement repose en partie sur des probabilités.

Différence entre Algorithmes Déterministes et Probabilistes

Critères de comparaison	Déterministes	Probabilistes
Nature des Résultats	Les résultats sont toujours identiques, précis, et entièrement reproductibles.	Les résultats peuvent varier d'une exécution à une autre. Certains algorithmes garantissent une solution correcte (Las Vegas), tandis que d'autres fournissent une solution approximative avec un niveau de confiance (Monte Carlo).
Complexité et Efficacité	Les performances dépendent uniquement des étapes préalablement définies.	complexité dépend souvent des probabilités d'atteindre une solution satisfaisante.

Différence entre Algorithmes Déterministes et Probabilistes

Critères de comparaison	Déterministes	Probabilistes
Contexte d'Utilisation	Idéal pour les problèmes où une solution exacte est obligatoire. Utilisés dans les systèmes critiques, comme la finance ou la médecine, où l'aléatoire est inacceptable.	Préférés pour les problèmes complexes ou les environnements incertains. Utilisés dans des domaines comme l'optimisation, la cryptographie, ou l'intelligence artificielle.

Différence entre Algorithmes Déterministes et Probabilistes

Avantages et Inconvénients	Déterministes	Probabilistes
Précision	Résultats exacts et reproductibles.	Résultats parfois approximatifs ou probabilistes.
Efficacité	Moins adaptés aux problèmes complexes.	Rapides pour les grands problèmes complexes.
Prévisibilité	Comportement totalement prévisible.	Comportement dépendant du hasard.

05

Types d'Algorithmes Probabilistes

Types d'Algorithmes Probabilistes

Les algorithmes probabilistes se classent en plusieurs types selon leur utilisation de la randomisation et leur garantie sur les résultats.

01	• Algorithmes de Monte Carlo
02	• Algorithmes de Las Vegas
03	• Algorithmes Hybrides
04	• Algorithmes Stochastiques
05	• Algorithmes Quasi-Randomisés

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Définition et Principe Fondamental

Les algorithmes de Monte Carlo consistent à exécuter un processus aléatoire un grand nombre de fois pour estimer une valeur ou obtenir une solution.

L'idée repose sur la **loi des grands nombres**, selon laquelle les moyennes des résultats obtenus à partir de répétitions indépendantes convergent vers la valeur réelle.

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Étapes Générales d'un Algorithme de Monte Carlo

1. Modélisation du problème : Représenter le problème sous forme d'un processus probabiliste.

2. Génération de variables aléatoires : Produire des échantillons aléatoires conformes à la distribution choisie.

3. Évaluation d'une fonction : Appliquer une fonction ou un test sur chaque échantillon.

4. Agrégation des résultats : Moyenne, somme ou autre opération statistique pour estimer le résultat final.

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Propriétés Mathématiques

Convergence : La précision augmente proportionnellement à $\frac{1}{\sqrt{n}}$, où n est le nombre de simulations. Cela implique qu'un grand nombre de simulations est souvent nécessaire pour atteindre une précision élevée.

Invariance dimensionnelle : Contrairement aux méthodes déterministes, l'efficacité de Monte Carlo est peu affectée par le nombre de dimensions du problème.

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas Calculer la probabilité d'obtenir pile en lançant une pièce

- L'idée est de simuler plusieurs lancers d'une pièce et d'estimer la probabilité d'obtenir **pile** en comptant les occurrences.
- La probabilité réelle est P(pile)=0.5, mais l'estimation varie en fonction du nombre de lancers.

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas Calculer la probabilité d'obtenir pile en lançant une pièce

```
import random

def lancer_piece(n):
    pile = 0
    for _ in range(n):
        r = random.random() -> Génère un nombre aléatoire entre 0 et 1
        if r < 0.5:
pile += 1
    return pile / n
```

Jeux de calcul
Posons n = 1000 Nombre de lancers
probabilite_pile = lancer_piece(n)
print(f"Probabilité estimée d'obtenir pile avec {n} lancers : {probabilite_pile}")

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas pour vérifier un produit de matrices

Étant donné deux matrices A et B, et un produit attendu C, l'objectif est de vérifier si C est bien le produit de A et B sans calculer entièrement C de manière déterministe, mais en utilisant un algorithme probabiliste de type *Monte Carlo*.

