République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mohamed El-Bachir El-Ibrahimi

Bordj Bou Arreridj

Faculté Mathématique & Informatique

Département recherche opérationnelle

Mémoire de fin d'étude

En vue de l'obtention du diplôme de

Master en mathématique

Spécialité : méthodes et outils pour la recherche opérationnelle

Thème :

Etude comparative entre l'EO , PSO et FA

Présenté par :

- Benyoucef Hayet
- Harchaou Khalissa

Devant le jury composé de

Mr. Maza Sofiane

Mr.Zaouech Djafar

Rapporteur : Dr. Attia Abdelouahab

Année universitaire : 2020-2021

Remerciements

*On tiens à remercier tout d'abord **ELLAH** le tout puissant de nous avoir donné la foi et de nous avoir permis d'en arriver là.*

Nos remerciements vont également à nos parents, nos frères et sœurs de tous les sacrifices qu'ils ont consentis pour nous permettre de suivre nos études dans les meilleures conditions possibles et n'avoir jamais cessez de m'encourager tout au long de nos années d'étude.

*On tiens à exprimer notre reconnaissance et nos vifremerciements à notre encadreur **Dr attia abdelouahab**, pour sa précieuse collaboration, ses encouragements et son enthousiasme afin de nous guider durant la réalisation de ce projet.*

Nos remerciements vont naturellement à chalabi nour el houda et à toute personne ayant donnée un plus pour la réalisation de ce travail de près ou de loin , notamant et sans oublier d'adresser nos sincères remerciements au tous les enseignants de la département de l'informatique.

Dédicaces

Avec l'expression de ma reconnaissance, je dédie ce modeste travail à ceux qui, quels que soient les termes embrassés, je n'arriverais jamais à leur exprimer mon amour sincère.

A celui qui vive au plus profond de mon cœur, et était derrière chaque pas que je fais, à mon support dans ma vie, qui m'a appris m'a supporté et ma dirigé vers la gloire, à celui qui m'a appris à me tenir debout face aux difficultés : Mon cher père

A celle qui m'a donné l'espoir, a celle qui m'a donné sa vie, à celle que m'a arrosé de tendresse et d'espoirs, à la source d'amour, à celle qui a éclairé la voie pour moi et a été pour moi l'idéal à qui si je lui donne ma vie ne lui suffira pas : Ma aimée chère mère

A ma seule chère sœur, qui m'a toujours soutenu et encouragé tout au long de mes études

A l'homme qui m'a choisi pour me donner son nom à jamais:
Mon mari chéri

A toutes les personnes de ma grande famille, source d'espoir et de motivation

A tous ceux qui porte le nom: HARCHAOU, ABBACI, MAAOUI

A celle qui a partagé avec moi ce modeste travail, mon binôme "Hayat"

A celui que m'a beaucoup aidé et soutenu dans ce travail,
Mon amie Houda

A tous mes amis

A vous cher lecteur

KHALISSA

Je dédie ce travail à …

Mes chers parents, que nulle dédicace ne peut exprimer mes

Sincères

Sentiments, pour leur patience illimitée

Leur encouragement contenu, leur aide, en témoignage de mon

Profond amour et respect pour ses grands sacrifices.

Ma chère Houda, que je considère comme ma sœur, qui m'a aidée et m'a beaucoup soutenue dans ce travail.

Mes chères sœurs sihem et malika

Pour leur grand amour et leur soutien qu'ils trouvent ici

L'expression de ma haute gratitude.

Mes chères amies

Mon cher grand-père (Mohamed)

Et à toute ma famille et à tous ceux que j'aime

HAYET

Sommaire

Sommaire

Chapitre II : L'état de l'art

Chapitre III : Algorithme d'Optimisateur d'Equilibre

Liste des figures

Liste des tableaux

Liste des abréviations

OPE : Optimisation par Essaim de Particules

PSO : Particle Swarn Optimization (ou OPE en français)

FA : Firefly Algorithm (ou Algorithme de lucioles en français)

PSOG : PSO Globale

GA : l'algorithme génétique

GPU : Graphics Processing Unit

SC-BR-APSO : Subtractive Clustering based Boundary Restricted Adaptive Particle Swarm Optimization.

MPSO : Modified Particle Swarm Optimization

FCMPSO : Fuzzy C-Means and Particle Swarm Optimization

PSCS : Particle Swarm Contour Search

DG : la génération distribuée

PV : Procès-Verbal

PQ : puissance constante

FA-BP : Protéine de liaison aux acides gras

PWM : pulse width modulation

EOA : Equilibrium Optimizer Algorithm (Algorithme d'optimisation d'équilibre en français)

EO : Equilibrium Optimizer

Introduction générale

Dans ces dernières années, une évolution croissante du nombre d'études menées sur les animaux vivants en groupe ou en société et plus particulièrement les insectes sociaux. Ces études dans la théorie de l'auto-organisation ont inspiré un grand nombre des chercheurs pour développer une nouvelle approche appelé les métaheuristiques.

Les métaheuristiques forment un ensemble de méthodes utilisées en recherche opérationnelle et en intelligence artificielle pour résoudre des problèmes d'optimisation réputés difficiles. Résoudre un problème d'optimisation combinatoire. Utilisées dans de nombreux domaines, ces méthodes présentent l'avantage d'être généralement efficaces, sans pour autant que l'utilisateur ait à modifier la structure de base de l'algorithme qu'il utilise. Parmi celles-ci:

• Optimisation par essaim de particules (PSO) : Est inspirée du comportement social des essaims (les oiseaux migrateurs ou les poissons). L'essaim particulaire correspond à une population d'agents appelés « particules ». Chaque particule, est modélisée comme une solution potentielle au problème d'optimisation parcourt l'espace de recherche en quête de l'optimum global et elle est caractérisée par sa position et son vecteur de changement de position (ou vélocité).

• Algorithme de luciole (FA) : Elle est inspirée du comportement clignotant des lucioles dans la nature, où les lucioles lumineuses s'attirent entre elles. L'algorithme FA est un algorithme à une population de solutions (lucioles) basé sur deux processus importants qui sont l'intensité lumineuse et l'attractivité des lucioles.

Est-ce-qu'il y'a un autre algorithme plus effective que PSO et FA ?

L'équilibre est un état dans lequel des forces opposées sont équilibrées. Les modèles d'optimisation et d'équilibre sont couramment utilisés en génie et en science de l'environnement. Cela permet de résoudre différents types de problèmes d'équilibre, tels que les problèmes d'optimisation multi-objectifs, etc... L'optimiseur d'équilibre de chaque solution de particule avec sa position de concentration agit comme un agent de recherche. Les agents de recherche mettent à jour aléatoirement leur concentration par rapport au meilleur solutions à ce jour, à savoir les candidats à l'équilibre, pour finalement atteindre l'état d'équilibre du résultat optimal. La recherche comprend des propriétés de stabilité et d'approximation, ainsi que la conception et l'évaluation d'algorithmes efficaces sur le plan informatique. Par conséquent, nous développons un nouvel algorithme nommé Equilibrium Optimizer (EO).

Outre l'introduction, ce travail est organisé en trois chapitres répartis comme suit :

• Le premier chapitre intitulé "Métaheuristique" présente les méthodes bio-inspirées: l'algorithme d'optimisation par essaim de particules et l'algorithme des lucioles.

• Le second chapitre porte le titre "L'état de l'art" qui expose les différentes applications de PSO et FA.

• Le dernier chapitre intitulé "L'algorithme d'optimisation d'équilibre" développe les notions essentielles de l'algorithme d'optimisation d'équilibre et les différentes fonctions de benchmark qui applique sur cet algorithme.

Comme on a commencé par la présent introduction Nous terminons avec une conclusion générale qui récapitulons notre contribution et proposer des perspectives sur la base des travaux effectués dans notre mémoire.

Notre objectif dans ce travail est de trouver l'algorithme qui nous donne la meilleure solution possible.

CHAPITRE I :
Métaheuristique

I.1.Introduction

La métaheuristique visant à résoudre les problèmes d'optimisation combinatoire. Ce type de problème apparait simple à résoudre du fait que nous connaissons presque souvent comment calculer les déférentes solutions possibles. La simulation de chaque solution peut mener à la sélection de la meilleure (l'optimale). L'ensemble de ces solutions est appelé l'espace de recherche. Ce qui fait, le problème d'optimisation consiste à trouver la meilleure solution parmi cet espace. Pouvoir tester toutes les solutions possibles mène à ce que le problème ne sera pas difficile à résoudre.

Cependant, lorsque l'espace de recherche devient très large, l'énumération de toutes les solutions ne serait pas possible à cause de la très longue durée que le processus de recherche prendra. Les métaheuristiques représentent l'ensemble des méthodes ayant la possibilité de résoudre ce type de problèmes.

Parmi les algorithmes bio-inspirés les plus connus on trouve les algorithmes évolutifs, inspirés de l'évolution biologique. Un autre exemple d'algorithmes bio-inspirés et les réseaux de neurones, o*ù* leur principe de fonctionnement est inspiré du mécanisme de algorithmes bio-inspirés, qui est l'algorithme d'optimisation par essaims particulaires (OEP), o*ù* particular Swarm Optimisation (PSO) en anglais. Développé par by ***Eberhart*** et ***Kennedy*** en ***1995***, le principe de l'algorithme PSO se base sur le comportement collectif des essaims (oiseaux, poissons) dans la nature.

Ainsi il y'a un autre algorithme, qui est l'algorithme des lucioles ou firefly algorithm (FA) en anglais. Introduite par ***Dr Xin-She Yan*** à l'université Cambridge en **2007**. L'algorithme est basé sur le principe d'attraction entre les lucioles et simule le comportement d'un essaim de lucioles dans la nature.

I.2.L'optimisation

L'optimisation est une branche des mathématiques cherchant à modéliser, à analyser et à résoudre analytiquement ou numériquement les problèmes qui consistent à minimiser ou maximiser une fonction sur un ensemble. L'optimisation est une discipline en plein essor qui entre en jeu dans beaucoup de domaines, comme dans la conception de circuits électroniques, la recherche opérationnelle, la biologie, mais aussi pour répondre aux besoins croissants des secteurs économique et industriel (maximisation des performances, minimisation des couts).

I.3.L'optimisation combinatoire

L'optimisation combinatoire est une branche de l'optimisation en mathématiques appliquées et en informatique, également liée à la recherche opérationnelle, l'algorithmique et la théorie de la complexité. On parle également d'optimisation discrète.

Un problème d'optimisation combinatoire consiste à trouver la meilleure solution dans un ensemble discret de solutions appelé ensemble des solutions réalisables. En général, cet ensemble est fini mais de cardinalité très grande et il est décrit de manière implicite, c'est-à-dire par une liste de contraintes que doivent satisfaire les solutions réalisables. Pour définir la notion de meilleure solution, une fonction, dite fonction objectif, est introduite. Pour chaque solution, elle renvoie un réel, et la meilleure solution ou solution optimale est celle qui minimise ou maximise la fonction objectif.

I.4.Les méthodes d'optimisation combinatoire

I.4.1.Méthode exactes

Dans Les méthodes exactes toutes les solutions de l'espace de recherche sont énumérées implicitement en utilisant des mécanismes qui détectent des échecs (calcul de bornes). Grâce à Ces méthodes on peut trouver des solutions optimales. Mais ces méthodes s'avèrent malgré les progrès réalisés plutôt inefficaces à mesure que la taille du problème devient importante.

Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable et rencontrent généralement des difficultés face aux applications de taille importante.

I.4.2.Méthodes approchées

Une méthode approchée (incomplètes) est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objective en un temps raisonnable, mais sans garantie d'optimalité. L'avantage principal de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles. Ils peuvent être classés en deux catégories : les heuristiques et les métaheuristiques.

