Optimisation du problème du sac à dos par l'algorithme des colonies de fourmis

Réaliser par : Hamid JELLOUL

TABLE DES MATIERES

Introduction :

Dans le domaine de l'optimisation combinatoire, de nombreux problèmes difficiles peuvent être résolus efficacement à l'aide d'algorithmes bio-inspirés. Parmi eux, l'**algorithme des colonies de fourmis (Ant Colony Optimization – ACO)** est largement utilisé pour résoudre des problèmes comme le **problème du sac à dos (Knapsack Problem)**.

I. Compréhension du problème :

1. Le problème du sac à dos :

Le **problème du sac à dos** est un problème classique en informatique et en mathématiques :

On dispose d'un **sac** pouvant contenir une **capacité maximale** (par exemple 10 kg).

On a une **liste d'objets**, chacun ayant : un **poids** et une **valeur** (ou utilité).

2. Objectif :

L'objectif de ce projet est de **choisir un ensemble d'objets à mettre dans le sac sans dépasser sa capacité maximale**, de façon à **maximiser la valeur totale**.

➢ *Exemple simple :*

Objet	Poids (kg)	Valeur (dh)
A	3	60
B	2	100
C	4	120

- Capacité du sac = 5 kg
- Solution optimale : Objet B + Objet A → Poids = 5 kg, Valeur = 160 dh

II. Présentation de l'algorithme des colonies de fourmis (ACO) :

1. Origine et principe :

L'**algorithme ACO** est inspiré du comportement des **fourmis réelles** pour chercher de la nourriture. Lorsqu'une fourmi trouve un bon chemin vers une source de nourriture, elle **dépose une substance chimique appelée phéromone** sur son chemin. D'autres fourmis ont tendance à suivre les chemins avec **plus de phéromones**, renforçant les bons chemins au fil du temps.

2. Composants de base d'ACO :

- **Solutions** : construites progressivement par des agents (fourmis).
- **Phéromone** : mémoire collective partagée représentant la qualité des solutions.
- **Heuristique** : information locale (comme la valeur/poids d'un objet dans notre cas).
- **Probabilité** : une fourmi choisit les éléments à ajouter à sa solution en fonction d'une probabilité basée sur la phéromone et l'heuristique.
- **Évaporation** : la phéromone s'estompe avec le temps pour éviter le piégeage local.

III. Application de l'ACO au problème du sac à dos :

1. Représentation du problème :

Dans le cadre du **problème du sac à dos**, les **objets** représentent les **nœuds** du graphe. Les fourmis construisent une solution en **ajoutant un objet à la fois** dans le sac, tant que la capacité n'est pas dépassée.

2. Étapes de l'algorithme ACO pour le sac à dos :

- **Initialisation** :

- Définir la phéromone initiale pour chaque objet.

- Définir les paramètres (nombre de fourmis, taux d'évaporation…).

- **Construction des solutions** :

- Chaque fourmi commence avec un sac vide.

- À chaque étape, elle choisit probabilistiquement un objet non encore choisi en fonction de la quantité de phéromone sur l'objet et de la valeur heuristique (souvent valeur/poids).

- Elle ajoute l'objet si la capacité le permet.

- **Évaluation** :

- Chaque fourmi calcule la valeur totale de sa solution.

- **Mise à jour des phéromones** :

- Plus une solution est valable et de bonne qualité, plus la phéromone déposée est importante.

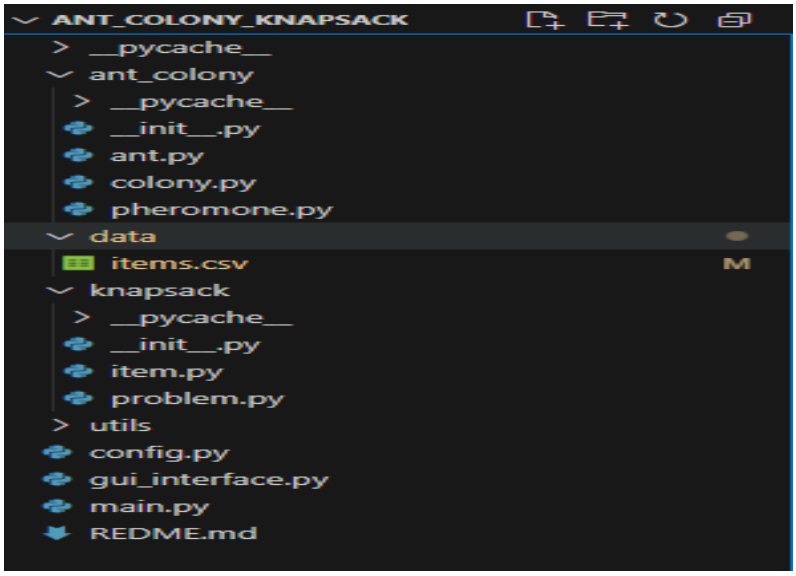
- Une évaporation est appliquée à toutes les phéromones pour éviter le surapprentissage.