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas pour vérifier un produit de matrices
Etapes :

1. **Calcul de $Y=B\times X$** : On commence par calculer le produit de la matrice B avec un vecteur X aléatoire.
2. **Calcul de $Z=A\times Y$** :
3. **Vérification $Z=C\times X$** : Si $Z=C\times X$, alors vous pouvez conclure que le produit matriciel C est correct.

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas pour vérifier un produit de matrices

$$\begin{pmatrix} 2 & -1 & 1 & 2 \\ -1 & 0 & 3 & 1 \\ 2 & -3 & -2 & 1 \\ 1 & 2 & 2 & 1 \end{pmatrix} \begin{pmatrix} 3 & 1 & -1 & 1 \\ 2 & 4 & 1 & -2 \\ -2 & 1 & 1 & 1 \\ 2 & 1 & 2 & -1 \end{pmatrix} \begin{pmatrix} 6 & 1 & 2 & 3 \\ -7 & 3 & 6 & 1 \\ 6 & -11 & -5 & 5 \\ 5 & 11 & 5 & -2 \end{pmatrix}$$

A

B

C

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas pour vérifier un produit de matrices

$$\begin{pmatrix} 2 & -1 & 1 & 2 \\ -1 & 0 & 3 & 1 \\ 2 & -3 & -2 & 1 \\ 1 & 2 & 2 & 1 \end{pmatrix} \begin{pmatrix} 3 & 1 & -1 & 1 \\ 2 & 4 & 1 & -2 \\ -2 & 1 & 1 & 1 \\ 2 & 1 & 2 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 6 & 1 & 2 & 3 \\ -7 & 3 & 6 & 1 \\ 6 & -11 & -5 & 5 \\ 5 & 11 & 5 & -2 \end{pmatrix}$$

A

B

X

C

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas pour vérifier un produit de matrices

$$\begin{pmatrix} 3 & 1 & -1 & 1 \\ 2 & 4 & 1 & -2 \\ -2 & 1 & 1 & 1 \\ 2 & 1 & 2 & -1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \\ 2 \\ 3 \end{pmatrix}$$

B

X

Y

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas pour vérifier un produit de matrices

$$\begin{pmatrix} 2 & -1 & 1 & 2 \\ -1 & 0 & 3 & 1 \\ 2 & -3 & -2 & 1 \\ 1 & 2 & 2 & 1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 5 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 3 \\ 9 \\ -16 \\ 17 \end{pmatrix}$$

A Y Z

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas pour vérifier un produit de matrices

$$\begin{pmatrix} 6 & 1 & 2 & 3 \\ -7 & 3 & 6 & 1 \\ 6 & -11 & -5 & 5 \\ 5 & 11 & 5 & -2 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 9 \\ -16 \\ 16 \end{pmatrix}$$

C

X

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas pour vérifier un produit de matrices

$$\begin{pmatrix} 3 \\ 9 \\ -16 \\ 17 \end{pmatrix}$$

Z

EST DIFFERENT

$$\begin{pmatrix} 3 \\ 9 \\ -16 \\ 16 \end{pmatrix}$$

C *X

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas pour vérifier un produit de matrices

```
import random

def verifie_multiplication(A, B, C):
n = len(A)       # Nombre de lignes de A
m = len(B[0])    # Nombre de colonnes de B
 p = len(B)       # Nombre de lignes de B (égal au nombre de colonnes de A)
    q = len(C)       # Nombre de lignes de C (devrait être égal à n)
    r = len(C[0])    # Nombre de colonnes de C (devrait être égal à m)

essais = 100
```

Types d'Algorithmes Probabilistes

Algorithmes de Monte Carlo

Etude de cas pour vérifier un produit de matrices

```
  for _ in range(essais):
X = [random.randint(1, 10) for _ in range(m)]

Y = [sum(B[i][k] * X[k] for k in range(p)) for i in range(n)]

Z = [sum(A[i][k] * Y[k] for k in range(m)) for i in range(n)]

CX = [sum(C[i][k] * X[k] for k in range(r)) for i in range(n)]
if Z != CX:
            return False

return True
```

Types d'Algorithmes Probabilistes

Algorithmes de Las Vegas

Définition

Un **algorithme de Las Vegas** est un algorithme probabiliste dans lequel le **temps d'exécution** peut varier, mais qui **produit toujours un résultat correct**.