I.5.Heuristique

Le terme heuristique vient du verbe grec ***Heuriskein*** signifiant trouver. Une heuristique permet de trouver une "bonne" solution, en un temps raisonnable, seulement elle n'offre aucune garantie sur l'optimalité de la solution trouvée.

I.6.Métaheuristique

Méta est un préfixe signifiant « au-delà », « plus que ça », « a un niveau supérieur », lorsqu'il est rajout à l'heuristique il donne Métaheuristique qui veut dire trouver au-delà, ou trouver plus que ça. En ***1996, I.H. Osman et G. Laporte*** définissaient la métaheuristique comme « un processus itératif qui subordonne et qui guide une heuristique, en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque-optimales » **[1].**

Les méthodes dites méta heuristiques sont des méthodes générales, des heuristiques polyvalentes applicables sur une grande gamme de problèmes. Elles peuvent construire une alternative aux méthodes heuristiques lorsqu'on ne connaît pas l'heuristique spécifique à un problème donné.

Les métaheuristiques groupées en deux classes : des métaheuristiques à base de solution unique (recherche de tabou, le recuit simulé, …) et des métaheuristiques à base de population de solutions (FA, PSO, …).

I.6.1.Les Métaheuristiques à solution unique

Les méthodes métaheuristiques à solution unique sont toutes basées sur un algorithme de recherche du voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage.

Parmi les méthodes méta heuristiques à solution unique en trouve Algorithme du recuit simulé, Algorithme de recherche tabou et l'Algorithme de descente.

I.6.2.Les métaheuristique à population de solution

I.6.2.1.L'Optimisation par Essaim Particulaire

I.6.2.1.1.Histoirique

L'optimisation par essaim de particules (PSO) a été développée en **1995** par le ***Dr.EBERHAT*** (ingénieur en électricité) et le ***Dr. KENNEDY*** (socio psychologue)**[2] [3].**

Cette méthode est inspirée de la biologie pour résoudre des problèmes d'optimisation. L'origine de cette méthode vient des observations du comportement social des animaux qui vivent en essaim tels que les oiseaux migrateurs et les bancs de poissons, Le PSO s'inspire du modèle développé par ***Craig Reynolds*** **[4]** pour simuler le déplacement grégaire de certains animaux (troupeaux de bovins, volées d'oiseaux…).

I.6.2.1.2.Principe générale

L'optimisation par essaim de particules repose sur un ensemble d'individus originellement disposés de façon aléatoire et homogène, que nous appellerons dès lors des particules, qui se déplacent dans l'espace de recherche et constituent chacune une solution potentielle. Chaque particule dispose d'une mémoire concernant sa meilleure solution visitée ainsi que la capacité de communiquer avec les particules constituant son entourage. A partir de ces informations, la particule va suivre une tendance faite, d'une part, de sa volonté à retourner vers sa solution optimale, et d'autre part, de son mimétisme par rapport aux solutions trouvées dans son voisinage. A partir des optimums locaux et empiriques, l'ensemble des particules va normalement converger vers la solution optimale globale du problème traité**,** voir **la Figure I.1**

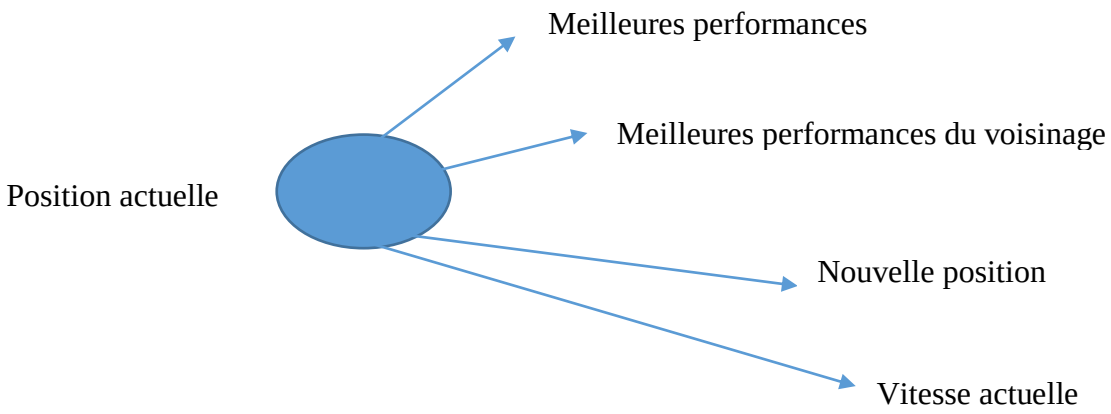


Figure I.1 : Eléments du comportement des particules d'un essaim.

- ***Modèle de Reynolds (birds)***

On commence par expliquer le modèle de simulation de déplacement d'un essaim d'oiseaux de ***Craig Reynolds*** **[4]** qui représente les oiseux artificiels comme des 'birds', chaque bird se déplace aléatoirement en suivant trois règles simples comme le montre la **figure I.2** pour maintenir la stabilité de l'essaim :

a) Séparation : les boids gardent une certaine distance entre eux pour éviter les collisions.

b) Cohésion : les boids sont attirés vers la position moyenne du groupe.

c) Alignement : les boids suivent le même chemin que leurs voisins.

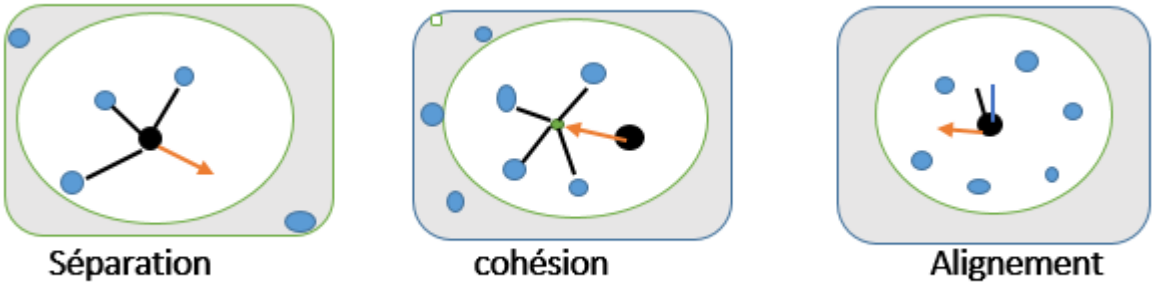


Figure I.2 : les trois règles de Reynolds

- ***La notion du voisinage***

Le voisinage d'une particule est le sous-ensemble de particules de l'essaim avec lequel il a une communication directe. Ce réseau de rapports entre toutes les particules est connu comme la sociométrie, ou la topologie de l'essaim. Il existe deux principaux types de voisinage :

Les voisinages géographiques : les voisins sont considérés comme les particules les plus proches. Cependant, à chaque itération, les nouveaux voisins doivent être recalculés à partir d'une distance prédéfinie dans l'espace de recherche, c'est donc un voisinage dynamique

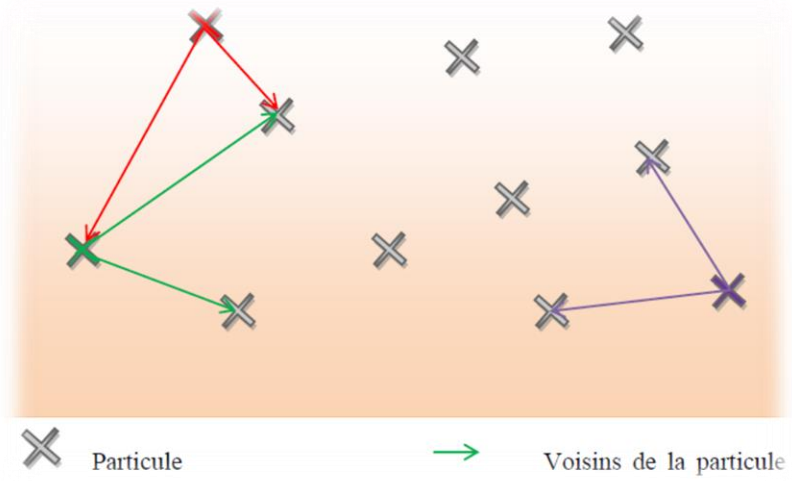


Figure I.3 *:* Un voisinage géographique **[5]**

Dans cette figure **I.3** le voisinage de la particule est composé des deux particules les plus proches.

Les voisinages sociaux : les voisins sont définis à l'initialisation et ne sont pas modifiés ensuite. C'est le voisinage le plus utilisé, pour plusieurs raisons :

- Il est plus simple à programmer.
- Il est moins coûteux en temps de calcul.
- En cas de convergence, un voisinage social tend à devenir un voisinage géographique.

Pour ce faire, on dispose (virtuellement) les particules en cercle puis, pour la particule étudiée, on inclut progressivement dans ses informatrices, d'abord elle-même, puis les plus proches à sa droite et à sa gauche, jusqu'à atteindre la taille voulue. On peut aussi choisir les informatrices au hasard

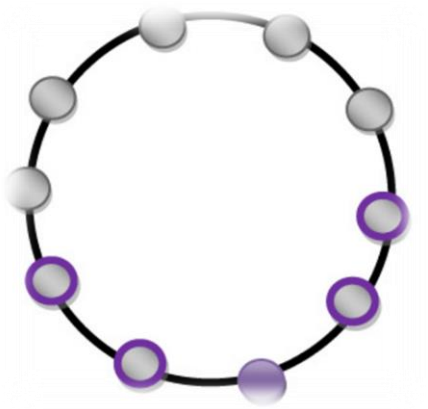


Figure I.4 *:* Un voisinage en cercle **[5]**

Dans cette figure I.4, la particule principale est en bas et ses informatrices correspondent au deux particules directement à sa droite et à sa gauche.

I.6.2.1.3.Formalisation

En réalité, il existe deux types d'algorithmes de PSO : le PSO discret, et le PSO continu, nous allons nous intéresser uniquement au PSO continu tel qu'il est défini dans les travaux de ***Michel Gourgand*** et ***Sylverin Kemmoé Tchomté*** **[6].**

Dans l'espace de recherche de dimension **D**, une particule ***i*** de l'essaim est représentée par son vecteur position et par son vecteur vitesse ; formulés ainsi :

$$\overrightarrow{x_i} = (x_{i1}, x_{i2}, \ldots, x_{iD}) \quad \text{Et} \quad \overrightarrow{v_i} = (v_{i1}, v_{i2}, \ldots, v_{iD})$$

L'appréciation de la qualité de sa position est arrêtée par la valeur de la fonction "objectif" en ce point. Il est indispensable que cette particule puisse mémoriser la meilleure position par laquelle elle est déjà passée, formulée comme suit :

$$\vec{p}best_i = (pbest_{i1}, pbest_{i2}, \ldots, pbest_{iD})$$

La meilleure position atteinte par les particules de l'essaim est formulée comme suit :

$$\vec{g}best = (gbest_1, gbest_2, \ldots, gbest_D)$$

La notion de **gbest (**global best), est calquée sur la version PSO Globale (PSOG) où toutes les particules de l'essaim sont issues de la particule ***i***.

Au commencement de l'algorithme, les particules de l'essaim sont initialisées de manière aléatoire /régulière dans l'espace de recherche. Par la suite, à chaque itération, les particules se déplacent, en fusionnant les trois composantes citées ci-dessus :

$\mathbf{w}v_{i,j}^{t}$ **:** représente la composante d'inertie du déplacement, où le paramètre **w** gère l'influence de la direction de déplacement sur le déplacement futur.