- **Répétition** :

- Répéter le processus pendant un certain nombre d'itérations ou jusqu'à convergence.

IV. Architecture et paramétrage :

1. Architecture du système :



- **main.py :** Point d'entrée du programme
- **config.py :** Paramètres ACO et constantes
- **knapsack/item.py :** Classe représentant un objet
- **knapsack/problem.py :** Classe définissant le problème (capacité, vérifications)
- **ant_colony/ant.py :** Classe Fourmi ;Contient l'implémentation de l'algorithme ACO (logique de construction des solutions, gestion des phéromones, etc.).
- **ant_colony /colony.py :** Classe Colonie : gère toutes les fourmis
- **ant_colony /pheromone.py :** Gestion des niveaux de phéromone
- **data/items.csv :** Fichier d'entrée : poids et valeurs des objets
- **utils/heuristics.py :** Fonction de ratio valeur/poids
- **visualizer.py :** Pour la visualisation des résultats (graphique de convergence, etc.)
- **README.md :** Description du projet

2. Enchaînement typique de l'utilisateur dans les fichiers :

1. Lancement de l'application :

L'utilisateur exécute gui_interface.py (point d'entrée principal).

2. Chargement d'un problème :

Depuis l'interface (gui_interface.py), l'utilisateur charge un fichier de données (appel à une fonction de chargement, éventuellement dans knapsack.py ou utils/io.py).

3. Paramétrage de l'algorithme :

L'utilisateur règle les paramètres (alpha, beta, évaporation, nombre de fourmis, etc.) via l'interface graphique.

4. Exécution de l'algorithme :

L'utilisateur clique sur « Lancer » ou utilise un raccourci clavier. gui_interface.py appelle alors la fonction d'optimisation (qui peut se trouver dans ant_colony.py).

5. Traitement du problème :

L'algorithme ACO construit des solutions, met à jour les phéromones, etc. (ant_colony.py).

6. Affichage des résultats :

Les résultats sont renvoyés à l'interface (gui_interface.py), qui les affiche à l'utilisateur (tableaux, graphiques, etc.).

7. Sauvegarde ou export :

L'utilisateur peut sauvegarder les résultats ou exporter un graphique (appel à des fonctions dans utils/io.py ou visualizer.py).

3. Paramètres configurables :

Les paramètres configurables dans le projet ACO pour le sac à dos, ainsi que leur signification :

- **Alpha (α) :**

Rôle : Pondère l'importance de la phéromone dans le choix des objets.

Effet : Plus alpha est élevé, plus les fourmis suivent les traces de phéromone (exploitation des solutions déjà trouvées).

- **Beta (β) :**

Rôle : Pondère l'importance de l'heuristique (valeur/poids) dans le choix des objets.

Effet : Plus beta est élevé, plus les fourmis privilégient les objets avec un bon rapport valeur/poids (exploration guidée par l'heuristique).

- **Taux d'évaporation (ρ) :**

Rôle : Contrôle la diminution des phéromones à chaque itération.

Effet : Un taux élevé favorise l'exploration (les anciennes solutions sont vite oubliées), un taux faible favorise l'exploitation (les bonnes solutions persistent plus longtemps).

- **Nombre de fourmis :**

Rôle : Nombre de solutions candidates générées à chaque itération.

Effet : Plus il y a de fourmis, plus la recherche est diversifiée, mais le temps de calcul augmente.

- **Nombre d'itérations :**

Rôle : Nombre de cycles de construction de solutions et de mise à jour des phéromones.

Effet : Plus il y a d'itérations, plus l'algorithme a de chances de trouver une bonne solution, mais le temps de calcul augmente.

