Réalisation d'une preuve de concept de l'optimisation par essaim de particules à apprentissage auto-adaptatif (SLPSO) pour la planification de trajectoires de robots mobiles

JULES ALEXANDRE HABA

MASTER INGÉNIERIE DES SYSTÈMES COMPLEXES

(Industrie du Futur et systèmes intelligents)

(Option : Commande des Systèmes Critiques)

ÉCOLE NORMALE SUPERIEURE PARIS-SACLAY (ENS)

2026

TABLE DES MATIÈRES

LISTE DES ABRÉVIATIONS ET DES ACRONYMES

PSO	Optimisation par essaim de particules
SLPSO	Optimisation par essaim de particules à apprentissage autonome
GA	Algorithme génétique
PoC	Preuve de Concept

CHAPITRE UN : INTRODUCTION

La planification de trajectoire constitue un problème fondamental en robotique mobile autonome. Elle consiste à déterminer un chemin optimal permettant à un robot de se déplacer depuis une position initiale vers une position cible tout en évitant les obstacles (statiques ou dynamiques) présents dans l'environnement. Ce problème est particulièrement complexe dans des environnements encombrés, où plusieurs critères de performance doivent être simultanément pris en compte, tels que la longueur du chemin, la sécurité vis-à-vis des obstacles et la faisabilité du mouvement du robot [1].

Dans ce contexte, la planification de chemin peut être formulée comme un problème d'optimisation sous contraintes. La difficulté majeure de ce problème réside dans sa nature multi-objectif et la complexité de son espace de recherche. Trouver un équilibre entre la sécurité (distance aux obstacles) et l'efficacité (longueur du chemin) est complexe, car ces critères sont souvent antagonistes. De plus, à mesure que l'on augmente la précision de la trajectoire par l'ajout de points de passage, le nombre de variables à optimiser explose, rendant le problème difficile à résoudre en temps réel sans rester piégé dans des optimums locaux.

De nos jours, diverses méthodes ont été développées, notamment l'algorithme A*, la méthode des champs de potentiel, les algorithmes génétiques (GA) [2] [3] et l'optimisation par essaim de particules (PSO) [4], [5]. Le PSO est un algorithme d'optimisation stochastique inspiré du comportement collectif d'animaux sociaux. Bien qu'il présente une mise en œuvre simple et une bonne capacité de convergence, le PSO standard souffre de certaines limitations, notamment une convergence prématurée vers des optima locaux et une faible adaptabilité aux environnements complexes.

Pour pallier ces limitations, Li et Chou et al ont proposé un algorithme d'optimisation par essaim de particules à apprentissage auto-adaptatif (SLPSO) [1]. Cette approche introduit plusieurs stratégies d'apprentissage distinctes et un mécanisme adaptatif permettant à chaque particule de sélectionner dynamiquement la stratégie la plus appropriée en fonction de son efficacité passée. L'objectif est d'améliorer à la fois les capacités d'exploration et d'exploitation de l'algorithme.

Cependant, une question scientifique subsiste : le SLPSO conserve-t-il ses performances lorsqu'il est implémenté par d'autres chercheurs en respectant strictement la méthodologie

originale ? Afin de répondre à cette interrogation, la présente étude propose une reproduction rigoureuse de la méthodologie développée par Guangsheng Li et Wusheng Chou dans leur article *"Path Planning for Mobile Robot Using Self-Adaptive Learning Particle Swarm Optimization"*, publié en 2018 dans la revue *Science China: Information Sciences* [1].

Cette étude est motivée par les objectifs du cours *Techniques et outils pour la preuve de concepts*, qui vise à développer des compétences en programmation (Python et MATLAB), à reproduire et analyser des travaux scientifiques, à fournir un retour critique, à garantir la reproductibilité des résultats et à maîtriser les outils de présentation et de diffusion. Ce travail s'inscrit pleinement dans cette démarche.

1.1 Objectifs du PoC

1.1.1 Objectif general

L'objectif principal de cette PoC était d'implémenter et d'évaluer l'algorithme SLPSO pour la planification de trajectoire d'un robot mobile en environnement 2D avec obstacles statiques, et de comparer ses performances au PSO et au GA.

1.1.2 Objectif spécifiques

1) Lire l'article de Li et Chou, comprendre la méthodologie, identifier les entrées, les hypothèses de modélisation et les sorties produites.
2) Implémenter et afficher les métriques de performance du SLPSO, du PSO et du GA en respectant fidèlement la méthodologie proposée par les autheurs.
3) Comparer les résultats obtenus à ceux publiés dans l'article, réaliser des analyses critiques et fournir un retour constructif sur la publication.

1.2 Portée et limites du PoC

Cette étude s'est concentrée sur : (1) l'utilisation du langage de programmation Python dans Jupyter Notebook pour l'implémentation des algorithmes SLPSO, PSO et GA ; (2) la modélisation d'un nouvel environnement de test pour le robot mobile ; (3) la transformation de la théorie présentée dans l'article de Li et Chou en un modèle mathématique simulable et validable ; (4) la validation expérimentale de ces algorithmes dans Anaconda via Jupyter Notebook. Dans ce travail, aucun algorithme de planification de trajectoire n'est réinventé ; le SLPSO existant est reproduit, validé expérimentalement et analysé de manière critique.

Les limites de cette étude sont : (1) Les tests en conditions réelles avec un robot TurtleBot2 équipé de LiDAR n'ont pas été réalisés ; (2) L'environnement 2D décrit dans l'article n'a pas

été reproduit exactement, une simulation d'un environnement complexe a été modélisée ; (3) Le langage C++, le système d'exploitation Linux, le logiciel ROS (Robot Operating System), l'outil de simulation Gazebo et autres n'ont pas été utilisés.

1.3 Structure du rapport de PoC

Le reste de ce rapport est organisé comme suit. Au chapitre 2, la modélisation mathématique du problème, les hypothèses pour l'implémentation du SLPSO, du PSO et du GA sont présentés. Le chapitre 3 présente les résultats comparatifs des trois algorithmes obtenus après simulation, les analyses critiques par rapport aux données de la publication originale, ainsi que le retour constructif. Enfin, le chapitre 5 conclut le rapport du PoC et présente les orientations pour les travaux futurs.

.

CHAPTER DEUX : MÉTHODOLOGIE

Cette section présente l'approche adoptée lors de la phase d'implémentation des objectifs de cette PoC. La Figure 2-1 illustre l'organigramme de la méthodologie utilisée dans ce travail.