$c_1 r_{1i,j}^{t}\left[pbest_{i,j}^{t} - x_{i,j}^{t}\right]$ **:** représente la composante cognitive du déplacement, où le paramètre $c_1$gère le comportement cognitif de la particule.

$c_2 r_{2i,j}^{t}\left[gbest_{j}^{t} - x_{i,j}^{t}\right]$ **:** représente la composante sociale du déplacement où le paramètre c_2 gère l'aptitude sociale de la particule.

La fusion de ces trois composantes nous permet d'abord de dégager la vitesse qui elle-même nous offre la possibilité de calculer la position de la particule.

Ces deux variables sont obtenues par l'utilisation de l'équation **(1)**, successivement comme suit :

$$v_{i,j}^{t+1} = wv_{i,j}^{t} + c_1 r_{1i,j}^{t}\left[pbest_{i,j}^{t} - x_{i,j}^{t}\right] + c_2 r_{2i,j}^{t}\left[gbest_{j}^{t} - x_{i,j}^{t}\right], j\epsilon\{1, 2, D\} \quad (1)$$

$$v_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t} \qquad j\epsilon\{1, 2, \dots D\} \quad (2)$$

Où :

- w est une constante, appelée coefficient/pourcentage d'inertie.
- c_1, c_2 sont deux constantes, appelées coefficients/pourcentage d'accélération.
- r_1, r_2 sont deux nombres aléatoires tirés uniformément dans **[0, 1]**, et ce à chaque itération ***t*** et pour chaque dimension ***j***.

A la fin du déplacement des particules dans une itération donnée, les nouvelles positions sont évaluées et les deux vecteurs ***pbest*** et ***gbest*** sont réindexés, conformément aux deux équations **(3)** dans le cas d'une minimisation d'une fonction objective puis **(4)** dans une version globale de PSO, respectivement :

$$\vec{p}best_i\ (t+1) = \begin{cases} \vec{p}best_i(t) \ si\ f\left(\vec{x_i}\ (t+1)\right) \geq \vec{p}best_i(t) \\ \vec{x_i}(t+1)\ ,\ sinon \end{cases} \quad (3)$$

$$\vec{g}best\ (t+1) = arg \min_{\vec{p}best_i} f(\vec{p}best_i(t+1)) \qquad 1 \leq i \leq N \quad (4)$$

Où :

N représente le nombre de particules de l'essaim.

Cette procédure est représentée dans l'Algorithme1 (Algorithme d'Optimisation par Essaim Particulaire) **[7].**

Algorithme 1 :

1 Initialiser aléatoirement N particules : position et vitesse.
2 Evaluer les positions des particules
3 Pour chaque particule i, $\vec{p}best_i = \vec{x_i}$
4 Calculer $\vec{g}best$ selon (4)
5 Tant que le critère d'arrêt n'est pas satisfait faire

Déplacer les particules selon (1) et (2)
Evaluer les positions des particules
Mettre à jour $\vec{p}best_i$ et $\vec{g}best$ selon (3) et (4)

Fin.

Ledit algorithme, va servir comme référence pour l'implémentation de notre application. Nous schématisons l'algorithme PSO à l'aide d'un organigramme se trouvant dans la **Figure I.5**

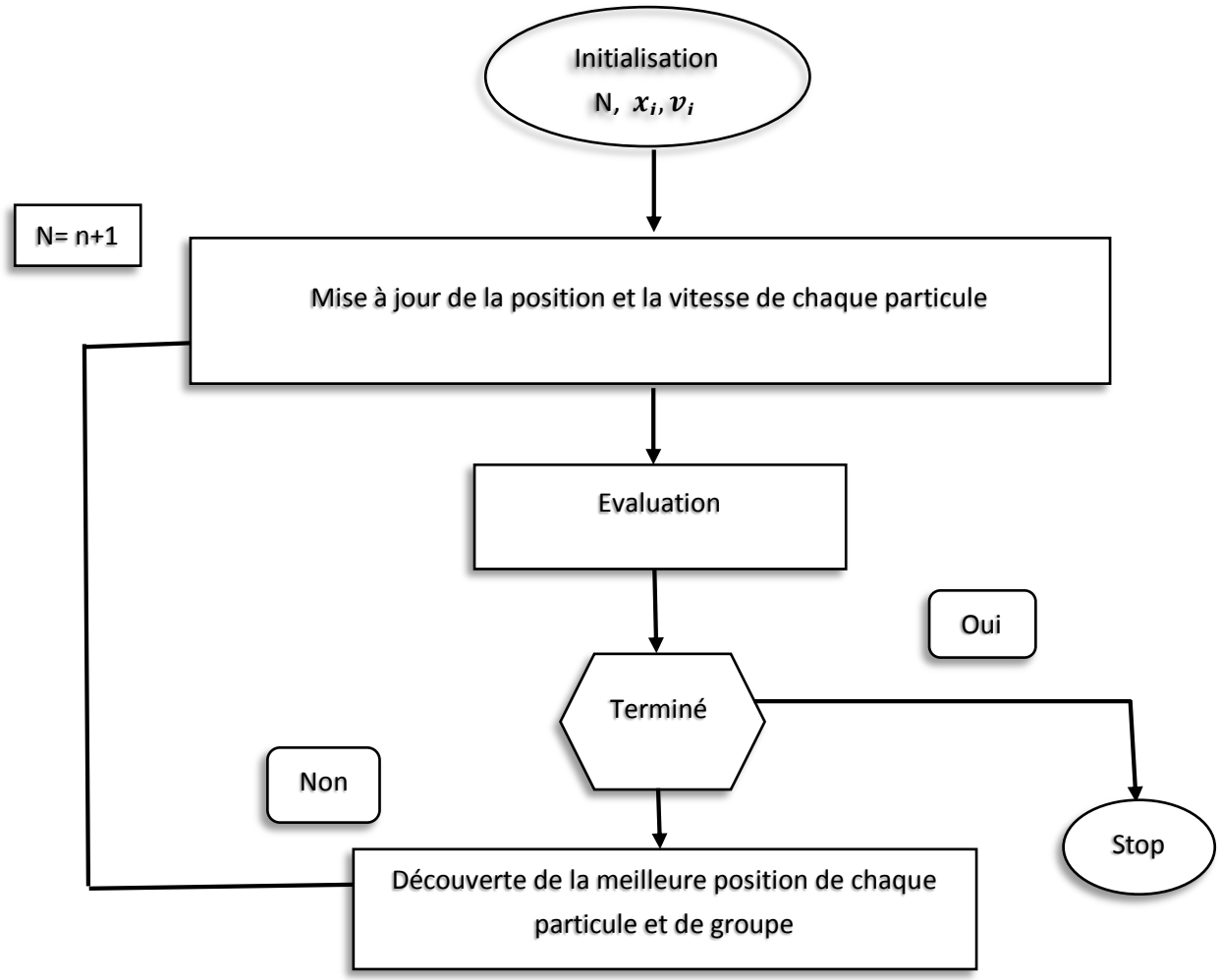


Figure I.5 : Organigramme de la méthode des essaims particulaires.

I.6.2.1.4.Configuration des paramètres

➢ Nombre de particules

Le nombre de particule utilisé pour la résolution du problème dépend essentiellement de deux facteurs, la taille de l'espace de recherche et le rapport entre les capacités de calcul de la machine et le temps maximum de recherche. Il n'y a pas de règle pour déterminer ce paramètre, faire de nombreux essais permet de se doter de l'expérience nécessaire à l'appréhension de ce paramètre. En général le choix se fait aléatoirement.

➢ Taille et topologie de voisinage

La topologie du voisinage constitue la structure du réseau social et définit avec qui chacune des particules va pouvoir communiquer. Il existe de nombreuses combinaisons dont les suivantes sont les plus utilisées **voir la Figure I.6**

a) Topologie en étoile : chaque particule est reliée à toutes les autres, l'optimum du voisinage est l'optimum global.

b) Topologie en anneau : chaque particule est reliée à n particules (**n = 3** en général), c'est la plus utilisée.

c) Topologie en rayon : les particules ne communiquent qu'avec une seule particule centrale.

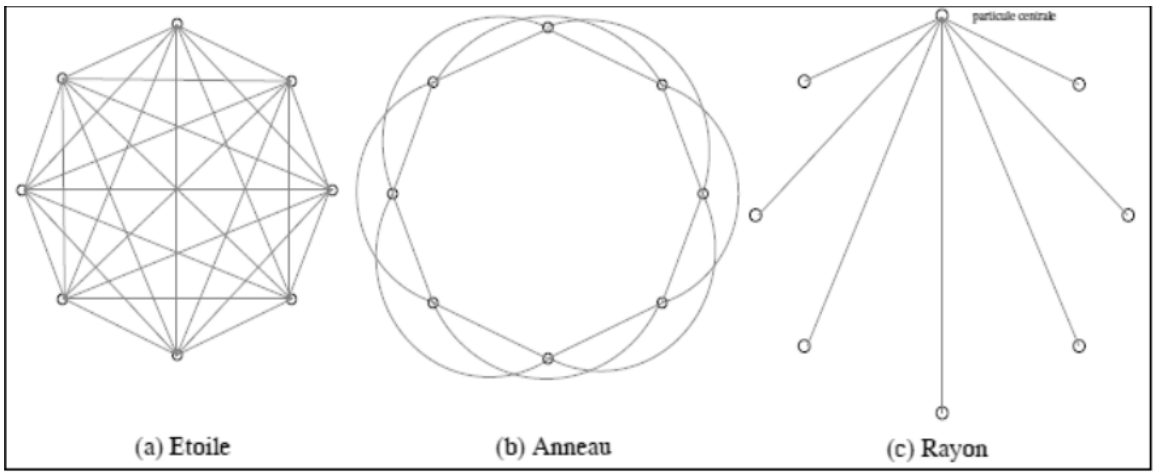


Figure I.6 : Topologies de voisinage

➢ Coefficients de confiance et coefficient d'inertie

- Les coefficients $c_1 r_1$ et $c_2 r_2$ de l'équation (1) sont appelés coefficients de confiance, ils permettent de pondérer les tendances des particules à suivre leur instinct de conservation ou leur panurgisme.
- Les coefficients r_1, r_2 sont des variables aléatoires évaluées à chaque itération suivant une loi uniforme sur le domaine [0 1].
- c_1, c_2 sont des constantes définies par la relation $c_1 + c_2 \leq 4$.

Le coefficient d'inertie appelé w dans la formule vue auparavant permet de définir la capacité d'exploration de chaque particule en vue d'améliorer la convergence de la méthode. Fixer ce paramètre

revient à trouver un compromis entre une exploration globale (**w> 1**) et une exploration locale (**w < 1**). Il représente l'instinct aventureux de la particule.

➢ *Vitesse maximale et coefficient de constriction*

Pour éviter que le déplacement des particules soit trop rapide, ce qui conduit à sortir de l'espace de recherche, nous pouvons introduire un nouveau paramètre ***Vmax***, qui permet de limiter la vitesse sur chaque dimension et ainsi de contrôler les particules et pour améliorer la convergence de l'algorithme.

Cependant on peut s'en passer si on utilise un coefficient de constriction ***k*** introduit par ***Maurice CLERC*** **[8]** et qui permet de resserrer l'hyper espace de recherche :

$$k = 1 - \frac{1}{p} + \frac{\sqrt{|p^2+4p|}}{2} \qquad (5)$$

Avec : $p = p_1 + p_2 > 4$

$$\overrightarrow{v_i}(t) = k.\left(\overrightarrow{v_i}(t-1) + p_1\left(\vec{x}pbest_i - \overrightarrow{x_i}(t)\right) + p_2\left(\vec{x}Vbest_i - \overrightarrow{x_i}(t)\right)\right) \qquad (6)$$

Les études de ***SHI*** et ***EBERHART*** indiquent que l'utilisation d'un coefficient de constriction donne généralement un meilleur taux de convergence sans avoir à fixer de vitesse maximale. Cependant, dans certains cas, le coefficient de constriction seul ne permet pas la convergence vers la solution optimale pour un nombre d'itérations donné. Pour résoudre ce problème, il peut être intéressant de fixer $\vec{v}$ **max =**$\vec{x}$ **max** en plus du coefficient de constriction, ce qui, selon les études de ***SHI*** et ***EBERHART***, permet d'améliorer les performances globales de l'algorithme.