Contrairement aux **algorithmes de Monte Carlo**, où la précision du résultat peut être influencée par la probabilité, un algorithme de Las Vegas garantit une solution correcte, mais son coût de computation peut fluctuer en fonction des tirages aléatoires.

Types d'Algorithmes Probabilistes

Algorithmes de Las Vegas

Caractéristiques principales

1. **Résultats Corrects** : Quel que soit le tirage aléatoire, le résultat renvoyé par l'algorithme est toujours correct.

2. **Temps d'exécution aléatoire** : Le temps nécessaire pour obtenir le résultat dépend des tirages effectués. En général, plus le nombre de tentatives (tirages) est élevé, plus le temps d'exécution sera long.

3. **Probabilité d'échec** : Contrairement aux algorithmes de Monte Carlo, il n'y a pas de risque d'erreur dans le résultat, mais le temps nécessaire pour obtenir le bon résultat peut être variable.

Types d'Algorithmes Probabilistes

Algorithmes de Las Vegas

Contexte d'Utilisation

Les algorithmes de Las Vegas sont souvent utilisés dans des **problèmes d'optimisation combinatoire**, où la recherche d'une solution optimale ou correcte peut être améliorée par un tirage aléatoire, mais où une solution erronée n'est pas tolérée.

Ces algorithmes sont utilisés dans des cas où :

- La garantie d'une solution correcte est cruciale.
- Le temps de calcul variable est acceptable en fonction des exigences de performance.

Types d'Algorithmes Probabilistes

Algorithmes de Las Vegas

```
import random
def recherche(T, r):
    if len(T) == 1:
        return T[0]
i = random.randint(0, len(T)-1)
    pivot = T[i]

    petit = []
    grand = []

for x in T:
        if x <= pivot:
            petit.append(x)
        else:
            grand.append(x)

    L = len(petit)

    if L == r:
return pivot
    elif L > r:
return recherche(petit, r)
    else: return recherche(grand, r - L)

def median(T):
    return recherche(T, len(T) // 2)
```

Etude de cas **Quick Select un exemple de type Las Vegas pour trouver le média**

Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Définition

- Les **algorithmes hybrides** sont des algorithmes qui combinent deux ou plusieurs techniques ou paradigmes algorithmique afin de tirer parti des avantages de chacune d'elles pour résoudre un problème plus efficacement.

- Ces algorithmes exploitent des approches complémentaires, souvent en fonction des caractéristiques spécifiques du problème à résoudre.

- L'objectif est généralement de combiner la **robustesse** et l'**efficacité** pour optimiser le temps de calcul et la qualité des résultats.

Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Caractéristiques Principales

1. **Combinaison de différentes approches** : Les algorithmes hybrides mélangent des techniques comme les **algorithmes exacts**, **approximatifs**, **probabilistes** et **heuristiques**.

2. **Optimisation des performances** : Ils sont souvent utilisés pour améliorer les performances dans des cas où une approche purement exacte ou approximative pourrait être moins efficace.

3. **Flexibilité** : L'algorithme peut être ajusté pour exploiter les meilleures caractéristiques de plusieurs techniques en fonction du problème à résoudre.

Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Contexte d'Utilisation

Les algorithmes hybrides sont particulièrement utiles dans des situations où un seul type d'algorithme (par exemple, exact, heuristique ou probabiliste) n'est pas suffisant pour offrir un bon compromis entre **temps d'exécution** et **qualité de la solution**.