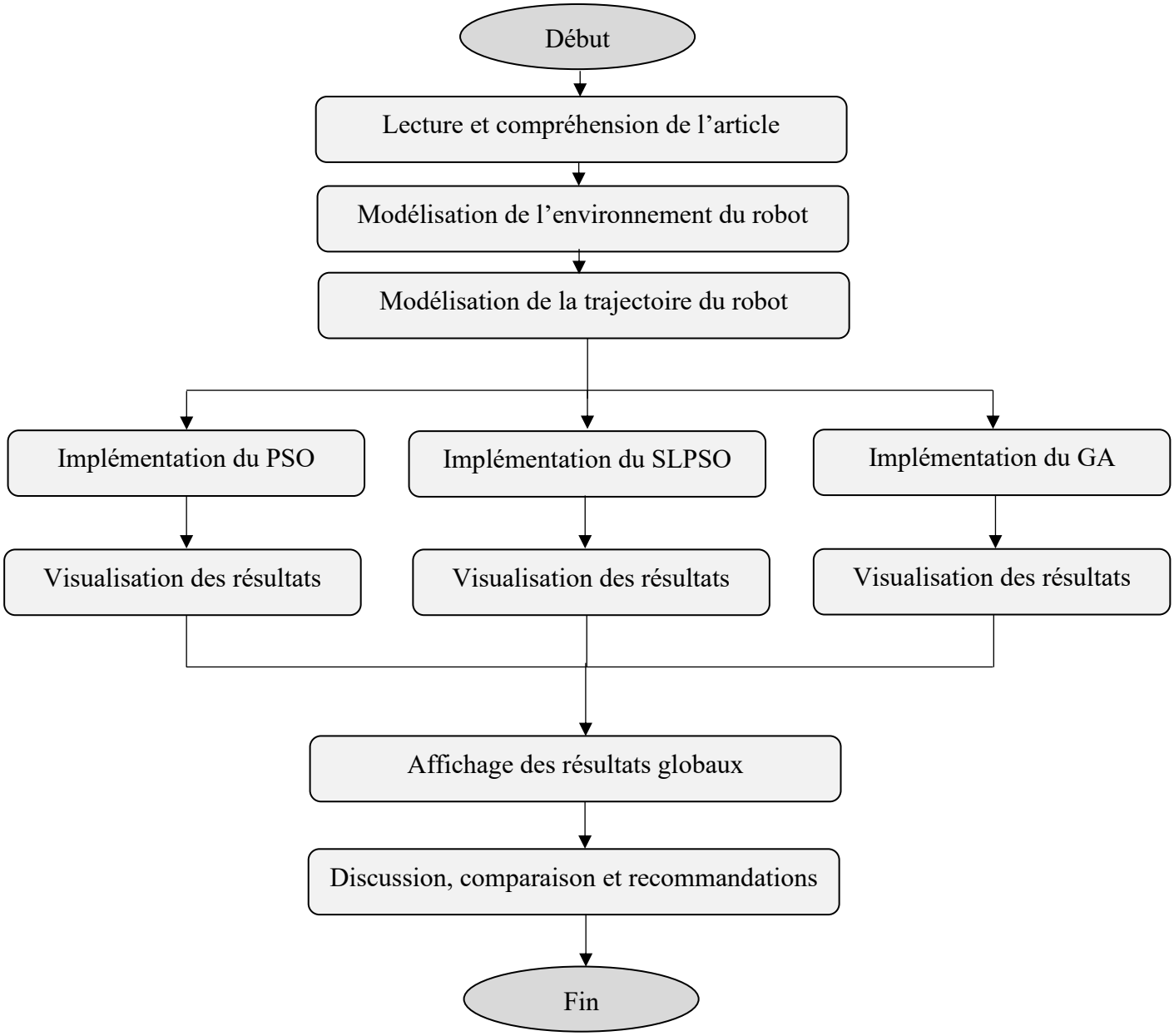

Figure 2-1: Diagramme d'activité du processus méthodologique du PoC

Le processus a débuté par la phase préparatoire de compréhension théorique et de modélisation de l'environnement ainsi que de la trajectoire. Ensuite, les algorithmes PSO, SLPSO et GA ont été implémentés, et leurs résultats ont été visualisés de manière indépendante. Enfin, une

analyse de synthèse a été réalisée afin de présenter les résultats globaux, d'effectuer une discussion comparative des performances et de formuler un retour constructif sur la publication.

3.1 Modélisation et critères d'évaluation de la trajectoire du robot mobile

L'évaluation de la qualité d'une trajectoire de robot mobile repose sur plusieurs critères, car un chemin très court peut augmenter le risque de collision, tandis qu'un chemin très sûr peut être trop long ou difficile à suivre. Trois critères principaux sont utilisés, conformément à Li et Chou : la longueur du chemin, le risque de collision et la douceur (lissage) de la trajectoire [1].

- **Longueur totale du chemin (L(P))** :

La longueur du chemin est un critère fondamental en planification de trajectoire, car elle influence le temps de déplacement et la consommation énergétique du robot. Elle correspond à la distance totale parcourue entre le point de départ et la cible, l'objectif étant de la minimiser. Elle est calculée suivant selon l'équation 3.1 présentée dans l'article [1].

$$L(P) = \sum_{i=0}^{D} \|P_{i+1} - P_i\| \qquad (3.1)$$

- **Le degré de risque de collision (R(X)) :**

Le degré de risque de collision quantifie la proximité du chemin par rapport aux obstacles. Contrairement à une approche binaire, il évalue progressivement le danger : plus la valeur est faible, plus la trajectoire est sûre pour le robot. Le risque est modélisé par une fonction gaussienne bidimensionnelle [1]. L'équation 3.2 définit le risque total de collision pour la trajectoire et l'équation 3.3 calcule le risque de collision dû à un obstacle spécifique j. Ce modèle pénalise les trajectoires proches des obstacles et favorise des chemins sûrs.

$$R(X_{rob}, X_{obs}) = \sum_{j=1}^{n} R_{rob,obsj}(X_{rob}, X_{obsj}) \qquad (3.2)$$

$$R_{rob,obsj}(X_{rob}, X_{obsj}) = \begin{cases} e^{-0.5\,(\frac{\|X_{rob}-X_{obsj}\|^2}{\rho^2})^C}, & si\ \|X_{rob} - X_{obsj}\| \le R_j, sinon \\ 0 \end{cases} \qquad (3.3)$$

- **Le degré de fluidité ou douceur de la trajectoire (S(P)) :**

La douceur de la trajectoire évalue la régularité des changements de direction, essentielle pour la faisabilité du déplacement du robot et la stabilité de ses mouvements. Plus cette valeur est faible, plus le chemin est facile à suivre pour le robot. Elle est mesurée par la variation angulaire entre deux segments consécutifs du chemin et est définie selon l'équation 3.4, telle que présentée dans l'article [1]. θ_i est l'angle entre les segments P_iP_{i+1} et $P_{i+1}P_{i+2}$.

$$S(P) = \sum_{i=1}^{D} \theta_i \qquad (3.4)$$

- **Le Coût global (J[L(P), R(X), S(P)]) :**

Les trois critères précédents étant conflictuels, le problème de planification de trajectoire est formulé comme une optimisation multi-objectifs sous contraintes. L'objectif global est de minimiser la fonction J tout en garantissant que le chemin généré évite les obstacles présents dans l'environnement. Elle est calculée suivant selon l'équation 3.5 présentée dans l'article [1].

$$J[\mathrm{L(P)}, \mathrm{R(X)}, \mathrm{S(P)}] = w_1 * L(P) + w_2 * R(X) + w_3 * S(P) \qquad (3.5)$$

Où w_1, w_2 et w_3 sont des coefficients de pondération reflétant l'importance relative de chaque critère. Conformément à [1], ces coefficients sont fixés à w_1 = 0.6, $w_2 = 0.3$ et $w_3 = 0.1$ privilégiant la réduction de la longueur du chemin tout en maintenant un compromis entre sécurité et douceur de trajectoire.

3.2 Présentation du jeu de données et de ses métriques

L'évaluation d'un algorithme de planification de trajectoire nécessite des environnements de test représentatifs. En robotique mobile, il n'existe pas de jeux de données standards, c'est-à-dire d'environnements universellement acceptés pour la planification de trajectoire. Il est donc courant de modéliser et d'utiliser des environnements comportant des obstacles. Dans ce PoC, l'environnement est modélisé comme un espace 2D discret de taille fixe, dans lequel évolue un robot mobile ponctuel. Il est défini par :

- une position initiale $S = (x_s, y_s)$
- une position cible $T = (x_t, y_t)$
- un ensemble d'obstacles statiques représentés par des cercles, caractérisés par $Obstacles_j = [x_j, y_j, \ rayon_j]$.

L'environnement est supposé connu à l'avance et sans obstacles dynamiques, conformément à l'article de référence, afin de se concentrer sur l'évaluation des performances de l'algorithme. Les jeux de données sont générés à l'aide d'un code python, garantissant la création d'environnements complexes et la reproductibilité des expériences. Les principales caractéristiques du jeu de données utilisé sont présentées dans le Tableau 2.1.