➢ *Initialisation de l'essaim*

La position des particules ainsi que leur vitesse initiale doivent être initialisés aléatoirement selon une loi uniforme sur [0...1] cependant, en ce qui concerne la position des particules, il est préférable d'utiliser un générateur de séquence de SOBOL qui est plus pertinent dans la disposition homogène des particules dans un espace de dimension ***n***.

➢ *Critères d'arrêt*

Comme a été mentionné, la convergence vers la solution optimale globale n'est pas garantie dans tous les cas. Il est donc important de doter l'algorithme d'une porte de sortie en définissant un nombre maximum d'itérations (**nb**$Iter_{max}$).

Le programme s'arrête alors si et seulement si le nombre maximum d'itérations est atteint ou que la valeur du critère obtenue est acceptable pour l'utilisateur.

Ou bien l'algorithme doit alors s'exécuter tant que l'un des critères de convergence suivant n'a pas été atteint :

- **nb**$Iter_{max}$ a été atteinte.
- La variation de la vitesse est proche de **0**.
- La finesse de la solution est suffisante.

I.6.2.1.5.Algorithme de synthèse

L'algorithme suivant a été implémenté sous Matlab et exécuté sur un PC

[Initialisation]
Initialiser aléatoirement la population
…
[Traitement]
Répéter
Pour i=1 jusqu'à nb faire
  Si F ($\overrightarrow{x_i}$)> pbest$_i$ alors
    Pbest$_i$= F ($\overrightarrow{x_i}$)
    $\vec{x}$Pbest$_i$= $\overrightarrow{x_i}$
  Fin si
  Si F ($\overrightarrow{x_i}$) > vbest$_i$ alors
    Vbest$_i$=F ($\overrightarrow{x_i}$)
    $\vec{x}$Vbest$_i$=$\overrightarrow{x_i}$
  Fin si
Fin pour
Pour i= 1 to nb faire
  $\overrightarrow{v_i}(t) = k.\left(\overrightarrow{v_i}(t-1) + p_1\left(\vec{x}Pbest_i - \overrightarrow{x_i}(t)\right) + p_2\left(\vec{x}Vbest_i - \overrightarrow{x_i}(t)\right)\right)$
  $\overrightarrow{x_i}= \overrightarrow{x_i} + \overrightarrow{v_i}$
Fin pour
Jusqu'à ce que (le processus converge)

I.6.2.1.6.Les avantages de PSO

- Elle présente l'avantage d'être efficace sur une vaste gamme des problèmes.
- Facile à programmer dans n'importe quel langage de programmation.
- Simple a implémenté, adapté.
- Efficace sur un espace de recherche continu.

I.6.2.1.7.Les inconvénients de PSO

- **E**xplosion du système (si une particule utilise une vitesse très grande elle peut sortir de l'espace de recherche)
- L'initialisation aléatoire des positions, peut conduire l'algorithme à stagner dans un optimum local.
- Influencé par la topologie de voisinage choisie.
- Manque de diversité par rapport aux algorithmes évolutionnaires.

I.6.2.2. Algorithme de luciole (firefly algorithm)

I.6.2.2.1.Historique

L'algorithme de Luciole est une métaheuristique, bio-inspirée, introduite par ***Dr Xin-She Yan*** à l'université Cambridge en **2007[9].** Les lucioles (en anglais fireflies), sont des insectes qui appartiennent à la famille des abeilles. Les lucioles sont dotées d'ailes, leur nom vient du phénomène de bioluminescence. Les lucioles produisent une « lumière froide » sans fréquences infrarouges ou ultra-violets.

Cette lumière chimique est générée à partir de l'abdomen inférieur du corps de ces insectes. La couleur de cette lumière peut être jaune, verte ou rouge pale, avec une longueur d'onde entre 510 à 670 nanomètres.

Les femelles peuvent imiter les signaux lumineux des autres espèces afin d'attirer des mâles qu'elles les capturent et les dévorent. Les lucioles ont un mécanisme de type condensateur, qui se décharge lentement jusqu'à ce que certain seuil est atteint, ils libèrent l'énergie sous forme de lumière. Le phénomène se répète de façon cyclique.

Figure I.7: Les lucioles naturelles

I.6.2.2.2. Principe générale

L'algorithme de luciole (firefly algorithm) est basé sur le principe d'attraction entre les lucioles et simule le comportement d'un essaim de lucioles dans la nature.

Cet algorithme prend en considération les trois règles suivant :

1- Toutes les lucioles sont unisexe, ce qui fait l'attraction entre celles-ci n'est pas en fonction de leur sexe.

2- L'attraction est proportionnelle à leurs luminosités, donc pour deux lucioles, la moins lumineuse se déplacera vers la plus lumineuse. Si aucune luciole n'est lumineuse qu'une luciole particulière, cette dernière se déplacera aléatoirement.

3- La luminosité des lucioles est déterminée en fonction d'une fonction objective (à optimiser) **[9].**

- Le processus de l'algorithme de luciole commence avec l'initialisation de la population des lucioles et donc chaque luciole dans une population représente une solution candidate.

- La taille de la population détermine le nombre de solutions ou la taille de l'espace de recherche dont le but est d'orienter la recherche à la meilleure localisation**.**

- Dans l'étape suivante, chaque luciole est évaluée en fonction de leur condition physique (intensité lumineuse). À chaque nouvelle étape itérative, la luminosité et l'attirance de chaque luciole est calculée.

- La distance entre toutes lucioles peut être définie comme une distance cartésienne.

- La fonction de la distance développée est utilisée pour trouver la distance entre deux lucioles.

- La fonction de l'attractivité est définie en utilisant l'intensité lumineuse, la distance entre lucioles, et un coefficient d'absorption.

- Le mouvement de luciole est défini par une fonction de mouvement, en utilisant la position actuelle, l'attractivité et une marche aléatoire, après avoir comparé la luminosité de chaque luciole avec toutes les autres lucioles, les positions des lucioles sont mises à jour en se basant sur les règles de connaissances sur les lucioles et sur leurs voisins.

- Après le déplacement, la nouvelle luciole est évaluée et l'intensité de sa lumière est mise à jour. Pendant la boucle de comparaison en paire, la meilleure solution actuelle est la mise à jour d'une manière itérative. Le Processus de comparaison par pair est répété jusqu'à la satisfaction des critères de résiliation**.**

Les étapes principales de l'algorithme de luciole sont données dans l'organigramme suivant:

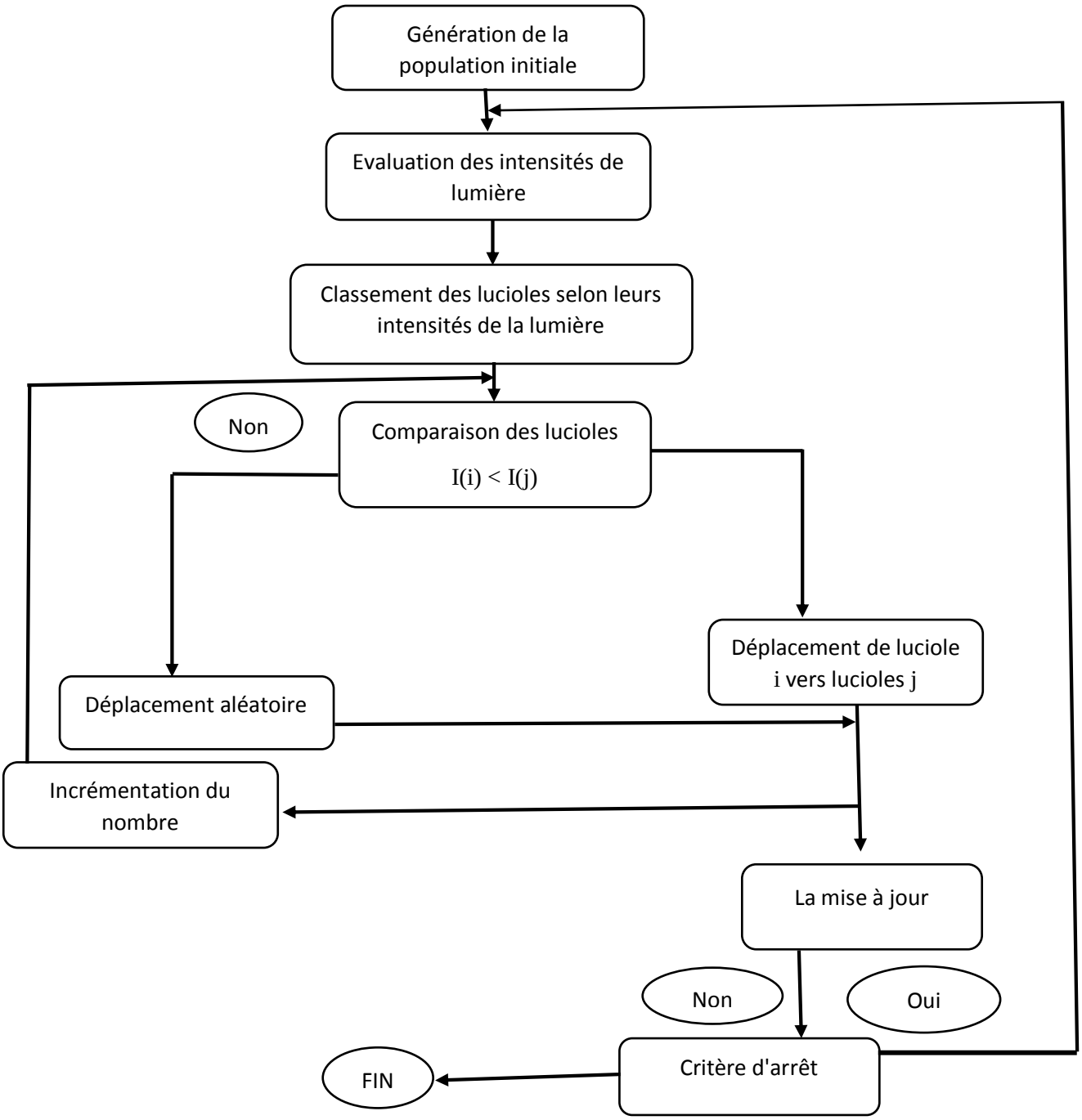


Figure I.8 : L'organigramme d'algorithme de luciole

I.6.2.2.3. Description de l'organigramme

- ***Génération de la population initiale***

Dans cette étape, l'algorithme de luciole "Firefly algorithm" génère une population initiale qui représente un ensemble de solutions possibles.

Cette population initiale est généralement générée aléatoirement, le choix de la population initiale est très important car il peut rendre plus ou moins rapide la convergence vers l'optimum global.

Dans le cas où l'on ne connaît rien sur le problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

En contrepartie, le choix d'une population trop élevée peut augmenter considérablement le temps de calcul, et si la taille de la population est trop petite, il y aura une convergence prématurée car l'algorithme n'a pas un grand échantillon de l'espace de recherche.

- ***Fonction d'évaluation***

La fonction d'évaluation (fitness) en terminologie ***anglo-saxome*** **[10]** ou de coût, attribut à chaque luciole une valeur numérique qui représente un coût de performance, elle est utilisée pour coder la luminosité des Fireflies. Grace à cette fonction l'algorithme converge vers l'optimum.