Ils sont largement utilisés dans des domaines tels que :

- ❑ **Optimisation combinatoire**
- ❑ **Recherche opérationnelle**
- ❑ **Apprentissage automatique**
- ❑ **Calcul scientifique**

Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Exemple de Technique Hybride : Algorithme Génétique avec Recherche Locale

Un exemple classique d'algorithme hybride est la combinaison d'un **algorithme génétique** avec une **méthode de recherche locale** (comme la descente de gradient ou l'algorithme de voisinage).

L'algorithme génétique est utilisé pour explorer un espace de solutions large de manière probabiliste, tandis que la recherche locale affine la solution en optimisant localement les résultats.

Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Exemple de Technique Hybride : Algorithme Génétique avec Recherche Locale

Principe :

1. L'algorithme génétique génère une population initiale de solutions possibles à un problème d'optimisation.

2. Ensuite, des techniques de recherche locale, comme la descente de gradient ou une recherche par voisinage, sont utilisées pour améliorer chaque solution.

3. Ce processus continue jusqu'à ce que la convergence vers une solution optimale soit atteinte.

Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Algorithme Génétique avec Recherche Locale

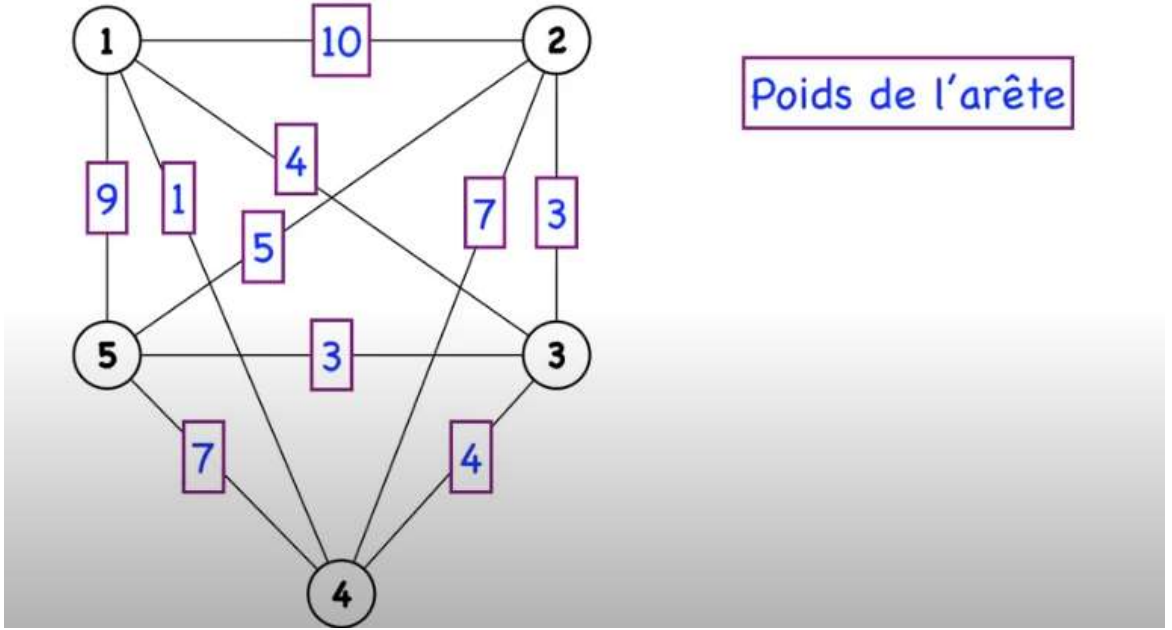
Input : Nombre de solutions à générer, nombre de générations, seuil de convergence
Output : Meilleure solution trouvée

1. Initialiser une population aléatoire de solutions
2. Pour chaque génération :
   a. Évaluer la fitness de chaque solution dans la population
   b. Sélectionner les solutions les mieux adaptées pour la reproduction
   c. Appliquer des opérations génétiques (croisement, mutation)
   d. Appliquer une recherche locale sur certaines solutions sélectionnées pour améliorer les résultats
   e. Remplacer la population avec la nouvelle génération
3. Retourner la meilleure solution trouvée