Table 2.1: Caractéristiques du jeu de données

Caractéristique	Description	Valeur
Dimension de l'environnement	Espace 2D discret	S = 12m×12m = 144 m^2
Nombre de scénarios	Environnement de test unique	1
Type d'obstacles	Obstacles circulaires statiques	9 obstacles
Surface totale des obstacles	Somme des surfaces des obstacles circulaires	$S_{obs} = \pi \times \sum_{i=1}^{9} r_i^2 = 37{,}49\ m^2$
Espace libre de déplacement	Surface accessible au robot	$S_{libre} = S - S_{obs} = 106{,}50\ m^2$
Densité de l'espace de déplacement	Rapport entre la surface occupée par les obstacles et la surface de l'environnement	$\delta = \frac{S_{obs}}{S} = 26\ \%$
Position de départ	Coordonnées fixes du robot	$S = (x_s, y_s) = (1, 1)$
Position d'arrivée	Coordonnées fixes de la cible	$T = (x_t, y_t) = (11, 11)$

Ces caractéristiques permettent de décrire la complexité de l'environnement de test utilisé dans cette PoC. La dimension de l'espace, le nombre et la surface des obstacles, ainsi que la densité de l'espace de déplacement fournissent des indicateurs objectifs de la difficulté du scénario. Ces métriques constituent une base essentielle pour l'analyse et l'interprétation des performances des algorithmes de planification de trajectoire.

3.3 Modélisation et mise en œuvre du PSO

Le PSO est un algorithme d'optimisation stochastique basé sur une population. Il s'inspire du comportement social des nuées d'oiseaux ou des bancs de poissons en mouvement dans la nature. Le concept fondamental repose sur la notion de particule, et l'implémentation du PSO est simple, car il n'utilise qu'une seule stratégie d'apprentissage « monotone » pour toutes les particules. Chaque particule évolue au sein d'un espace de recherche à *D* dimensions et possède

deux attributs dynamiques : sa position X_k^i et sa vitesse V_k^i. La population, composée de *N* particules, collabore pour explorer l'espace de manière optimale.

Le PSO intègre une dimension cognitive et sociale grâce à une mémoire interne : *Pbest* (X_{pbest}), la meilleure position personnelle enregistrée par la particule, et *Gbest* (X_{gbest}), la meilleure position globale découverte par l'ensemble de l'essaim. À chaque itération, l'algorithme évalue l'adaptation (fitness) de chaque particule. Les vitesses et les positions sont ensuite mises à jour pour l'itération suivante *k+1* selon les équations de mouvement 3.6 et 3.7 ce qui permet à l'essaim de converger vers la solution optimale [1].

$$V_i^{k+1} = w * V_i^k + c_1 * r_1 * \left(Pbest_i - X_i^k\right) + c_2 * r_2 * \left(Gbest_i - X_i^k\right) \qquad (3.6)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \qquad (3.7)$$

Où V_i^k est la vitesse de la particule i à l'itération k, X_i^k sa position, $Pbest_i$ la meilleure position individuelle, $Gbest_i$ la meilleure position globale, w le facteur d'inertie, $c_1 et\ c_2$ les coefficients d'accélération cognitive et sociale, $r_1 et\ r_2$ sont des variables aléatoires $\in$ [0, 1]. Dans le contexte de la planification de trajectoire, ces équations permettent d'ajuster progressivement les points intermédiaires du chemin afin de minimiser la fonction objective globale.

Le pseudocode général de l'algorithme PSO appliqué à la planification de trajectoire est présenté ci-dessous.

Algorithme 1 : PSO pour la planification de trajectoire d'un robot mobile

```
# Identification des entrées
Position de depart S, position cible T, Environnement 2D avec obstacles statiques
Nombre de particules N, Nombre d'itérations (Max_iter), Nombre de waypoints D
Paramètres du PSO : facteur d'inertie w,  coefficients cognitifs et sociaux c1, c2

# Identification de la Sortie
Trajectoire optimale reliant S à T

# Procédure :
Initialiser un essaim de N particules représentant des trajectoires candidates
Initialiser aléatoirement la vitesse de chaque particule
Évaluer chaque trajectoire avec la fonction objectif globale
Initialiser les meilleures positions individuelles (Pbest) et globale (Gbest).
Pour chaque itération :
```

15 Mettre à jour la vitesse des particules (Eq 3.6)
16 Mettre à jour leurs positions (Eq 3.7)
17 Corriger les trajectoires non valides.
18 Réévaluer les nouvelles trajectoires générées
19 Mettre à jour (*Pbest*) et (*Gbest*) si une amélioration est observée
20 Retourner la trajectoire associée à la meilleure solution globale trouvée.

3.4 Modélisation et mise en œuvre du SLPSO

Le SLPSO est une extension du PSO, conçue pour améliorer ses capacités d'exploration et éviter la convergence prématurée vers des optima locaux. Contrairement au PSO standard, où les règles de mise à jour sont fixes, le SLPSO introduit un mécanisme d'apprentissage auto-adaptatif basé sur plusieurs stratégies d'apprentissage concurrentes.

Dans le cadre de la planification de trajectoire d'un robot mobile, chaque particule représente une trajectoire candidate reliant une position de départ à une position cible. L'objectif de l'algorithme est de minimiser une fonction objectif globale intégrant la longueur du chemin, le risque de collision et la douceur de la trajectoire.

Le SLPSO repose sur l'utilisation de quatre stratégies d'apprentissage distinctes, notées $\emptyset_s = \{a, b, c, d\} = \{Exploitation, Convergence,\ Saut\ et\ Exploration\}$. La sélection de l'opérateur dépend d'un mécanisme auto-adaptatif. À chaque itération, chaque particule sélectionne une stratégie selon une probabilité associée, mise à jour en fonction de la performance passée de la stratégie. La mise à jour de la vitesse et de la position d'une particule est respectivement donnée par les équations 3.8 et 3.9 :

$$V_i^{k+1} = w * V_i^k + \emptyset_s * \left(X_{learned} - X_i^k\right) \qquad (3.8)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \qquad (3.9)$$

où w est le facteur d'inertie, $\emptyset_s$ un coefficient dépendant de la stratégie d'apprentissage sélectionnée et $X_{learned}$ la position de référence définie par la stratégie choisie.

Les quatre stratégies d'apprentissage utilisées dans le SLPSO sont définies comme suit :

- **Opérateur a : apprentissage individuel ou personnel (Exploitation)**

Cet opérateur permet à la particule d'exploiter sa propre expérience passée. Il est défini par les équations (3.10) et (3.11), où n_3 est un coefficient d'accélération, $rand_i^k$ est une variable aléatoire ∈ [0, 1] et $X_{Pbest_i}^k$ la meilleure position historique de la particule i.

$$V_i^k = w * V_i^k + n_3 * rand_i^k * \left(X_{Pbest_i}^k - X_i^k\right) \qquad (3.10)$$

$$X_{learned} = Pbest_i \qquad (3.11)$$

- **Opérateur b : apprentissage local ou celle voisin proche (Convergence)**

Cet opérateur permet à la particule d'apprendre à partir de la meilleure particule voisine. Il est défini par les équations (3.12) et (3.13), où $Pbest_{i\ plus\ proche}$ est la meilleure position individuelle parmi les particules voisines de la particule i.

$$V_i^k = w * V_i^k + n_3 * rand_i^k \left(X_{Pbest_{i\ plus\ proche}}^k - X_i^k\right) \qquad (3.12)$$

$$X_{learned} = Pbest_{i\ plus\ proche} \qquad (3.13)$$

- **Opérateur c : apprentissage combines ou perturbation gaussienne (Saut)**

Cet opérateur favorise l'exploration de l'espace de recherche par une perturbation aléatoire. Il est défini par les équations (3.14) et (3.15), où $V_{average}^k$ est la vitesse moyenne de l'essaim à l'itération k et $N(0,1)$ est une variable aléatoire suivant une loi normale centrée réduite.

$$X_i^k = X_i^k + V_{average}^k * N(0,1) \qquad (3.14)$$

$$X_{learned} = \frac{Pbest_i + Gbest}{2} \qquad (3.15)$$

- **Opérateur d : apprentissage global (Exploration)**

Cet opérateur oriente la particule vers la meilleure solution globale trouvée par l'essaim. Il est défini par les équations (3.16) et (3.17), où X_{Gbest}^kest la meilleure position globale de l'essaim à l'itération k.