L'efficacité de l'algorithme en termes de pertinence de la solution et le temps de calcul dépend principalement de la fonction objective, pour cela elle doit définir les fonctions objectives de façon plus fidèle que possible.

Il existe deux types de fonction d'évaluation, soit mono critère ou multicritère :

** Une fonction d'évaluation mono critère*

Signifie que la fonction dépend d'une seule et même fonction objectif. La résolution de la fonction d'adaptation (fitness), dans ce cas est simple et ne pose généralement aucun problème.

** Une fonction d'évaluation multicritère*

Généralement, les problèmes d'optimisation doivent souvent satisfaire des objectifs multiples. Une méthode classique consiste à définir des fonctions objectifs élémentaires dont certains sont concurrents, traduisant chaque objectif à atteindre, et de les fusionner au sein d'une seule fonction.

- ***Classement***

Le classement s'effectue par rapport à la fonction objective, donc dans notre algorithme le classement se fait selon l'intensité de la lumière de chaque luciole.

Généralement le classement sert à déterminer le meilleur individu ou le mauvais individu, si la fonction objective cherche à maximiser les critères, le classement par ordre croissant et donc le meilleur est le maximum et le mauvais est le minimum, sinon si la fonction objective cherche à minimiser les critères, le classement se fait par ordre décroissant et donc le meilleur est le minimum et le mauvais est le maximum.

- ***Déplacement et mise à jour***

À chaque nouvelle étape itérative, la luminosité et l'attirance de chaque firefly est calculée, c'est la fonction objectif.

Après avoir comparé la luminosité de chaque luciole avec toutes les autres lucioles, les positions des lucioles sont mises à jour en se basant sur des règles de connaissance de la luciole et de leurs voisins, ces règles sont généralement la position initiale, la distance entre deux lucioles comparés et un mouvement aléatoire.

Après le déplacement, la nouvelle luciole est évaluée, sa position et son intensité de lumière sont mises à jour.

Pendant la boucle de comparaison en deux à deux, la meilleure solution est mise à jour de manière itérative. Le processus de comparaison par pair est répété jusqu'à la satisfaction des critères de résiliation **[11].**

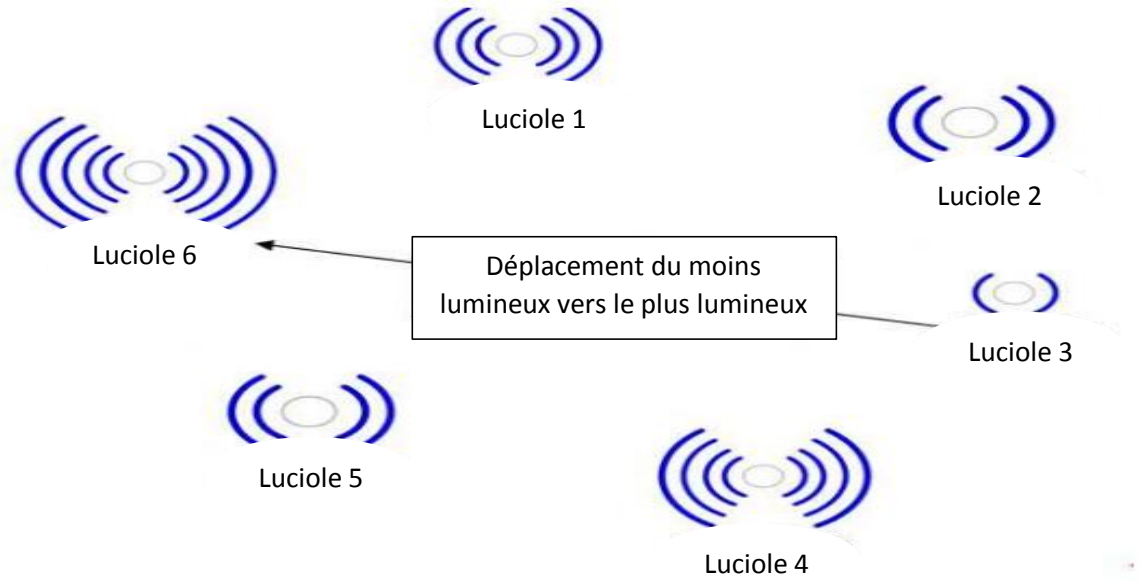


Figure I.9: Déplacement des lucioles dans une itération

- ***Critère d'arrêt***

Les étapes précédentes (déplacement et mise à jour) appliquées d'une manière itérative, cette boucle s'arrêt jusqu'à ce que la condition d'arrêt soit satisfaite. Cette condition correspond soit à un nombre maximum de génération fixée au départ, ou quand une solution satisfaisable proche de la solution optimale est atteinte. Aussi on peut arrêter la boucle quand les résultats de l'algorithme sont devenus stable, pour éviter la perte de temps.

L'algorithme suivant a été implémenté sous Matlab et exécuté sur un PC comme suit :

Algorithme : Algorithme des lucioles

Début

t=0

Définir une fonction objective ***f(x)***

Générer une population initiale de fireflies x_i *tel que* ***(i=1,…, n)***

Déterminer les intensités de lumière I_i *à un point* x_i *par la fonction objective* $f(x_i)$

Tant que *(t <* ***Max itération****)* ***faire***

Pour *i=1* ***à*** *n (toutes les fireflies)*

Pour *j=1* ***à*** *n (toutes les fireflies)*

Si *(*f_i *ne domine pas* f_j *)* ***alors***

Calcule la distance entre firefly ***i*** *et firefly* ***j*** *(*r_{ij}*)*

Attractivité $\boldsymbol{\beta}$ *varie selon la distance* $\boldsymbol{r_{ij}}$

Déplacer firefly **i** *vers firefly* **j** *avec l'attractivité* $\boldsymbol{\beta}$

Fin si

Evaluer la nouvelle solution **i**

Insérer la nouvelle solution **i**

Fin

Fin

Classer les lucioles

Trouver la meilleure firefly en fonction objectif

t++

Fin tant que

Fin

I.6.2.2.4.Paramétres d'algorithme des lucioles

- ***Nombre des lucioles***

Le nombre de lucioles aussi appelé la taille de la population initiale, à une influence directe sur l'algorithme des Lucioles pour cela il est très important de bien choisir ce paramètre pour garantir un meilleur compromis entre la qualité de la solution et la rapidité de l'exécution D'après les différents tests effectués dans **[12]** , on constate que plus la taille de la population est grande, sa diversité augmente est donc la qualité de solution est meilleure. Par conséquent si le temps d'exécution de l'algorithme augmente, il affecte l'efficacité de l'algorithme.

Par contre si le nombre de luciole est petit, il y aura alors une probabilité de converger vers un optimum local et donc il est plus efficace d'avoir un nombre important de lucioles pour assurer une diversité et éviter le problème des minima locaux.

- ***Intensité***

Dans l'algorithme des lucioles, il y a deux points importants : la variation de l'intensité de la lumière et la formulation de l'attractivité. Par souci de simplicité, ***Xin-SheYang*** suppose que l'attraction d'une luciole est déterminée par sa luminosité qui à son tour est associée à la fonction d'objectif.

Dans la forme la plus simple, l'intensité lumineuse $\boldsymbol{I(r)}$ varie en fonction de la loi du carré inverse $\boldsymbol{I(r) = I_0/r^2}$ où $\boldsymbol{I_0}$ est l'intensité à la source, ou dans un milieu donné avec une lumière fixe $\boldsymbol{\gamma}$ coefficient d'absorption, l'intensité lumineuse varie avec la distancer, qui est :

$$\boldsymbol{I\,(r) = I_0 e^{-\gamma r}} \qquad \textbf{(1)}$$

Afin d'éviter la singularité à $\boldsymbol{r = 0}$ dans l'expression $\boldsymbol{I(r) = I_0/r^2}$, l'effet combiné de la fois la loi carrée inverse et l'absorption peuvent être estimés à l'aide de la forme suivante :

$$\boldsymbol{I(r) = I_0 e^{-\gamma r^2}} \qquad \textbf{(2)}$$

Parfois, on peut avoir besoin d'une fonction qui diminue de façon monotone à un rythme plus lent. Dans ce cas, nous pouvons utiliser l'approximation suivante :

$$I(r) = \frac{I_0}{1+\gamma r^2} \quad (3)$$

- ***Attractivité***

L'attractivité varie en fonction de la distance $r_{i,j}$ entre les lucioles ***i*** et ***j***. En outre, l'intensité lumineuse diminue en s'éloignant de sa source, et la lumière est également absorbée par l'entourage, de sorte de permettre à l'attractivité de varier avec le degré d'absorption.

Sachant que l'attraction d'une luciole est proportionnelle à l'intensité des lucioles adjacentes, La formule de cette attractivité β d'une luciole peut être définie comme :

$$\beta(r) = \beta_0 e^{-\gamma r^2} \quad (4)$$

Où β_0 est l'attraction à $r = 0$. Comme il est souvent plus rapide pour calculer $\frac{1}{1+r^2}$ D'une fonction exponentielle, le cas échéant, peut avantageusement être remplacé par :

$\beta = \frac{\beta_0}{1+\gamma r^2}$, équation *(4)* définit une caractéristique de distance $r = \frac{1}{\sqrt{\gamma}}$, sur laquelle l'attractivité change sensiblement de β_0 à $\beta_0 e^{-1}$ Dans la mise en œuvre, la forme réelle de la fonction de l'attractivité $\beta(r)$ peut être des fonctions décroissantes de façon monotone comme la généralisation suivante :

$$\beta(r) = \beta_0 e^{-\gamma r^m} \quad , m \geq 1 \quad (5)$$

- ***Coefficient d'absorption***

Le coefficient d'absorption γ contrôle la variation de l'attractivité en fonction de la distance entre deux lucioles communiquées. Il est dans l'intervalle $[0, \infty]$.

$\gamma = 0$ correspond à aucun changement, pas de variation ou attractivité constante,

$\gamma = \infty$, correspond à une recherche aléatoire complète.

Nous préférons garder la valeur de $\gamma \in [0,1]$, $\gamma = 1$ entraîne une attractivité proche de zéro qui est encore équivalente à la recherche aléatoire complète.

Ce coefficient d'absorption personnalisé pourrait être basé sur la "longueur caractéristique" de l'espace de recherche optimisé.

- ***Distance***

La distance entre n'importe quelles deux lucioles ***i*** et ***j*** aux emplacements X_i et X_j respectivement est la distance cartésienne :

$$r_{ij} = \| X_i - X_j \| = \sqrt{\sum_{k=1}^{d} (X_{i,k} - X_{j,k})^2} \quad (6)$$

Où X_i , est la ***kéme*** composante de la ***iéme*** luciole, et ***d*** est le nombre de dimensions. Pour ***d = 2,*** on a :

$$r_{ij} = \sqrt{(y_i - y_j)^2 + (x_i - x_j)^2} \qquad (7)$$

- ***Mouvement***

Le déplacement d'une luciole ***i*** attirée par une luciole ***j*** plus lumineuse (attrayante), est déterminé par :

$$x_i = x_i + \beta_0 e^{-\gamma r^2}(x_j - x_i) + \alpha(rand - 1/2) \qquad (8)$$

Où le premier terme présent la position actuelle d'une luciole, le second terme est utilisé pour prendre en compte l'attractivité d'une luciole à l'intensité lumineuse vue par des lucioles adjacentes, et le troisième terme est utilisé pour le mouvement aléatoire d'une luciole dans le cas où il n'y a pas d'autre plus lumineuse.

Le coefficient α est un paramètre de répartition aléatoire, tandis que $rand$ est un générateur de nombre aléatoire distribué de façon uniforme dans l'espace ***[0,1]*** **[10]**.