- **Capacité du sac :**

Rôle : Limite maximale de poids que le sac peut contenir.

Effet : Définit la contrainte principale du problème.

➢ ***Résumé :***

Ces paramètres permettent d'ajuster le comportement de l'algorithme entre exploration (découverte de nouvelles solutions) et exploitation (amélioration des solutions déjà trouvées), ainsi que la durée et la diversité de la recherche.
Ils sont généralement modifiables via l'interface graphique de ton projet.

V. Expérimentation et résultats :

1. Justification des choix d'implémentation :

- Utilisation de l'algorithme ACO (Ant Colony Optimization) :

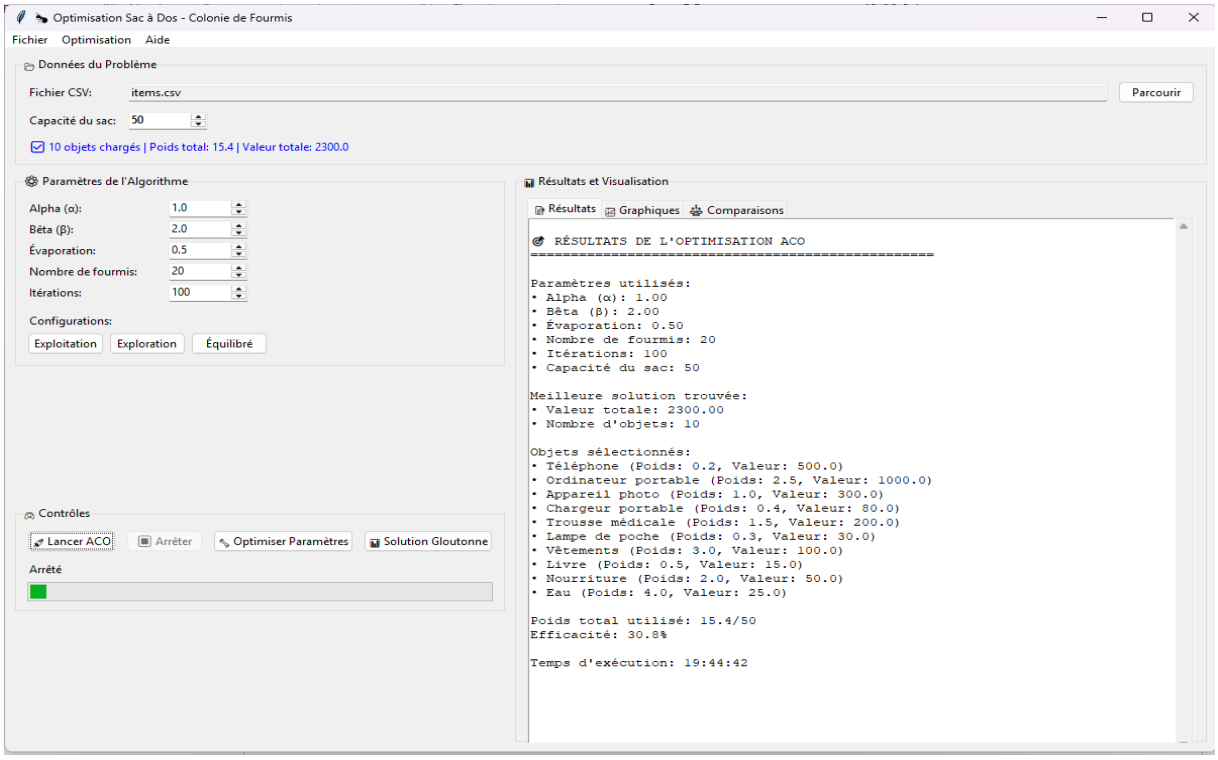
L'ACO est particulièrement adapté aux problèmes combinatoires comme le sac à dos, car il permet d'explorer efficacement un grand espace de solutions tout en exploitant les meilleures découvertes grâce au mécanisme de phéromones.