Types d'Algorithmes Probabilistes

Algorithmes Hybrides

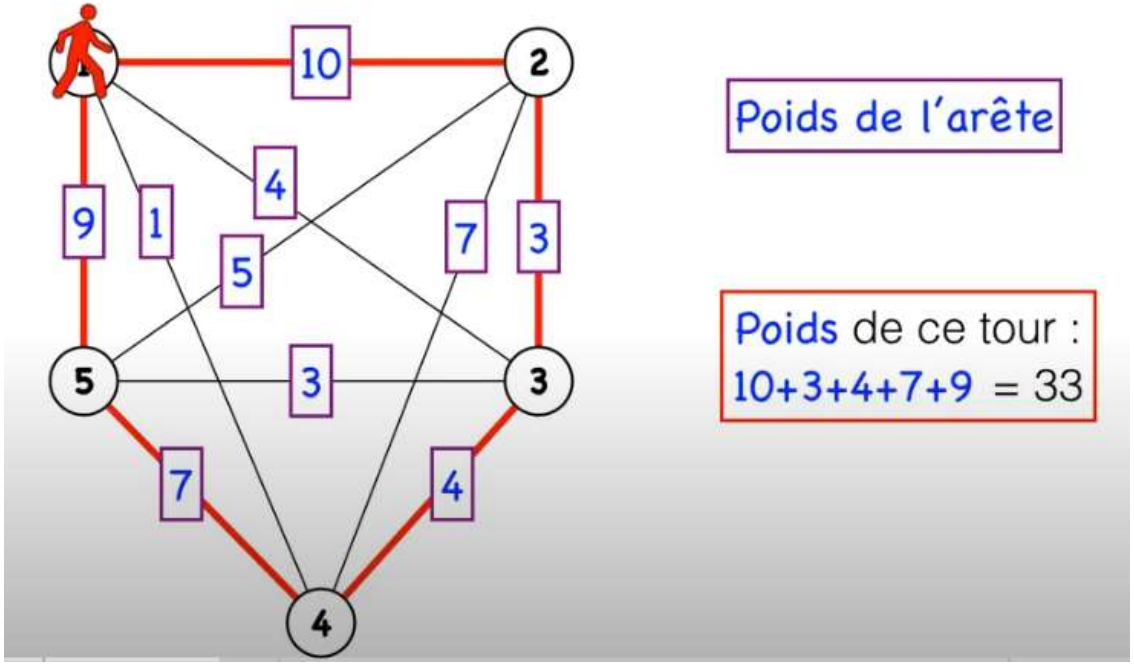
Exemple d'Application : Problème du Voyageur de Commerce (TSP)



Types d'Algorithmes Probabilistes

Algorithmes Hybrides

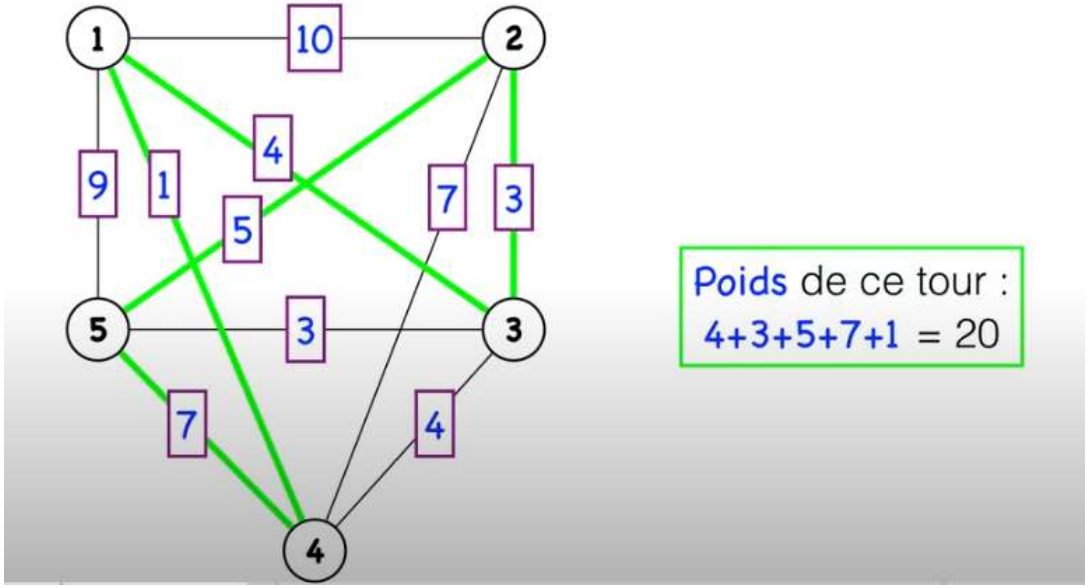
Exemple d'Application : Problème du Voyageur de Commerce (TSP)



Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Exemple d'Application : Problème du Voyageur de Commerce (TSP)



Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Exemple d'Application : Problème du Voyageur de Commerce (TSP)



Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Exemple d'Application : Problème du Voyageur de Commerce (TSP)



Construire un tour de poids minimal

Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Exemple d'Application : Problème du Voyageur de Commerce (TSP)

Étapes pour résoudre le TSP avec une approche hybride

1. Initialisation : Créer aléatoirement des chemins (solutions possibles) entre les villes.

2. Amélioration locale : Pour chaque chemin, essayer de trouver un chemin légèrement meilleur (par exemple, en échangeant deux villes).

3. Répéter plusieurs fois : Répéter les étapes ci-dessus plusieurs fois pour trouver un bon chemin.

Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Exemple d'Application : Problème du Voyageur de Commerce (TSP)

1. Générer un chemin initial (ordre aléatoire des villes).
2. Calculer la distance totale pour ce chemin.
3. Répéter pendant un certain nombre d'itérations :
    a. Faire un petit changement au chemin (exemple : inverser deux villes).
    b. Calculer la distance pour le nouveau chemin.
    c. Si le nouveau chemin est meilleur, le garder.
4. Retourner le meilleur chemin trouvé.

L'algorithme ne garantit pas la solution optimale, mais il trouvera un chemin relativement court, ce qui est suffisant pour de nombreux cas.

Types d'Algorithmes Probabilistes

Algorithmes Hybrides

Autres Exemples d'Algorithmes Hybrides

1. Algorithmes de recherche locale combinés avec des algorithmes génétiques
2. Algorithmes de Monte Carlo et heuristiques
3. Algorithmes de recherche tabou avec des stratégies gloutonnes

Types d'Algorithmes Probabilistes

Algorithmes Stochastiques

Définition :

- Les algorithmes stochastiques utilisent des éléments de probabilité ou des processus aléatoires dans leur fonctionnement.
- Contrairement aux algorithmes déterministes (résultats identiques à chaque exécution), les algorithmes stochastiques peuvent donner des résultats différents, même pour la même entrée, à cause des tirages aléatoires.
- Utilisés dans des situations où une approche exacte est trop complexe ou coûteuse, et où une solution approximative mais fiable est suffisante

Types d'Algorithmes Probabilistes

Algorithmes Stochastiques

Caractéristiques Principales

Processus aléatoire	• Ils utilisent des tirages aléatoires (probabilistes) à chaque étape de l'algorithme.
Résultats non déterministes	• En raison des tirages aléatoires, les résultats peuvent varier d'une exécution à l'autre.
Convergence probable	• Bien que les résultats varient, un algorithme stochastique tend à converger vers une solution optimale ou proche de l'optimum au fil des itérations.
Efficacité pour de grands espaces de recherche	• Ils sont particulièrement utiles pour explorer de grands espaces de solutions où une approche exhaustive serait inefficace

Types d'Algorithmes Probabilistes

Algorithmes Quasi-Randomisés

Définition:

- Les algorithmes quasi-randomisés utilisent une combinaison de techniques **déterministes** et **aléatoires** pour améliorer l'efficacité des algorithmes classiques.