- ***Nombre de génération***

La convergence vers la solution optimale globale n'est pas garantie dans tous les cas même si les expériences dénotent la grande performance de la méthode. De ce fait, il est fortement conseillé de doter l'algorithme d'une portée de sortie en définissant un nombre maximum d'itération.

I.6.2.2.5.Les avantages de FA

- Assez bons résultats dans beaucoup d'applications, même si le critère est stochastique par nature.
- Avec une initialisation exhaustive, l'optimum global est très rarement raté.
- Bien adaptés aux implémentations sur des machines parallèles ou processeurs graphiques (même avec plusieurs essaims).

I.6.2.2.6.Les inconvénients de FA

- Algorithme Firefly à quelques inconvénients tels que se faire piéger en plusieurs optima locaux.
- Firefly algorithme effectue parfois la recherche locale ainsi et parfois est incapable de débarrasser complètement d'eux.
- En outre Firefly ne pas mémoriser ou de se rappeler toute l'histoire de meilleure situation, et ils peuvent finir par manquer leurs situations.
- Il très difficile (voir impossible) de démontrer des résultats de convergence .Le nombre de firefly doit être assez grand pour une bonne efficacité.

I.7.Conclusion

Dans ce chapitre a été mentionné, nous avons essayé d'aborder quelque notions de base en optimisation et des définitions de notions confrontées souvent dans le domaine de l'optimisation combinatoire puis nous avons présenté aussi les méthodes d'optimisation combinatoire, qui sont « les méthodes exactes » et « les méthodes approchés », ensuite nous avons dévoilé les métaheuristiques à solution unique .Enfin, nous avons tenté d'élucider, les métaheuristique à population de solution l'algorithme " PSO " et " Firefly" qui sont présenté en détail ou on a pu déceler les avantages et les inconvénients de chaque Algorithme.

Par la suite, nous allons présenter l'état de l'art de les deux algorithmes PSO et FA et leurs domaines d'applications.

CHAPITRE II :
Etat de l'art

II.1. Introduction

Les deux dernières décennies ont vu un développement sans précédent dans le domaine de l'intelligence informatique avec l'avènement du GPU et l'introduction de plusieurs algorithmes d'optimisation puissants qui faire peu ou pas d'hypothèses sur la nature du problème. L'optimisation de l'essaim de particules (PSO) est l'une des nombreuses techniques de ce type et a été largement utilisée dans le traitement des problèmes d'optimisation de fonction continus / discrets, contraints ou non. Beaucoup comme les paradigmes informatiques évolutionnistes populaires tels que les algorithmes génétiques et différentiels Évolution, le fonctionnement interne du PSO utilise suffisamment les règles de transition probabilistes pour faire des recherches parallèles dans l'hyperespace de la solution sans hypothèse explicite de dérivées informations. Le modèle physique sous-jacent sur lequel les règles de transition sont basées est l'un des comportements collectifs émergents résultant de l'interaction sociale des troupeaux d'oiseaux et des bancs de poissons.

Le calcul inspiré de la nature est devenu un nouveau paradigme en matière d'optimisation, d'apprentissage automatique, d'exploration de données et d'intelligence computationnelle avec un large éventail d'application. L'essence de l'informatique inspirée de la nature réside dans les algorithmes inspirés de la nature telle que l'algorithme génétique (GA), l'optimisation des essaims de particules (PSO) et l'algorithme firefly.

II.2. Les Différents domaines d'application de la méthode PSO

Le domaine de la formation des réseaux neuronaux c'est la première application pratique de l'optimisation des essaims de particules qui réalise par ***Kennedy*** et ***Eberhart*** en ***1995***. Au fil du temps, PSO a été utilisé dans un large éventail d'application, par exemple, Conception d'antenne, le traitement du signal, La mise en réseau, Biomédicale, Electronique et électromagnétique, Robotique, Conception et La modélisation, Image et Graphique, Pouvoir génération et Contrôle, Systèmes flous, Regroupement, données exploitation minière, Optimisation, Prédiction et prévision **[13]**.

Ces derniers temps, l'algorithme pso a indiqué énormément d'intérêt et a été utilisé à de nombreuses applications notamment, ***Ghorpade-Aher J*** et ***al*** **[14]** ont collaboré sur le regroupement de données multidimensionnelles avec l'algorithme PSO, un algorithme PSO avancé intitulé Limite basée sur la segmentation soustractive Essaim de particules adaptatives restreint Algorithme d'optimisation (SC-BR-APSO) pour grouper des données multidimensionnelles. Les auteurs comparent leur algorithme avec plusieurs algorithmes utilisant neuf jeux de données différentes et affirment les résultats avec un taux d'erreur minimum et un taux de convergence maximum.

Dudeja **[15]** a suggéré l'Algorithme d'optimisation de l'essaim de particules modifié à base floue pour les problèmes de chemin le plus court est Une méthode qui réduit le coût et la consommation de temps à l'aide de règles floues. L'auteur propose une amélioration de l'exécution de Modification Particule Swarm Optimisation (MPSO) pour évaluer le calcul de manière la plus limitée avec des règles floues. Cette méthode hybride intitulée L'optimisation de l'essaim de particules modifiées basées sur le flou a montré une efficacité d'encodage améliorée, la consommation de temps et le coût.

Mahesa et ***al*** **[16]** Ont travaillé sur l'Optimisation de c-signifie groupage flou en utilisant l'optimisation de l'essaim de particules dans la segmentation de l'image de tumeur cérébrale, une technique de clustering utilisant le flou c-signifie optimisé par l'utilisation de l'algorithme PSO étiqueté comme (FCMPSO). L'étude vise à prouver que cette optimisation donne de meilleurs résultats que la version non optimisée des c-means

flous. Pour ce faire, les auteurs ont testé avec six images de tumeurs cérébrales et ont démontré que l'utilisation de PSO améliore les résultats de clustering.

L'apprentissage du renforcement a été utilisé dans plusieurs algorithmes d'optimisation métaheuristique, y compris PSO, cinq opérations différentes pour les particules dans PSO sont définies, effectué par ***Samma*** et ***al*** **[17]**, tel que l'exploration, la convergence, le saut en hauteur, le saut en bas et le réglage fin, et Qlearning est appliqué à chaque particule pour sélectionner une opération parmi les cinq candidats en fonction de l'état actuel.

Li Ke et ***al*** **[18]** a utilisé une méthode d'apprentissage du renforcement pour apprendre des méthodes d'optimisation par gradient existantes et a utilisé un filet stratégique bien formé pour optimiser les poids des différents réseaux neuronaux dans la classification. Une méthode d'apprentissage de renforcement continu basée sur la distribution normale a été appliquée pour trouver la fonction politique/mise à jour qui a le taux de convergence le plus élevé. Dans ces travaux, une fois que le politique (fonction de mise à jour) a été formé, la politique ne changerait pas lorsqu'elle est utilisée pour optimiser d'autres problèmes.

Xu Yue et ***al*** **[19]** ont utilisé la méthode Q-learning pour sélectionner différentes topologies de communication à chaque itération du processus PSO. La méthode Q-learning a également été appliquée dans un algorithme de recuit simulé (SA) pour résoudre des problèmes d'ingénierie contraints, où l'apprentissage du renforcement a été utilisé pour choisir les meilleures valeurs de paramètres parmi un ensemble de candidats (***Samma*** et ***al***) **[20]**.

Dominik Weikert et ***al*** **[21]** ont présenté un nouvel algorithme appelé « Particle Swarm Contour Search » (PSCS) a Particle Swarm Algorithme d'optimisation inspiré pour trouver les contours des objets dans des environnements 2D. La plupart des algorithmes de repérage des contours sont basés sur le traitement d'images et nécessitent une vue d'ensemble complète de l'espace de recherche dans lequel le contour doit être trouvé. Toutefois, pour des applications réelles, cela exigerait une connaissance complète de l'espace de recherche, ce qui n'est pas toujours faisable ou possible. Les auteurs suggèrent que cet algorithme supprime cette exigence et est uniquement basé sur les informations locales des particules pour identifier avec précision un contour. Les particules recherchent le contour d'un objet, puis le traversent en utilisant leurs informations connues sur les positions à l'intérieur et à l'extérieur de l'objet, leurs expériences prouvent que l'algorithme PSCS proposé peut fournir des résultats comparables à l'état de la technique.

II.3. Les Différents domaines d'application de la méthode FA

En raison de sa simplicité et de son efficacité de recherche, L'algorithme Firefly et ses variantes ont attiré beaucoup l'attention et ont été appliquées à divers problèmes du monde réel notamment les problèmes de conception techniques et multimodale hautement non linéaires, la conception des antennes, Les classifications et les regroupements, formation de réseaux neuronaux, prédire les maladies cardiaques et reprogrammer de l'énergie réelle pour la gestion de la congestion concernant les centrales hydroélectriques à pompage turbinage **[22]**. De plus, FA déterminer l'emplacement et la taille optimale de la génération distribuée (DG) dans les réseaux de distribution électriques, calculer les distances entre les dispositifs, problème du contrôle de la fréquence de charge dans le système d'énergie thermique de réchauffage interconnecté, pour minimiser les pertes de puissance causées par un courant élevé et amélioré le profil de tension dans le réseau de distribution **[23]**.

U. Singh et ***M. Rattan*** **[24]**, présentent l'application de l'algorithme Firefly pour la conception de réseaux d'antennes circulaires concentriques multiples dilués. L'étude vise à réaliser un ensemble d'éléments

isotropes à excitation uniforme qui généreront un faisceau crayon dans le plan vertical avec un niveau minimum de lobes latéraux (SLL). Ils appliquent cet algorithme pour deux cas différents d'amincissement de réseaux circulaires concentriques (anneaux). Le premier cas est avec un espacement d'intérêt uniforme fixé à 0,5λ ou multiple et le second cas est avec un espacement inter éléments optimal ou multiple. Le pourcentage d'amincissement du réseau est maintenu égal ou supérieur à 50 % et la largeur du faisceau est maintenue égale ou inférieure à celle d'un réseau annulaire entièrement peuplé, uniformément excité et espacé de 0,5 λ du même nombre d'éléments et d'anneaux.

M.Othman, W.ElKhattam, A.Y. Abdelaziz et ***Y.G. Hegazy*** **[25]**, appliquent l'algorithme de Firefly (FA) pour déterminer l'emplacement optimal et la capacité de tension contrôlée DG afin d'atteindre une perte de puissance prévue est proposée. Le DG de l'algorithme proposé est modélisé comme un bus à tension contrôlée (PV) avec la flexibilité nécessaire pour être converti en bus à puissance constante (PQ) en cas de violation de la limite de puissance réactive. Ils examinent cette méthode sur le système de test IEEE 37-bus. Ils comparent les résultats obtenus à partir de l'algorithme proposé avec les résultats obtenus à partir d'autres techniques d'optimisation.

De plus, FA a été appliqué dans les domaines allant de la technique de conception aux systèmes énergétiques et de l'ordonnancement à traitement d'images. Par exemple, FA a été appliqué à la conception de dilatateurs radiaux dans les cycles Rankine organiques, à l'optimisation de la conception des cadres en acier, au système de génération distribué, à la conception des faisceaux, à l'optimisation des réseaux neuronaux d'ondes, à l'identification des modèles d'hystérésis, détection d'anomalies sismo ionosphériques TEC et recherche structurale en chimie, la conception des échangeurs de chaleur de la coque et du tube, sélectionner les paramètres des processus d'usinage avancés, l'optimisation du flambage thermique des plaques composites, optimiser les modulateurs de résonateur à l'échelle micrométrique, et une ADI supervisée pour obtenir un emplacement optimal des génératrices distribuées **[26]**.