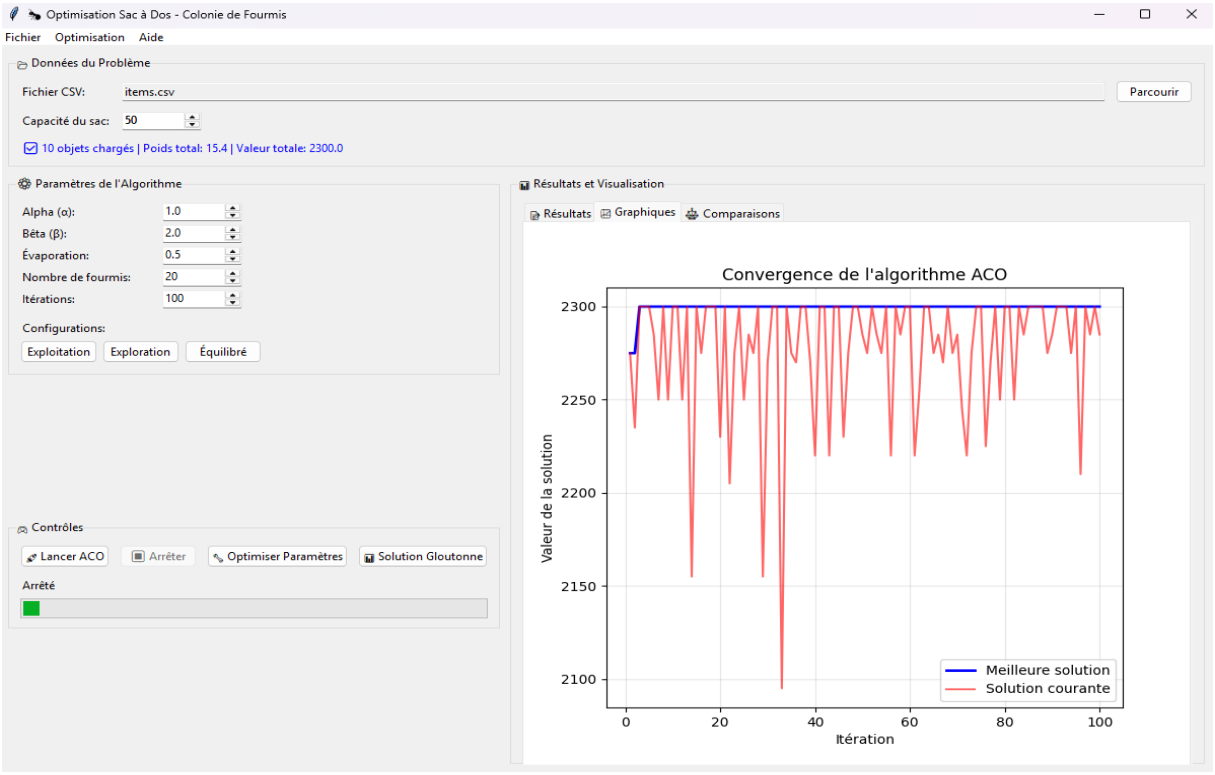
Son aspect probabiliste et adaptatif offre un bon compromis entre exploration et exploitation.