- Ils génèrent des **séquences quasi-aléatoires**, qui sont des suites déterministes ayant des propriétés similaires à celles des suites aléatoires.

- Ces séquences assurent une bonne couverture de l'espace de recherche, tout en limitant la variabilité des résultats par rapport aux méthodes purement aléatoires.

Types d'Algorithmes Probabilistes

Algorithmes Quasi-Randomisés

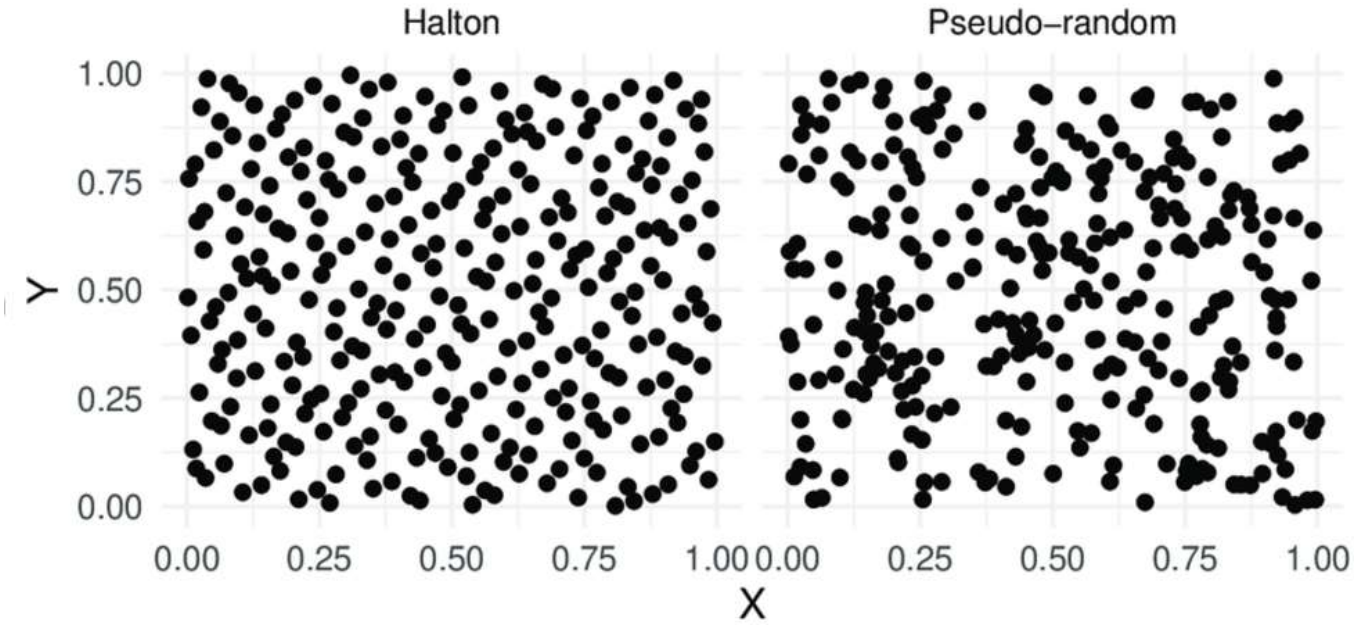
Caractéristiques Principales

Séquences Quasi-Aléatoires	• Contrairement aux méthodes purement aléatoires, les points ne sont pas générés au hasard, mais suivent des séquences comme celles de Sobol, Halton ou Faure.
Équilibre entre déterminisme et aléatoire	• Combinent déterminisme et aléa, offrant un compromis entre prévisibilité et exploration de l'espace de recherche.
Exploration efficace de l'espace	• Garantissent une couverture uniforme de l'espace, idéale pour l'optimisation et la simulation nécessitant une couverture complète et non biaisée.
Convergence rapide	• Assurent une couverture uniforme de l'espace, idéale pour l'optimisation et la simulation.