$$V_i^k = w * V_i^k + n_3 * rand_i^k \left(X_{Gbest}^k - X_i^k\right) \qquad (3.16)$$

$$X_{learned} = Gbest \qquad (3.17)$$

Ces stratégies permettent d'équilibrer l'exploration globale et l'exploitation locale au cours du processus d'optimisation. La spécificité du SLPSO réside dans la **mise à jour dynamique des probabilités de sélection des stratégies**. Le cœur de cette méthode est son mécanisme auto-adaptatif. À chaque itération, une particule ne suit pas une règle prédéfinie, mais sélectionne une stratégie selon une probabilité associée. La sélection s'appuie sur un système de recompense : chaque stratégie $\emptyset_s = \{a, b, c, d\}$ possède un score de performance Q_s^k. Ce score évolue selon l'équation 3.18, où ΔJ est l'amélioration apportée à la fonction objective par la stratégie utilisée. Plus une stratégie permet de réduire le coût du chemin, plus son score augmente.

$$Q_s^{k+1} = Q_s^k + \Delta J \qquad (3.18)$$

Enfin, pour transformer ces scores en un choix décisionnel, l'algorithme procède à une normalisation pour obtenir les probabilités de sélection. La probabilité d'une stratégie pour l'itération suivante est calculée selon l'équation 3.19

$$P_s^{k+1} = \frac{Q_s^k}{\sum_{j=1}^{4} Q_s^{k+1}} \qquad (3.19)$$

Ce processus garantit que les stratégies les plus efficaces à un instant "t" du parcours du robot sont favorisées, tandis que les moins performantes perdent en influence. Une fois la stratégie choisie, la mise à jour physique de la particule suit les équations de vitesse et de position (3.8) et (3.9), adaptées selon les paramètres de l'opérateur sélectionné.

Le pseudocode général de l'algorithme SLPSO appliqué à la planification de trajectoire est présenté ci-dessous.

Algorithme 2 : SLPSO pour la planification de trajectoire d'un robot mobile

```
# Identification des entrées
Position de depart S, position cible T, Environnement 2D avec obstacles statiques
Nombre de particules N, Nombre d'itérations (Max_iter), Nombre de waypoints D
Paramètres du S LPSO : stratégies d'apprentissage (opérateur a, b, c, d),
Probabilités de sélection des opérateurs U_f

# Identification de la Sortie
```

```
8   Trajectoire optimale reliant S à T
9
10  # Procédure :
11  Initialiser un essaim de N particules représentant des trajectoires candidates
12  Initialiser les vitesses des particules.
13  Initialiser les probabilités associées à chaque stratégie d'apprentissage
14  Évaluer la qualité initiale de chaque trajectoire à l'aide de la fonction objectif globale
15  Initialiser les meilleures positions individuelles (Pbest) et globale (Gbest).
16  Pour chaque itération :
17      Sélectionner une stratégie d'apprentissage pour chaque particule
18      Mettre à jour la vitesse selon la stratégie choisie
19      Mettre à jour la position de la particule et vérifier la validité du chemin généré
20      Évaluer la nouvelle trajectoire à l'aide de la fonction objectif
21      Mettre à jour les meilleures positions individuelles (Pbest) et globale (Gbest).
22      Mettre à jour la performance de chaque stratégie d'apprentissage
23      Ajuster dynamiquement et automatiquement leurs probabilités de sélection.
24  Retourner la trajectoire correspondant à la meilleure solution globale obtenue.
```

La différence entre le PSO et le SLPSO réside dans le fonctionnement de l'algorithme. Le PSO utilise une formule fixe pour chaque particule, basée sur son propre souvenir et celui du groupe, tandis que le SLPSO choisit dynamiquement parmi quatre stratégies d'apprentissage pour optimiser son comportement. En clair, le SLPSO ajoute une intelligence collective absente du PSO. Malgré cette théorie séduisante, il reste essentiel de tester les deux algorithmes et de comparer leurs performances.

3.5 Modélisation et mise en œuvre du GA

L'Algorithme Génétique (GA) est une méthode d'optimisation stochastique qui puise son inspiration dans les mécanismes fondamentaux de la biologie et de l'évolution naturelle. Contrairement au PSO et au SLPSO qui s'appuient sur le mouvement de particules, le GA fait évoluer une population de chromosomes.Au sein de ce paradigme, une population de solutions candidates, appelées individus, évolue au fil des générations successives. Dans ce contexte, chaque individu (chromosome) représente un chemin potentiel, composé de D gènes, correspondant aux points de passage intermédiaires (*waypoints*) entre le départ S à la cible T. Le processus d'évolution repose sur trois opérateurs fondamentaux :

- **la selection:**

Cet opérateur choisit les individus les plus performants pour la reproduction, favorisant la transmission des meilleures caractéristiques au fil des générations. Le processus privilégie les trajectoires ayant une mesure d'aptitude (*fitness*) élevée. Le fitness est défini comme l'inverse de la fonction objectif J(P), selon à l'équation (3.20).

$$fitness(P) = \frac{1}{1 + J(P)} \quad (3.20)$$

- **le croisement**

Il combine l'information génétique de deux parents afin de produire de nouveaux individus. Dans ce contexte, le croisement consiste à échanger une partie des points intermédiaires entre deux trajectoires selon l'équation 3.21 ou p_c est le taux de croisement.

$$P_{child} = p_c * P_{parent\,1} + (1 - p_c) * P_{parent\,1} \quad (3.21)$$

- **la mutation**

Elle introduit une perturbation aléatoire sur certains gènes afin de maintenir la diversité génétique et d'éviter une convergence prématurée. Ce processus ajoute une perturbation δ suivant une loi normale $N(0, \sigma^2)$, à certains points du chemin : $x_i^{new} = x_i + \delta$

À chaque génération, une nouvelle population est formée. Pour assurer une amélioration progressive, les individus les plus performants sont conservés d'une itération à l'autre. Le processus itératif se poursuit jusqu'à l'atteinte du nombre maximal de générations ou d'un critère de convergence prédéfini.

Pour cette étude, la population initiale est fixée à 30 chromosomes, garantissant une base de comparaison équitable avec le PSO et le SLPSO. Les paramètres de contrôle retenus sont un taux de croisement P_c = 0,9 et un taux de mutation P_m = 0,08. Cette configuration permet au GA d'explorer efficacement les trajectoires dans des environnements complexes afin de valider ses performances au sein de cette preuve de concept.

Le pseudocode général de l'algorithme SLPSO appliqué à la planification de trajectoire est présenté ci-dessous.

Algorithme 3 : GA pour la planification de trajectoire d'un robot mobile

```
# Identification des entrées
Position de depart S, position cible T, Environnement 2D avec obstacles statiques
Taille de population P, Nombre générations (G_max)  Nombre de waypoints D
```

Paramètres du GA : taux de croisement p_c , taux de mutation p_m

Identification de la Sortie

Trajectoire optimale reliant S à T

Procédure :

Initialiser une population P de trajectoires candidates.

Évaluer chaque trajectoire à l'aide de la fonction objectif globale

Pour chaque génération :

    Sélectionner les individus parents en fonction de leur fitness

    Appliquer l'opérateur de croisement afin de générer de nouveaux individus

    Appliquer la mutation pour introduire de la diversité génétique dans la population

    Corriger les trajectoires invalides afin d'assurer l'absence de collision

    Évaluer les nouveaux individus générés

    Mettre à jour la population en conservant les meilleurs individus

Retourner la trajectoire correspondant au meilleur individu de la population finale.

Les trois algorithmes partagent le même objectif d'optimisation mais diffèrent dans leurs mécanismes d'exploration et d'exploitation. Le PSO repose sur une dynamique collective simple, le SLPSO introduit une adaptation intelligente des stratégies d'apprentissage, tandis que le GA exploite des opérateurs évolutionnaires. Cette diversité algorithmique permet une comparaison pertinente des performances dans le cadre des expériences numériques présentées ultérieurement. Les résultats des discussions seront présentés dans le chapitre trois.