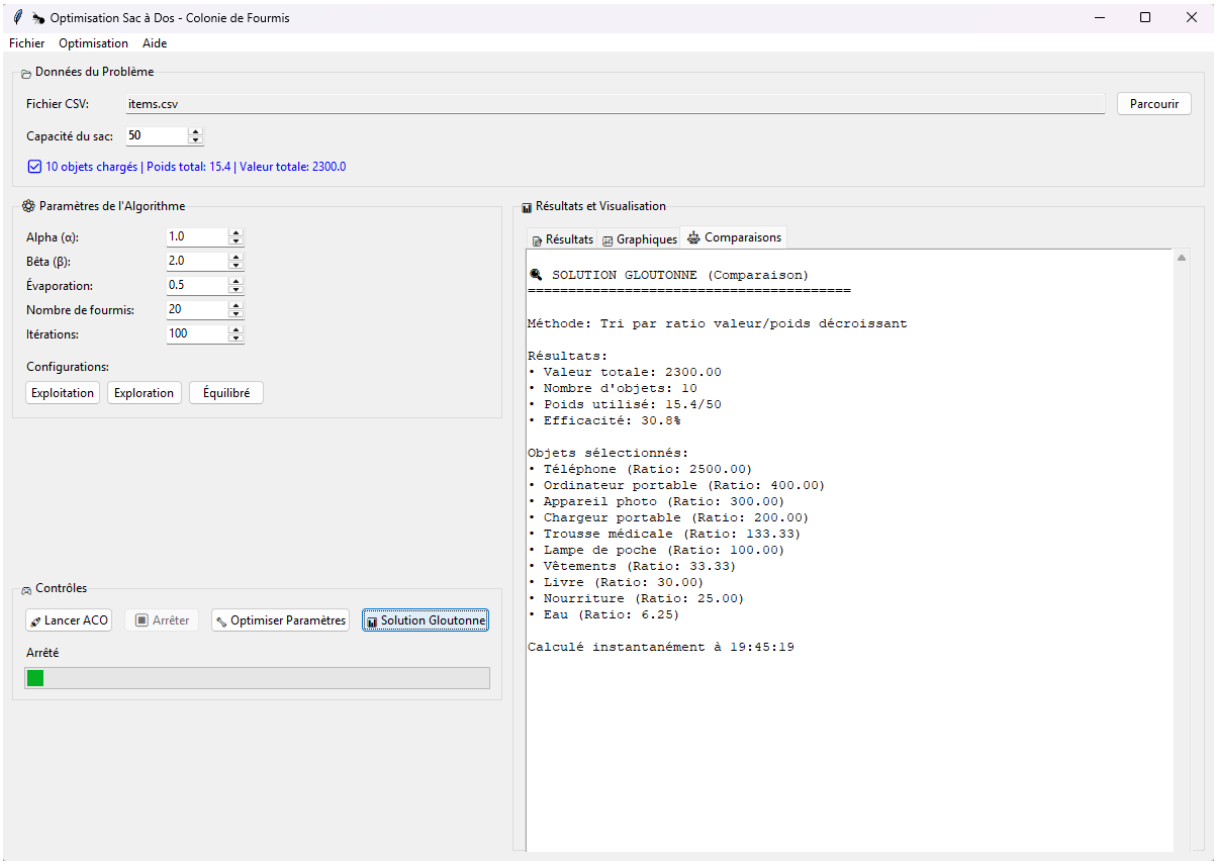
2. Les entrées :

data > items.csv > data

```
1  name,weight,value
2  Ordinateur portable,2.5,1000
3  Livre,0.5,15
4  Téléphone,0.2,500
5  Appareil photo,1.0,300
6  Vêtements,3.0,100
7  Trousse médicale,1.5,200
8  Nourriture,2.0,50
9  Eau,4.0,25
10 Lampe de poche,0.3,30
11 Chargeur portable,0.4,80
12
```




Optimisation Sac à Dos - Colonie de Fourmis
Fichier Optimisation Aide
Données du Problème
Fichier CSV: items.csv
Parcourir
Capacité du sac: 50
10 objets chargés | Poids total: 15.4 | Valeur totale: 2300.0
Paramètres de l'Algorithme
Alpha (α): 1.0
Béta (β): 2.0
Évaporation: 0.5
Nombre de fourmis: 20
Itérations: 100
Configurations:
Exploitation
Exploration
Équilibré
Contrôles
Lancer ACO
Arrêter
Optimiser Paramètres
Solution Gloutonne
Arrêté
Résultats et Visualisation
Résultats
Graphiques
Comparaisons
Convergence de l'algorithme ACO
Valeur de la solution
Itération
Meilleure solution
Solution courante


Optimisation Sac à Dos - Colonie de Fourmis
Fichier Optimisation Aide
Données du Problème
Fichier CSV: items.csv
Parcourir
Capacité du sac: 50
10 objets chargés | Poids total: 15.4 | Valeur totale: 2300.0
Paramètres de l'Algorithme
Alpha (α): 1.0
Béta (β): 2.0
Évaporation: 0.5
Nombre de fourmis: 20
Itérations: 100
Configurations:
Exploitation
Exploration
Équilibré
Contrôles
Lancer ACO
Arrêter
Optimiser Paramètres
Solution Gloutonne
Arrêté
Résultats et Visualisation
Résultats
Graphiques
Comparaisons
SOLUTION GLOUTONNE (Comparaison)
Méthode: Tri par ratio valeur/poids décroissant
Résultats:
• Valeur totale: 2300.00
• Nombre d'objets: 10
• Poids utilisé: 15.4/50
• Efficacité: 30.8%
Objets sélectionnés:
• Téléphone (Ratio: 2500.00)
• Ordinateur portable (Ratio: 400.00)
• Appareil photo (Ratio: 300.00)
• Chargeur portable (Ratio: 200.00)
• Trousse médicale (Ratio: 133.33)
• Lampe de poche (Ratio: 100.00)
• Vêtements (Ratio: 33.33)
• Livre (Ratio: 30.00)
• Nourriture (Ratio: 25.00)
• Eau (Ratio: 6.25)
Calculé instantanément à 19:45:19