Types d'Algorithmes Probabilistes

Algorithmes Quasi-Randomisés

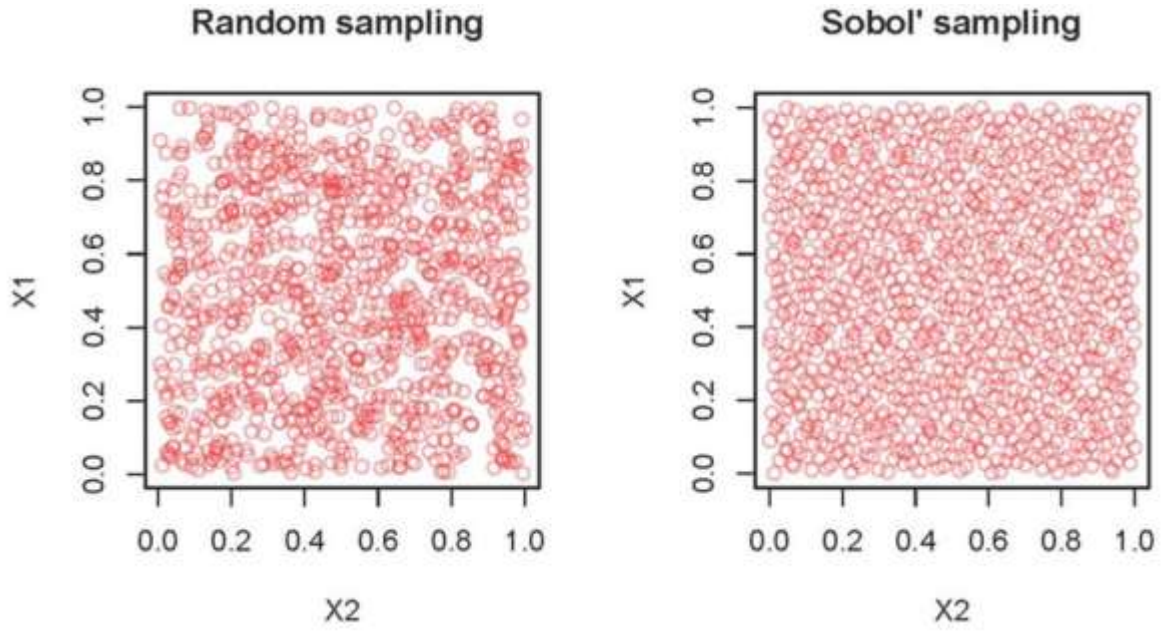
Halton VS Randomisés



Types d'Algorithmes Probabilistes

Algorithmes Quasi-Randomisés

Sobol VS Randomisés



Types d'Algorithmes Probabilistes

Algorithmes Quasi-Randomisés

Code Python
Halton VS Randomisés

```
import matplotlib.pyplot as plt
from scipy.stats.qmc import Halton
import numpy as np

n = 100
random_points = np.random.rand(n, 2)  -> Générer des points aléatoires

sampler = Halton(d=2, scramble=False) -> Générer des points quasi-randomisés (Halton)
quasi_points = sampler.random(n)

plt.figure(figsize=(12, 6)) ->Tracer les points
plt.subplot(1, 2, 1)
plt.scatter(random_points[:, 0], random_points[:, 1], color='blue', s=10) -> Points aléatoires
plt.title("Points aléatoires")
plt.grid()

plt.subplot(1, 2, 2)
plt.scatter(quasi_points[:, 0], quasi_points[:, 1], color='green', s=10) ->Points quasi-randomisés

plt.title("Points quasi-randomisés (Halton)")
plt.grid()
plt.tight_layout()
plt.show()
```

Types d'Algorithmes Probabilistes

Résumé des Différences entres les méthodes probabilistes

Type	Temps d'Exécution	Résultat	Exemple
Monte Carlo	Fixe	Approximatif	Intégration numérique
Las Vegas	Variable	Exact	Tri rapide avec pivot aléatoire
Hybrides	Fixe ou Variable	Exact ou Approximatif	Cryptographie hybride
Stochastiques	Fixe ou Adaptable	Approximatif	Recuit simulé
Quasi-Randomisés	Fixe	Approximatif précis	Méthode de Halton