Kaur M et ***Ghosh S*** **[27]** ont utilisé une méthode floue FA pour la reconfiguration des réseaux de distribution non équilibrés. De plus, ***Singh S.K, Sinha, N*** et ***Goswami A.K*** **[28]** a combiné la méthode des FA avec la méthode des moindres carrés pour estimer les harmoniques du système de puissance.

Dans le domaine de l'ingénierie énergétique et des systèmes énergétiques, l'algorithme de luciole est appliqué pour prédire les débits de gaz provenant des réservoirs de condensats de gaz, pour résoudre les applications des éoliennes. L'application d'un réseau neuronal FA-BP pour prévoir le prix de l'électricité la planification de l'expansion du transport en plusieurs étapes **[26]**. De plus, ***Satapathy P, Dhar S et Dash P.K*** **[29]** ont utilisé une approche hybride fondée sur les SS-FA pour améliorer la stabilité des Microgrid au diesel PV-BESS.

L'utilisation de FA dans le domaine médical pour la détection des disques optiques dans les images rétiniennes, pour le suivi visuel, et pour optimiser les réseaux neuronaux granulaires modulaires pour la reconnaissance humaine, une FA discrète pour l'extraction des images hyper spectrales **[26]**.

Dans le domaine des séries chronologiques et des prévisions, FA est appliqué un modèle combiné pour la prévision de la charge électrique, pour la prévision de la capacité en combinaison avec la machine vectorielle de soutien. En outre, et un modèle hybride pour la prévision du rayonnement solaire **[26]**.

Dans le domaine de la planification et de la navigation, les FA utilise pour planifier les trajectoires de navigation **[26]**.

Dans le domaine du deep learning et du génie logiciel, utilise une approche basée sur l'algorithme de Firefly pour générer des séquences de test logicielles optimales, pour les paramètres d'apprentissage dans

les réseaux de croyances profondes **[26]**. ***Kaushik A***, ***Tayal D.K***, ***Yadav K*** et ***Kaur A*** **[30]** ont intégré la FA dans un réseau neuronal artificiel pour prédire avec précision les coûts des logiciels.

D'autres applications comprennent l'optimisation structurelle à l'échelle nanométrique, l'identification du complexe protéique par, la prédiction de la structure protéique. En outre, utilise FA pour réaliser l'optimisation multimodale. En autre, un FA amélioré pour les PWM programmés dans les onduleurs multiniveaux avec des sources DC réglables **[26]**.

L'algorithme Firefly est appliqué dans la prévision de régression, la planification de l'expansion de la transmission en plusieurs étapes, prévision des maladies du cœur, sélection des caractéristiques, segmentation de l'image, planification des trajectoires, etc **[31]**. De plus, certains chercheurs ***Singh*** et ***Salgotra*** **[32]** ont utilisé des algorithmes de luciole pour résoudre des problèmes de conception de réseaux d'antennes, y compris la synthèse de réseaux d'antennes linéaires et l'amincissement de multiples réseaux d'antennes circulaires concentriques.

Pour les études plus récentes, des revues complètes sont effectuées. Plus récemment, les algorithmes de luciole basés sur le chaos pour l'optimisation de dômes en acier contreventés de manière cyclique de grande taille. Une FA modifiée a été utilisée pour suivre les déversements de pétrole en expansion et estimer la zone. De plus, FA a été utilisé pour découvrir les leaders d'opinion dans les réseaux sociaux en ligne. Une FA d'intégration a été développé pour résoudre l'optimisation concernant le système de planification de traitement assisté par ordinateur. La planification de chemin dans un environnement incertain a été étudiée à l'aide de FA. De plus, une reconfiguration du réseau a été effectuée pour les réseaux de distribution déséquilibrés à l'aide d'une FA floue, tandis qu'un placement optimal de dispositif de commutation répulsif basé sur une FA a été utilisé pour les systèmes de distribution d'énergie. D'autre part, fournir un examen détaillé de FA et de ses applications dans le traitement d'images. Par exemple, un FA modifié a été utilisé pour la segmentation d'images couleur avec seuillage à plusieurs niveaux. FA a été modifié et utilisé pour la sélection de fonctionnalités pour la détection d'intrusion réseau. L'optimisation des réseaux de neurones granulaires modulaires pour la reconnaissance humaine a été réalisée par FA. La planification de la trajectoire du bras du robot a également été optimisée par un hybride FA et Q-learning. Le regroupement flou a été combiné avec FA pour la préservation de la confidentialité dans les réseaux sociaux **[33]**.

II.4.Conclusion

Dans ce chapitre, nous avons étudié les différents domaines d'application de certaines algorithmes inspires de la nature récemment introduits tels que : l'algorithme PSO et l'algorithme firefly, nous avons passée en revue les articles de recherche de divers auteurs, qui les ont utilisés pour résoudre des problèmes d'optimisation.

Dans le chapitre suivant, nous allons présenter nouveau algorithme d'optimisation c'est un algorithme d'optimisateur d'équilibre.

CHAPITRE III :

Algorithme d'Optimisation d'Equilibre

III.1.Introduction

Les approches d'optimisation inspirées par la nature récemment développée sont de bonnes techniques pour trouver des solutions globales pour des applications d'optimisation de la vie réelle. L'Algorithme d'Optimisation d'Equilibre (en anglais Equilibrium Optimizer Algorithm EOA) est une nouvelle approche d'optimisation, qui s'inspire des modèles de bilan massique de volume de contrôle mis en œuvre pour déterminer les états dynamiques et d'équilibre.

L'équilibre est un état dans lequel des forces opposées sont équilibrées. Les modèles d'optimisation et d'équilibre sont couramment utilisés en génie et en science de l'environnement. Cela permet de résoudre différents types de problèmes d'équilibre, tels que les problèmes d'optimisation mono-objective, multi objectifs, etc…, Par conséquent, nous explorant le récent algorithme nommé Equilibrium Optimizer (EO), développés pour résoudre les problèmes d'optimisation.

Toutes ces informations que nous avons utilisées sont extraites de la référence **[34]**

III.2.Inspiration

Dans l'algorithme d'optimisation d'équilibre, chaque particule (solution) avec sa concentration (position) agit comme un agent de recherche. Les agents de recherche mettent à jour aléatoirement leur concentration par rapport aux meilleures solutions, à savoir les candidats à l'équilibre, pour finalement atteindre l'état d'équilibre (résultat optimal). Il est prouvé qu'un terme de « taux de génération » bien défini renforce la capacité d'EO en matière d'exploration, d'exploitation et d'évitement des minima locaux. Dans lequel une équation de bilan massique est utilisée pour décrire la concentration d'une substance non réactive composante dans un volume de contrôle en fonction de ses divers mécanismes de source et de puits. La masse équation de balance fournit la physique sous-jacente pour la conservation de la masse entrante, sortant, et générée dans un volume de contrôle. Une équation différentielle ordinaire de premier ordre exprimant l'équation générique de bilan massique, dans laquelle le changement de masse dans le temps est égal à la quantité de la masse qui entre dans le système plus la quantité générée à l'intérieur moins la quantité qui par le système est décrite comme suit :

$$V\frac{dC}{dt} = QC_{eq} - QC + G \qquad (1)$$

- C : est la concentration à l'intérieur du volume de contrôle (V).
- $V\frac{dC}{dt}$: est le taux de changement de masse dans le volume de contrôle.
- Q : est le débit volumétrique entrant et sortant du volume de contrôle.
- C_{eq} : est la concentration à un état d'équilibre où il n'y a pas de production à l'intérieur du volume de contrôle.
- G : est le taux de production de masse à l'intérieur du volume de contrôle.

Lorsque $V\frac{dC}{dt}$ atteint zéro, un état d'équilibre stable est atteint. Un réarrangement de l'Eq **(1)** permet de résoudre $\frac{dC}{dt}$ en fonction de $\frac{Q}{V}$, où $\frac{Q}{V}$ représente l'inverse du temps de résidence, appelé ici λ, ou le taux de rotation (c'est-à-dire $\lambda = \frac{Q}{V}$). par la suite, on peut également réorganiser l'Eq **(1)** pour déterminer la concentration dans le contrôle volume (C) en fonction du temps (**t**) :

$$\frac{dC}{\lambda C_{eq} - \lambda C + \frac{G}{V}} = dt \qquad (2)$$

Eq (**3**) montre l'intégration de l'Eq (**2**) au fil du temps :

$$\int_{C_0}^{C} \frac{dC}{\lambda C_{eq} - \lambda C + \frac{G}{V}} = \int_{t_0}^{t} dt \qquad (3)$$

Il en résulte :

$$C = C_{eq} + (C_0 - C_{eq})F + \frac{G}{\lambda V}(1 - F) \qquad (4)$$

Dans l'Eq (**4**), F est calculé comme suit :

$$F = e^{-\vec{\lambda}(t - t_0)} \qquad (5)$$

Où t_0 et C_0 sont le temps de démarrage et la concentration initiales, en fonction de l'intervalle d'intégration. Eq (**4**) peut être utilisé pour estimer la concentration dans le volume de contrôle avec un chiffre d'affaires connu ou pour calculer le taux de roulement moyen en utilisant une régression linéaire simple avec un taux de production et autres conditions.

La figure suivante présente un organigramme des étapes de l'Algorithme d'Optimisation d'Equilibre :