VI. Analyse des résultats :

Voici une **analyse détaillée** du résultat d'optimisation ACO obtenu :

1. Paramètres utilisés :

- **Alpha (1.00)** : Importance modérée de la phéromone, favorisant un équilibre entre exploitation et exploration.
- **Bêta (2.00)** : Importance forte de l'heuristique (valeur/poids), ce qui oriente les fourmis vers les objets les plus rentables.
- **Évaporation (0.50)** : Taux d'évaporation moyen, permettant d'oublier progressivement les anciennes solutions tout en conservant une mémoire des bonnes pistes.
- **Nombre de fourmis (20)** et **Itérations (100)** : Ces valeurs assurent une exploration suffisante de l'espace des solutions.
- **Capacité du sac (50)** : Contrainte principale du problème.

2. Qualité de la solution :

- **Valeur totale obtenue : 2300.00**

C'est la somme des valeurs des objets sélectionnés, ce qui indique une solution de bonne qualité.

- **Nombre d'objets sélectionnés : 10**

Cela montre que la solution privilégie des objets à forte valeur et faible poids.

3. Composition du sac :

Les objets choisis sont variés, avec une nette priorité donnée à ceux ayant un excellent rapport valeur/poids (ex : téléphone, ordinateur portable, appareil photo).

Les objets utilitaires (chargeur, trousse médicale, lampe, etc.) sont également présents, ce qui montre que l'algorithme ne se limite pas aux objets de très forte valeur mais optimise l'ensemble.

4. Utilisation de la capacité :

- **Poids total utilisé : 15.4/50 (30.8%)**

L'algorithme n'a utilisé qu'environ un tiers de la capacité du sac.

- **Efficacité : 30.8%**

Cela signifie que, pour maximiser la valeur, il vaut mieux remplir le sac avec des objets légers et très rentables plutôt que de chercher à saturer la capacité.

5. Interprétation :

- **Stratégie adoptée :**

L'algorithme, guidé par un bêta élevé, a privilégié les objets à très forte valeur relative, ce qui explique la faible utilisation de la capacité mais une valeur totale élevée.

- **Possibilité d'amélioration :**

Si l'objectif était de remplir davantage le sac, il serait possible de tester d'autres paramètres (par exemple, diminuer bêta ou augmenter alpha) pour voir si des solutions plus « remplies » mais avec une valeur totale similaire peuvent être trouvées.

6. Performance :

- **Temps d'exécution :**

Le résultat a été obtenu rapidement, ce qui montre que l'algorithme est efficace pour cette instance.

Conclusion :

Ce projet a permis d'explorer l'application de l'algorithme des colonies de fourmis (ACO) à un problème d'optimisation combinatoire classique : le sac à dos 0-1. À travers la modélisation rigoureuse du problème et l'adaptation des mécanismes de l'ACO, nous avons pu concevoir une solution approchée capable de fournir des résultats satisfaisants dans un temps raisonnable.

L'implémentation réalisée en Python a démontré la pertinence de cette métaheuristique bio-inspirée pour gérer la complexité du problème, en particulier face à la croissance exponentielle du nombre de combinaisons possibles. Les expérimentations ont mis en lumière l'influence déterminante des paramètres de l'algorithme sur la qualité et la stabilité des solutions obtenues.

Par ailleurs, ce travail a mis en évidence certaines limites, notamment la sensibilité aux choix des paramètres et la possible stagnation sur des optima locaux. Ces observations ouvrent des perspectives d'amélioration, telles que l'optimisation automatique des paramètres, la diversification des heuristiques, ou encore l'extension de la méthode à des variantes plus complexes du sac à dos.

Enfin, ce projet constitue une expérience enrichissante tant sur le plan théorique qu'appliqué, consolidant des compétences en programmation algorithmique, modélisation mathématique et analyse expérimentale, tout en illustrant la puissance des algorithmes bio-inspirés dans le domaine de l'intelligence artificielle.