3.6 Ressources informatiques

3.6.1 Les matériels

Le tableau 2.2 montre le matériel utilisé dans cette étude.

Table 2.2: Liste des matériaux

Éléments	Caractéristiques techniques
Ordinateur	HP ZBOOK Studio 8360 G5, 16 GB RAM, Intel Core i7, 9th Gen, 64-bit OS, 500 GB SSD, 4 GB Nvidia Quadro P1000 GPU
Périphériques	Wifi eduroam

3.6.2 Les logiciels

Le tableau 2.3 présente le logiciel utilisé dans cette étude.

Table 2.3: Liste des logiciels et outils de développement

Catégorie	Éléments	Application
Système d'exploitation	Windows	Plateforme hôte pour l'exécution des simulations.
Environnement de développement	Anaconda / Jupyter Notebook	Gestion des codes et prototypage interactif pour les tests et la visualisation.
Langage de programmation	Python 3.12 (64 bit)	Langage principal utilisé pour l'implémentation des algorithmes (PSO, SLPSO, GA).
Bibliothèques Python	NumPy, Matplotlib, pendas, SciPy, Random, Math, Time	Calculs matriciels, génération d'aléas, optimisation mathématique et visualisation des trajectoires.
IDE (Environnement de code)	Visual studio code	Édition, débogage et structuration du code source.

CHAPITRE TROIS : RESULTATS ET DISCUSSIONS

Ce chapitre est important car c'est ici qu'est démontré que l'article a été compris, que les algorithmes ont été bien implémentés, et qu'il est possible d'analyser scientifiquement des résultats. Le travail ici n'est pas de décrire le code sur Jupyter Notebook, mais d'interpréter les résultats de ce code. Ce chapitre doit savoir répondre à ces questions : Est-ce que le SLPSO fait mieux que le PSO et le GA ? Pourquoi ? Dans quels cas ? Avec quelles limites ?

4.1 Configuration expérimentale du jeu de données

Afin de garantir la reproductibilité des tests et la crédibilité des comparaisons liées au travail de Li et Chou [1], un cadre de simulation a été défini. Les figures 3.1 et 3.2 présentent respectivement le modèle d'environnement original issu de l'article [1] et le modèle reproduit pour cette PoC dédiée à la planification de chemin de robot mobile.

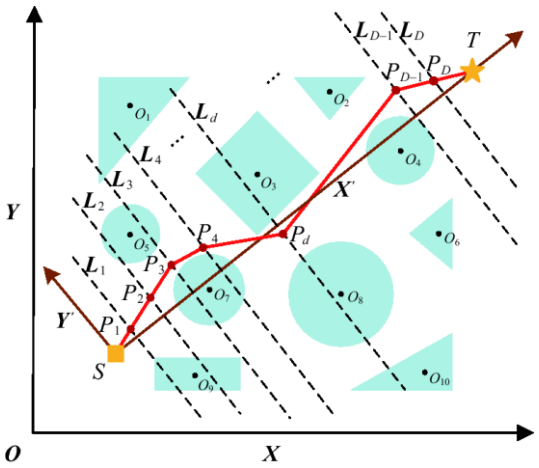

Figure 3-1: Modèle d'environnement original [1]

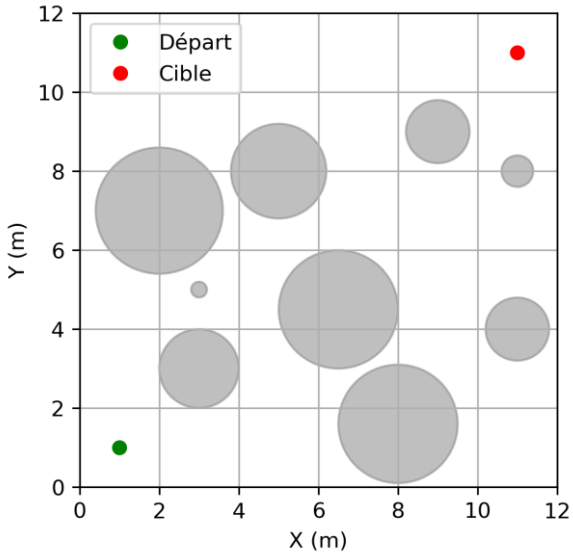

Figure 3-1: Modèle d'environnement reproduit pour le PoC

Bien que la taille de l'environnement soit identique dans les deux cas $12\,m * 12m$, des ajustements ont été réalisés pour la mise en œuvre pratique. Les points de départ et d'arrivée sont fixés respectivement à $(1, 1)$ et $(11, 11)$ afin de traverser l'espace en diagonale.

L'environnement du modèle original de Li et Chou [1] comporte 10 obstacles de formes variées (cercles, rectangles et carrés). En revanche, l'environnement modélisé pour le PoC utilise 9 obstacles, tous de forme circulaire. Ce choix permet de simplifier le calcul des distances de sécurité et la détection de collision sans sacrifier la complexité du problème. Ces obstacles sont très bien repartis pour créer un mouvement complexe de déplacement du robot mobile.

La surface totale occupée par les obstacles dans l'environnement du PoC est de 37,49 m². Cela signifie que 26 % de l'espace est obstrué, laissant 106,50 m² de zone libre pour la navigation. Une telle densité est représentative d'un environnement encombré, ce qui permet de tester efficacement la capacité des algorithmes (PSO, SLPSO et GA) à s'échapper des optima locaux et à trouver un chemin optimal dans un espace restreint.

4.2 Comparaison des performances dans l'environnement de simulation

Cette section présente l'analyse des performances entre le SLPSO, le PSO et le GA. Les expériences ont été menées au sein d'un scénario de simulation unique avec une variation de la dimension D (de 5 à 30 points de passage) pour garantir la cohérence des résultats.

L'implémentation de la planification de chemin a été réalisée sous Jupyter Notebook. Le langage Python a été utilisé avec les bibliothèques numPy, matplotlib, pandas et time …

L'environnement de simulation, illustré par la Figure 4.2, intègre des obstacles de tailles variées afin de représenter un environnement complexe et réaliste. Dans l'algorithme SLPSO, le nombre de points de passage (*waypoints*) du chemin optimal dépend de la dimension D des particules créées. Cette sous-section étudie l'effet de la variation de cette dimension sur l'efficacité globale de la planification.

- **Nombre de waypoints dans le chemin (D = 20)**

Les résultats obtenus lors de la simulation, pour un nombre de points de passage fixé à D = 20, sont présentés ci-dessous. Le tableau 3.1 présente une synthèse comparative de ces performances : la longueur du chemin (L), le risque de collision (R), la fluidité (S), le coût global (J) et le temps de calcul. Les chemins des robots et les courbes de convergence obtenus après l'implémentation des algorithmes SLPSO, PSO et GA sont présentés respectivement dans les figures 3.3 et 3.4.

Table 3.1 : Synthèse comparative des performances (D = 5)

Algorithme	**L(P)**	**R(X)**	**S(P)**	**Cout J**	**Temps de calcul**
SLPSO	18.457	0.003	7.891	8.173	307.7
PSO	21.166	0	2.707	8.737	326.9
GA	26.821	0.006	11.279	11.859	338.8

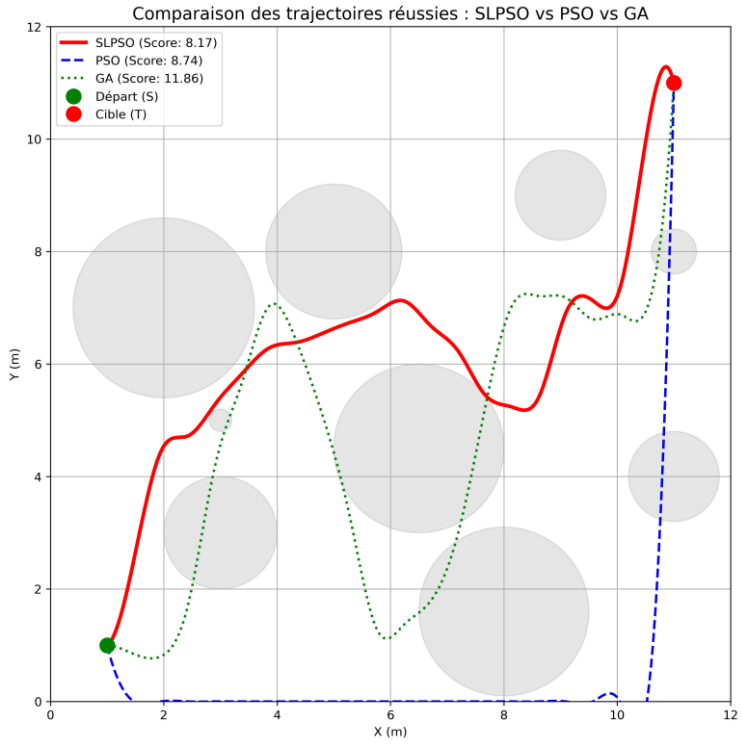

Figure 3-3: Chemins planifiés par les algorithmes SLPSO, PSO et GA (D = 5)

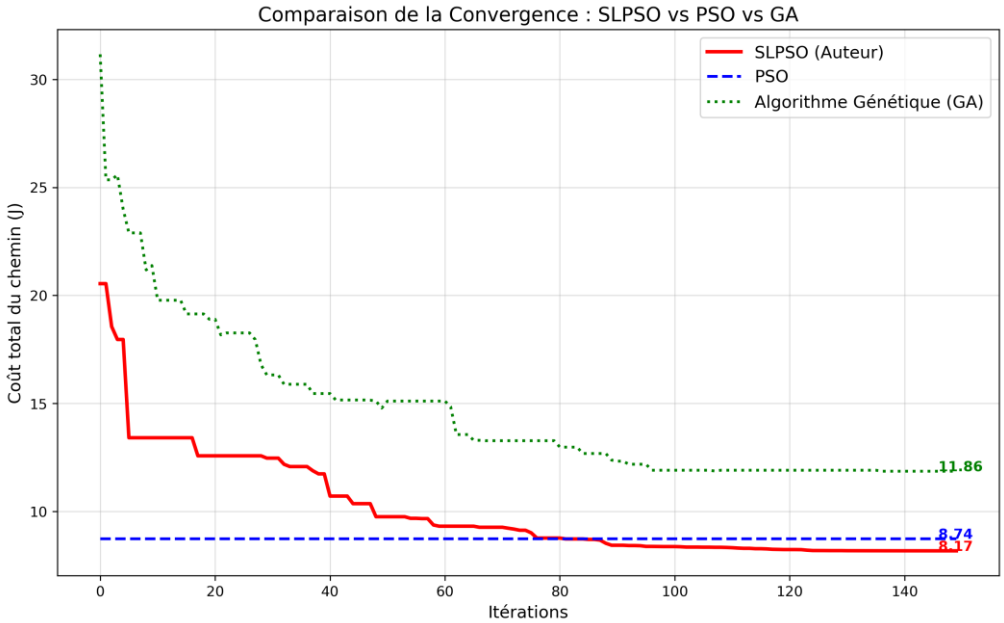

Figure *3-4*: Courbes de convergence du coût global (J) pour SLPSO, PSO et GA

L’analyse comparative du tableau 3.1 met en évidence la supériorité du SLPSO par rapport au PSO et au GA. Sur le plan de l'efficacité globale, le SLPSO atteint le coût de chemin (J) le plus bas avec une valeur de 8,173, contre 11,859 pour le GA. Cette performance s'explique par son mécanisme d'apprentissage adaptatif qui permet une exploration plus efficace de l'espace de recherche, évitant ainsi les optima locaux où le PSO standard reste souvent bloqué.

En observant les trajectoires sur la Figure 3.3, on constate des stratégies de navigation très différentes. Le PSO adopte une approche extrêmement prudente avec un risque de collision nul (R=0), mais au prix d'une trajectoire très longue (21,166 m) contournant largement les obstacles. À l'opposé, le SLPSO trouve un compromis optimal en acceptant un risque quasi négligeable (0,003) pour réduire la distance à 18,457 m, soit le chemin le plus court des trois méthodes. Le GA, quant à lui, peine à lisser sa trajectoire, résultant en une distance beaucoup plus longue de 26,821 m et une fluidité médiocre.

Les courbes de convergence montrent que le SLPSO stabilise sa solution plus efficacement au fil des itérations. Tandis que le GA stagne rapidement à un coût élevé, le SLPSO continue d'affiner la trajectoire, un atout crucial dans cet environnement complexe de 9 obstacles.

Le PSO se distingue par une stabilisation quasi instantanée dès les premières itérations. Cependant, cette stagnation rapide à un coût de 8,74 révèle une convergence prématurée : l'algorithme reste piégé dans un optimum local conservateur privilégiant la sécurité absolue au détriment de l'optimisation de la distance. Le SLPSO, à l'inverse, utilise son apprentissage pour affiner continuellement la solution, tout en restant le plus rapide avec un temps de calcul de 307,7 s.

Enfin, l'analyse de la fluidité (S) révèle que le PSO génère le chemin le plus "lisse" (2,707 rad), simplement parce qu'il évite les zones denses en obstacles. Toutefois, pour une application réelle en robotique mobile où l'économie d'énergie et la rapidité sont prioritaires, le SLPSO demeure le meilleur choix. Il offre une trajectoire directe et rapide, tout en maintenant une fluidité acceptable par rapport aux mouvements erratiques du GA (11,279 rad).

- **Nombre de waypoints dans le chemin (D = 5)**

Les résultats obtenus lors de la simulation, pour un nombre de points de passage fixé à D = 5, sont présentés ci-dessous. Le tableau 3.2 présente une synthèse comparative de ces performances : la longueur du chemin (L), le risque de collision (R), la fluidité (S), le coût global (J) et le temps de calcul. Les chemins des robots et les courbes de convergence obtenus après l’implémentation des algorithmes SLPSO, PSO et GA sont présentés respectivement dans les figures 3.5 et 3.6.

Table 3.2 : Synthèse comparative des performances (D = 20)

Algorithme	L(P)	R(X)	S(P)	Cout J	Temps de calcul
SLPSO	14.310	0.051	0.466	5.796	307.7
PSO	21.166	0	2.707	8.737	326.9
GA	21.404	0.011	12.463	9.814	338.8

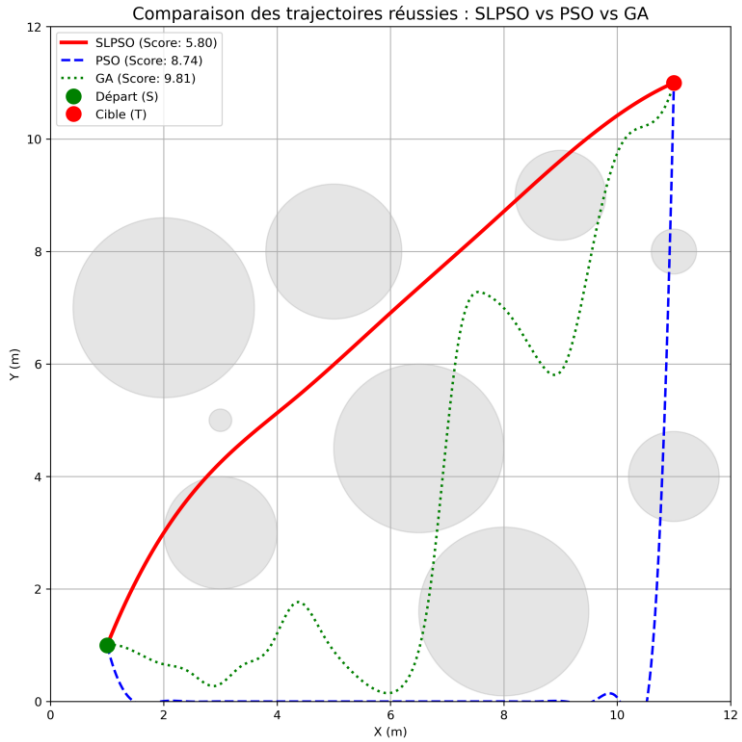

Figure 3-5 : Chemins planifiés par les algorithmes SLPSO, PSO et GA (D = 20)

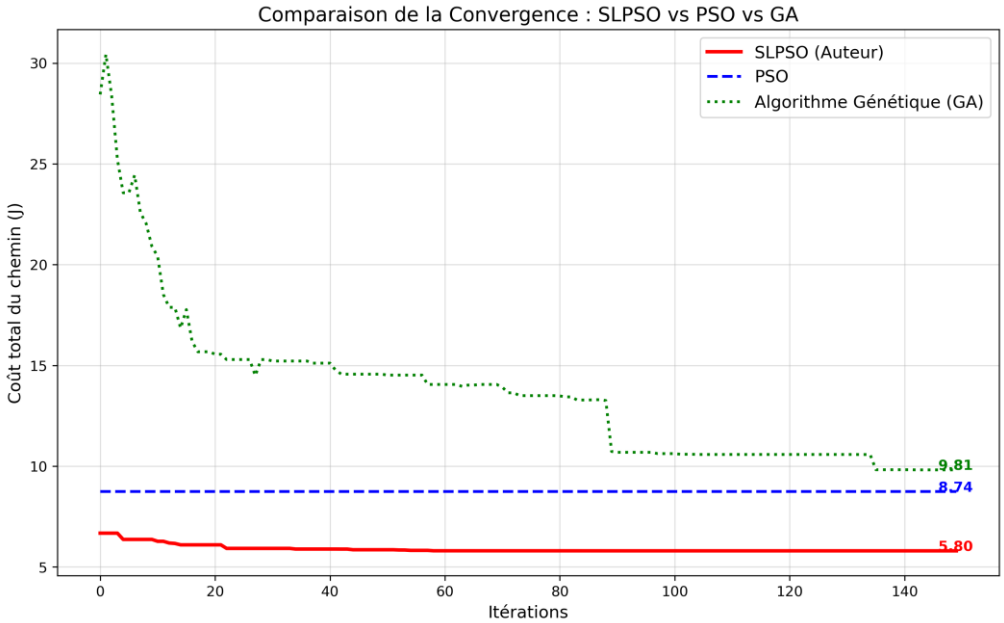

Figure 3-6 : Courbes de convergence du coût global (J) pour SLPSO, PSO et GA

Sur le plan de l'efficacité globale, l'analyse des performances du Tableau 3.2 démontre la nette supériorité de l'algorithme SLPSO. Avec un coût global (J) de 5,796, il surpasse largement le PSO (8,737) et le GA (9,814). Cette efficacité s'explique par sa capacité à trouver le chemin le plus court (14,310 m) tout en maintenant une excellente fluidité (0,466 rad), prouvant que son mécanisme d'apprentissage adaptatif évite efficacement les optima locaux.

Sur le plan de la sécurité et de la trajectoire, les comportements sont divergents. Le PSO adopte une stratégie ultra-prudente avec un risque nul (R=0), mais au prix d'une distance très longue (21,166 m) car il contourne largement les obstacles. À l'opposé, le SLPSO optimise le trajet en acceptant un risque infime (0,051) pour passer entre les obstacles, tandis que le GA fournit la solution la moins efficace avec une trajectoire saccadée et peu fluide.

Les courbes de convergence illustrent parfaitement la robustesse du SLPSO. Alors que le PSO stagne quasi instantanément (indiquant une convergence prématurée vers un optimum local), le SLPSO continue d'affiner sa solution durant les premières itérations pour atteindre un palier optimal stable. Le GA, bien qu'il progresse de manière constante, termine sa course avec un coût final très éloigné des performances des deux autres méthodes.

Enfin, le temps de calcul confirme que le SLPSO est l'algorithme le plus rapide en termes de calcul un temps estime à 307,7 s, prouvant que son mécanisme d'apprentissage adaptatif optimise la recherche sans alourdir la charge computationnelle.

- **Nombre de waypoints dans le chemin (D = 30)**

Les résultats obtenus lors de la simulation, pour un nombre de points de passage fixé à D = 15, sont présentés ci-dessous. Le tableau 3.3 présente une synthèse comparative de ces performances : la longueur du chemin (L), le risque de collision (R), la fluidité (S), le coût global (J) et le temps de calcul. Les chemins des robots et les courbes de convergence obtenus après l'implémentation des algorithmes SLPSO, PSO et GA sont présentés respectivement dans les figures 3.7 et 3.8.

Table 3.3 : Synthèse comparative des performances (D = 30)

Algorithme	**L(P)**	**R(X)**	**S(P)**	**Cout J**	**Temps de calcul**
SLPSO	19.334	0.042	3.688	8.123	307.7
PSO	20.093	0.046	10.741	9.135	326.9
GA	28.295	0.122	15.932	12.972	338.8

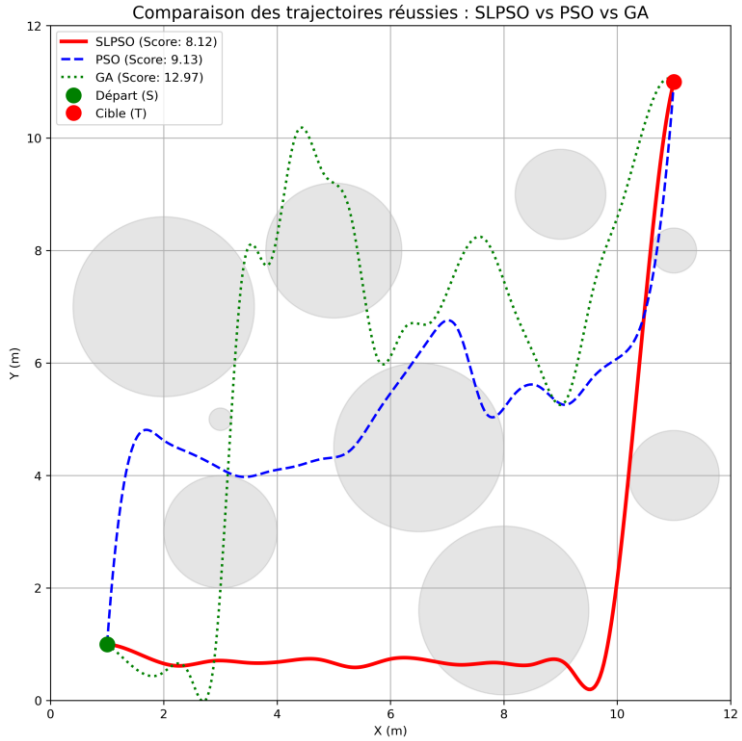

Figure 3-7 : Chemins planifiés par les algorithmes SLPSO, PSO et GA (D = 30)

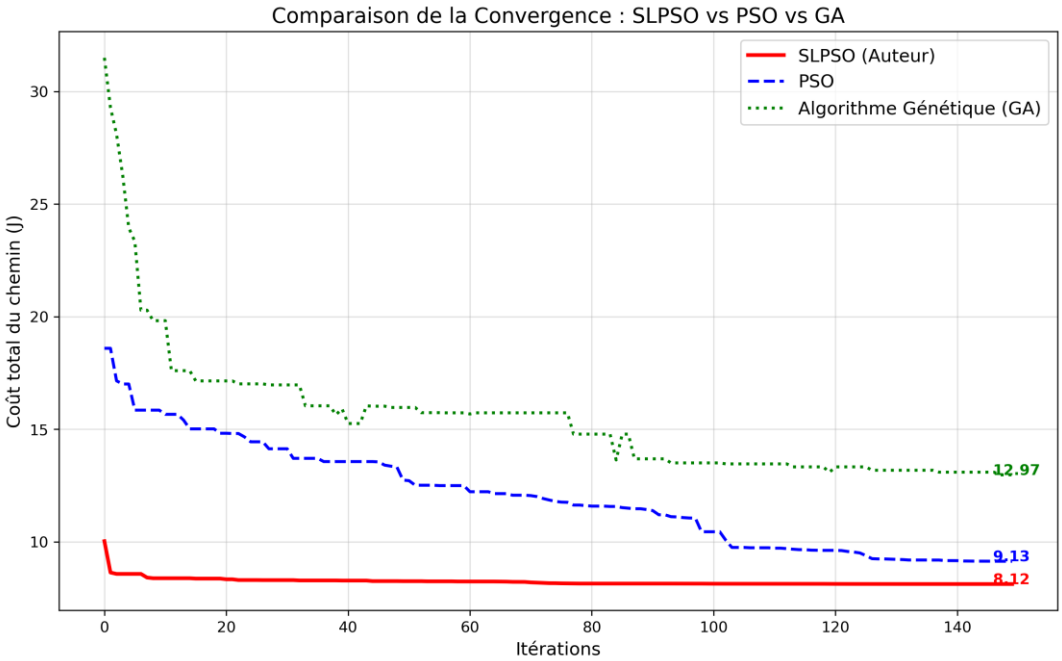

Figure 3-8 : Courbes de convergence du coût global (J) pour SLPSO, PSO et GA

L'augmentation de la dimension à D = 30 confirme la robustesse de l'algorithme SLPSO, qui maintient le coût global (J) le plus bas (8,123) face au PSO (9,135) et au GA (12,972). Cette performance s'explique par sa capacité à optimiser simultanément la distance et la sécurité. En effet, le SLPSO génère le chemin le plus court (19,334 m) tout en présentant le risque de

collision le plus faible (0,042), validant l'efficacité de son mécanisme d'apprentissage adaptatif dans des espaces de recherche plus complexes.

Sur le plan de la fluidité et de la trajectoire, le contraste est marqué. Alors que le SLPSO conserve une trajectoire fluide avec un score S de 3,688 rad, le PSO standard voit sa fluidité se dégrader fortement (10,741 rad) avec des oscillations visibles sur le graphique. Le GA reste l'algorithme le moins performant, produisant un chemin très long (28,295 m) et extrêmement saccadé (15,932 rad), ce qui le rend peu adapté à une navigation robotique réelle nécessitant une économie d'énergie.

Enfin, l'étude de la convergence montre que le SLPSO atteint sa solution optimale très rapidement, dès les premières itérations, et reste stable jusqu'à la fin des 150 itérations. À l'inverse, le PSO et le GA montrent une progression beaucoup plus lente et irrégulière, sans jamais égaler la précision du SLPSO. Avec un temps de calcul de 307,7 s, le SLPSO prouve qu'une densité de waypoints plus élevée (D=30) n'altère pas sa rapidité, offrant le meilleur compromis entre précision de trajectoire et efficacité computationnelle.

L'analyse comparative des performances pour les différentes configurations de waypoints (D=5, D=20 et D=30) confirme la supériorité constante de l'algorithme SLPSO sur le PSO standard et l'algorithme génétique (GA). Quel que soit le nombre de points de passage, le SLPSO obtient systématiquement le coût global (J) le plus bas, atteignant son efficacité maximale à D=20 avec un coût de 5,796. Cette performance repose sur une optimisation rigoureuse de la distance, avec des trajets plus directs que ses concurrents, et une fluidité nettement supérieure, particulièrement visible à D=30 où le SLPSO conserve une trajectoire stable (3,688 rad) tandis que le PSO et le GA génèrent des mouvements erratiques et saccadés. Enfin, le SLPSO se distingue par sa rapidité d'exécution invariable (307,7 s) et sa capacité à éviter les optima locaux conservateurs du PSO, prouvant qu'il offre le compromis idéal entre sécurité, économie d'énergie et rapidité pour la navigation robotique en environnement complexe.

CHAPITRE QUATRE : CONCLUSION

Cette preuve de concept avait pour objectif principal d'évaluer la pertinence de l'algorithme d'optimisation par essaim de particules à apprentissage auto-adaptatif (SLPSO) pour la planification de trajectoire d'un robot mobile en environnement encombré, et de comparer ses performances à celles du PSO et de l'algorithme génétique (GA). L'étude s'inscrivait dans une démarche de reproduction scientifique du travail de Li et Chou, avec une implémentation en Python et une analyse critique des résultats obtenus.

Les résultats expérimentaux issus des simulations ont montré de manière cohérente et reproductible que le SLPSO surpasse systématiquement le PSO et le GA, quel que soit le nombre de points de passage considérés (D = 5, 20 et 30). Grâce à son mécanisme d'apprentissage adaptatif fondé sur la sélection dynamique de stratégies, le SLPSO parvient à maintenir un équilibre efficace entre exploration globale et exploitation locale. Cette capacité lui permet d'éviter la convergence prématurée observée chez le PSO standard et de produire des trajectoires plus courtes, plus fluides et mieux optimisées que celles générées par le GA.

L'analyse détaillée des métriques a mis en évidence que le SLPSO offre le meilleur compromis entre distance parcourue, sécurité vis-à-vis des obstacles et douceur de trajectoire. De plus, malgré une complexité algorithmique plus élevée sur le plan conceptuel, le temps de calcul du SLPSO (307,7 s) est resté stable et inférieur à celui des deux autres méthodes, démontrant que l'adaptation intelligente des stratégies ne dégrade pas l'efficacité computationnelle globale.

Au-delà de la comparaison algorithmique, ce travail a également permis de confirmer la reproductibilité globale des résultats de l'article de référence, tout en mettant en lumière certaines hypothèses, notamment le choix des paramètres, la modélisation simplifiée de l'environnement. Cette PoC valide l'intérêt du SLPSO comme méthode robuste et performante pour la planification de trajectoire de robots mobiles en environnement complexe. Les perspectives de travaux futurs incluent l'extension à des environnements dynamiques, l'intégration de contraintes cinématiques et énergétiques réalistes, ainsi qu'une validation sur robot réel à l'aide de frameworks tels que ROS. Ces évolutions permettraient de renforcer la portée applicative du SLPSO et d'envisager son utilisation dans des systèmes robotiques autonomes du monde réel.

References

[1] G. Li and W. Chou, “Path planning for mobile robot using self-adaptive learning particle swarm optimization,” vol. 61, no. May, pp. 1–18, 2018, doi: 10.1007/s11432-016-9115-2.

[2] B. Liu, S. Jin, Y. Li, Z. Wang, D. Zhao, and W. Ge, “An Asynchronous Genetic Algorithm for Multi-agent Path Planning Inspired by Biomimicry,” *J. Bionic Eng.*, vol. 22, no. 2, pp. 851–865, 2025, doi: 10.1007/s42235-024-00637-w.

[3] C.-C. Tsai, H.-C. Huang, and C.-K. Chan, “Parallel Elite Genetic Algorithm and Its Application to Global Path Planning for Autonomous Robot Navigation,” *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4813–4821, 2011, doi: 10.1109/TIE.2011.2109332.

[4] P. Raja and S. Pugazhenthi, “ON-LINE PATH PLANNING FOR MOBILE ROBOTS IN DYNAMIC ENVIRONMENTS,” *Neural Netw. World*, vol. 22, pp. 67–83, 2012, [Online]. Available: https://api.semanticscholar.org/CorpusID:16662669

[5] M. Saska, M. Macas, L. Preucil, and L. Lhotska, “Robot Path Planning using Particle Swarm Optimization of Ferguson Splines,” in *2006 IEEE Conference on Emerging Technologies and Factory Automation*, 2006, pp. 833–839. doi: 10.1109/ETFA.2006.355416.