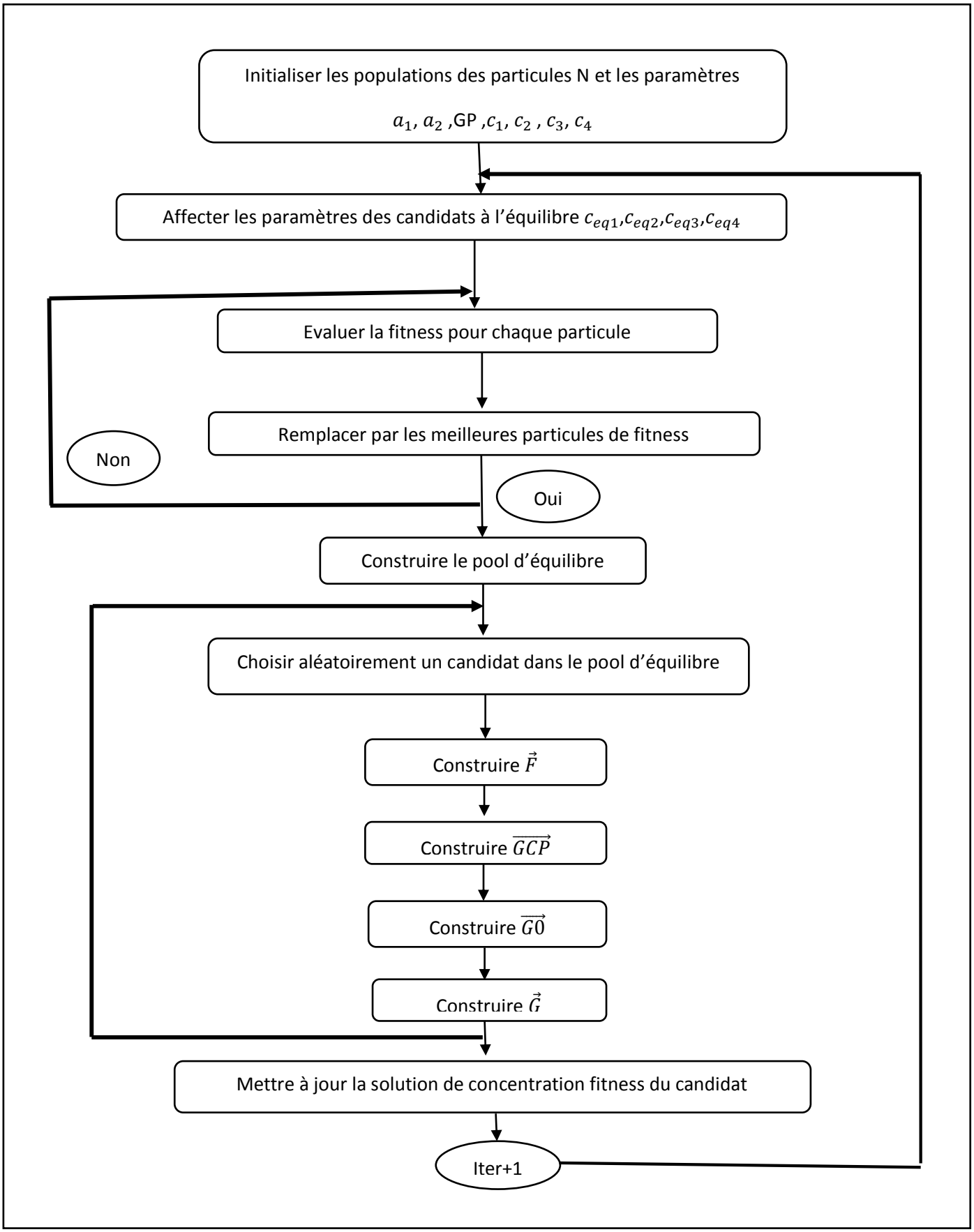


Figure III.1 : Organigramme de l'algorithme d'optimisation d'équilibre

III.3.Description de l'organigramme de l'algorithme EO

III.3.1.Initialisation et fonction d'évaluation

L'optimiseur d'équilibre (EO) est un algorithme métaheuristique basé sur la physique. Similaire aux autres algorithmes métaheuristiques, EO génère une population initiale distribuée aléatoirement dans l'espace de recherche avec l'équation suivante :

$$C_i^{initial} = C_{min} + rand_i(C_{max} - C_{min}) \qquad i=1, 2....n \qquad (6)$$

Où :

- $C_i^{initial}$: est le vecteur de concentration initiale de la particule i
- C_{min}, C_{max} : désignent les valeurs minimales et maximales pour les dimensions
- $rand_i$: est un vecteur aléatoire dans l'intervalle de **[0,1]**
- n : est un nombre de particules que la population. Les particules sont évaluées pour leur fonction de fitness et sont ensuite triées pour déterminer les candidats à l'équilibre.

III.3.2.Pool d'équilibre et candidats

Après l'initialisation, les quatre particules ayant la meilleure fitness sont sélectionnées pour former le pool d'équilibre, et les particules du pool d'équilibre sont mises à jour après chaque itération.

$$\vec{C}_{eq,pool} = \{\vec{C}_{eq(1)}, \vec{C}_{eq(2)}, \vec{C}_{eq(3)}, \vec{C}_{eq(4)}, \vec{C}_{eq(ave)}\} \qquad (7)$$

Où :

- $\vec{C}_{eq,pool}$: indique le pool d'équilibre.
- $\vec{C}_{eq(1)}$, $\vec{C}_{eq(2)}$, $\vec{C}_{eq(3)}$, $\vec{C}_{eq(4)}$: sont les quatre meilleures particules individuelles à ce jour.
- $\vec{C}_{eq(ave)}$: est la moyenne des quatre particules individuelles, telle que :

$$C_{ave} = (C_{eq1} + C_{eq2} + C_{eq3} + C_{eq4})/4 \qquad (8)$$

Chaque particule de chaque itération met à jour sa concentration avec une sélection aléatoire parmi les candidats choisis avec la même probabilité. Par exemple, dans la première itération, la première particule met à jour tous ses concentrations en $\vec{C}_{eq(1)}$, puis, dans la deuxième itération, il peut mettre à jour ses concentrations en fonction de $\vec{C}_{eq(ave)}$. Jusqu'à la fin du processus d'optimisation, chaque particule fera l'expérience du processus de mise à jour avec toutes les solutions candidats reçoivent approximativement le même nombre de mises à jour pour chaque particule.

III.3.3.Enregistrement de la mémoire des particules

L'ajout de procédures d'enregistrement de mémoire aide chaque particule à suivre ses coordonnées dans l'espace, qui influe également sur la valeur de sa condition physique. Ce mécanisme ressemble au concept ***pbest*** dans le PSO. La valeur d'aptitude de chaque particule dans l'itération en cours est comparée à celle de la précédente itération et sera écrasée si elle réalise un meilleur ajustement. Ce mécanisme facilite l'exploitation mais peut augmenter les chances d'être piégé dans les minima locaux si la méthode ne profité de la capacité d'exploration globale.

III.3.4.Terme exponentiel

Le terme suivant contribuant à la règle principale de mise à jour de la concentration est le terme exponentiel (***F***). La définition exacte de ce terme aidera EO à établir un équilibre raisonnable entre l'exploration et exploitation. Étant donné que le taux de rotation peut varier avec le temps dans un volume de contrôle réel, $\vec{\lambda}$ est supposé être un vecteur aléatoire dans l'intervalle de **[0,1].**

$$\vec{F} = e^{-\vec{\lambda}(t - t_0)} \qquad (9)$$

Où le temps t, est défini comme une fonction d'itération (***Iter***) et diminue donc avec le nombre d'itérations :

$$t = \left(1 - \frac{Iter}{Max_{iter}}\right)^{\left(a_2 - \frac{Iter}{Max_{iter}}\right)} \qquad (10)$$

Lorsque ***Iter*** et Max_{iter} présentent respectivement le nombre actuel et le nombre maximal d'itérations, et a_2 est une valeur constante utilisée pour gérer la capacité d'exploitation. Afin de garantir la convergence en ralentissant la vitesse de recherche et en améliorant la capacité d'exploration et d'exploitation de l'algorithme, cette étude prend également en compte :

$$\vec{t}_0 = \frac{1}{\vec{\lambda}} ln\left(-a_1 sign(\vec{r} - 0.5)\left[1 - e^{-\vec{\lambda}t}\right]\right) + t \qquad (11)$$

Où :

- a_1 : est une valeur constante qui contrôle la capacité d'exploration. Plus a_1 est élevé, plus la capacité d'exploration et par conséquent la performance d'exploitation plus faible.
- a_2 : est la meilleure capacité d'exploitation et la plus faible capacité d'exploration.
- $sign(\vec{r} - 0.5)$ effets sur la direction de l'exploration et de l'exploitation.
- ***r*** : est un vecteur aléatoire entre ***0*** et ***1*** respectivement.

Ces constantes sont sélectionnées par des tests empiriques d'un sous-ensemble de tests fonctions. Cependant, ces paramètres peuvent être réglés pour d'autres problèmes si nécessaire.

Eq (**12**) montre la version révisée d'Eq (**9**) avec la substitution d'Eq (**11**) en Eq (**9**) :

$$\vec{F} = a_1 sign(\vec{r} - 0.5)\left[e^{-\vec{\lambda}t} - 1\right] \qquad (12)$$

III.3.5.Taux de production et la mise à jour

Le taux de production ***(G)*** est l'un des termes les plus importants de l'algorithme proposé pour fournir la solution exacte en améliorant la phase d'exploitation. Dans de nombreuses applications d'ingénierie, de nombreux modèles peuvent être utilisés pour exprimer le taux de production en fonction du temps. Par exemple, un modèle polyvalent qui décrit les taux de production comme un processus de désintégration exponentielle de premier ordre est défini comme suit :

$$\vec{G} = \vec{G}_0 e^{-\vec{k}(t - t_0)} \quad (13)$$

Où

- G_0 : est la valeur initiale
- k : indique une constante de décroissance.

Afin d'avoir le modèle de recherche systématique et pour limiter le nombre de variables aléatoires, cette étude suppose k avoir. Et utilise le terme exponentiel précédemment dérivé. Ainsi, l'ensemble final des équations de taux de production est les suivantes :

$$\vec{G} = \vec{G}_0 e^{-\vec{\lambda}(t - t_0)} = \vec{G}_0 \vec{F} \quad (14)$$

Lorsque :

$$\vec{G}_0 = \overrightarrow{GCP}(\vec{C}_{eq} - \vec{\lambda}\vec{C}) \quad (15)$$

$$\overrightarrow{GCP} = \begin{cases} 0.5r_1 & r_2 \geq GP \\ 0 & r_2 < GP \end{cases} \quad (16)$$

Où:

- r_1 , r_2 : sont des nombres aléatoires dans *[0,1]*
- ***GCP*** : est un vecteur construit par la répétition de la même valeur résultant de l'Eq (***16***)

Dans cette équation (***16***), le ***GCP*** est défini comme le taux de production. Paramètre de contrôle, qui comprend la possibilité d'une contribution de la durée de production à la mise à jour processus. La probabilité de cette contribution qui spécifie combien de particules utilise la génération terme pour mettre à jour leur état est déterminée par un autre terme appelé probabilité de génération (***GP***). Le mécanisme de cette contribution est déterminé par l'Eq (***15***) et (***16***). Eq (***16***) se produit au niveau de chaque particule. Par exemple, si GCP est zéro, ***G*** est égal à zéro et toutes les dimensions de cette particule spécifique sont mises à jour sans terme de taux de production. Un bon équilibre entre l'exploration et l'exploitation est atteint avec ***GP = 0,5***. Enfin, la règle de mise à jour d'EO sera la suivante :

$$\vec{C} = \vec{C}_{eq} + (\vec{C} - \vec{C}_{eq})\vec{F} + \frac{\vec{G}}{\vec{\lambda}\vec{V}}(1 - \vec{F}) \quad (17)$$

Où ***F*** est défini dans l'Eq (**12**) et ***V*** est considéré comme unité.

Le premier terme de l'Eq (**17**) est une concentration d'équilibre, où les deuxième et troisième termes représentent les variations de concentration. Le deuxième terme est responsable de la recherche globale de l'espace à trouver un point optimal. Ce terme contribue davantage à l'exploration, profitant ainsi de fortes variations de concentration (c'est-à-dire une différence directe entre un équilibre et un échantillon particule). Comme il trouve un point, le troisième terme contribue à rendre la solution plus précise. Ce terme contribue donc davantage à l'exploitation et bénéficie de faibles variations de concentration, qui sont régies par le terme de taux de production (Eq (**14**)). Selon des paramètres tels que les concentrations de particules et les candidats à l'équilibre, ainsi que le taux de rotation (**λ**), les deuxième et troisième termes peuvent avoir les mêmes signes ou des signes opposés. Le même signe rend la variation grande, ce qui permet de mieux rechercher le domaine complet, et le signe opposé rend la variation petite, aidant dans les recherches locales.

Bien que le deuxième terme tente de trouver des solutions relativement loin de l'équilibre candidats et le troisième terme tente d'affiner les solutions plus près des candidats, ce qui n'est pas toujours le cas. Les faibles taux de roulement (*exp:* $\leq 0,05$) au dénominateur du troisième terme augmentent sa variation et aident l'exploration dans certaines dimensions ainsi.

La figure (***III.2***) démontre une version 1-D de la façon dont ces contribuer à l'exploration et à l'exploitation. $C_1 - C_{eq}$ est représentatif du deuxième terme en Eq (**17**) tandis que C_{eq}-λC_1 représente le troisième terme (**G** est la fonction deG_0). Les termes du taux de production (Eq **14,16**) contrôlent ces variations. Parce que λ change avec le changement de chaque dimension, cette grande variation n'arrive qu'aux dimensions avec de petites valeurs de λ. Il est à noter que cette fonctionnalité fonctionne comme un opérateur de mutation dans les algorithmes évolutifs et aide grandement EO à exploiter les solutions.

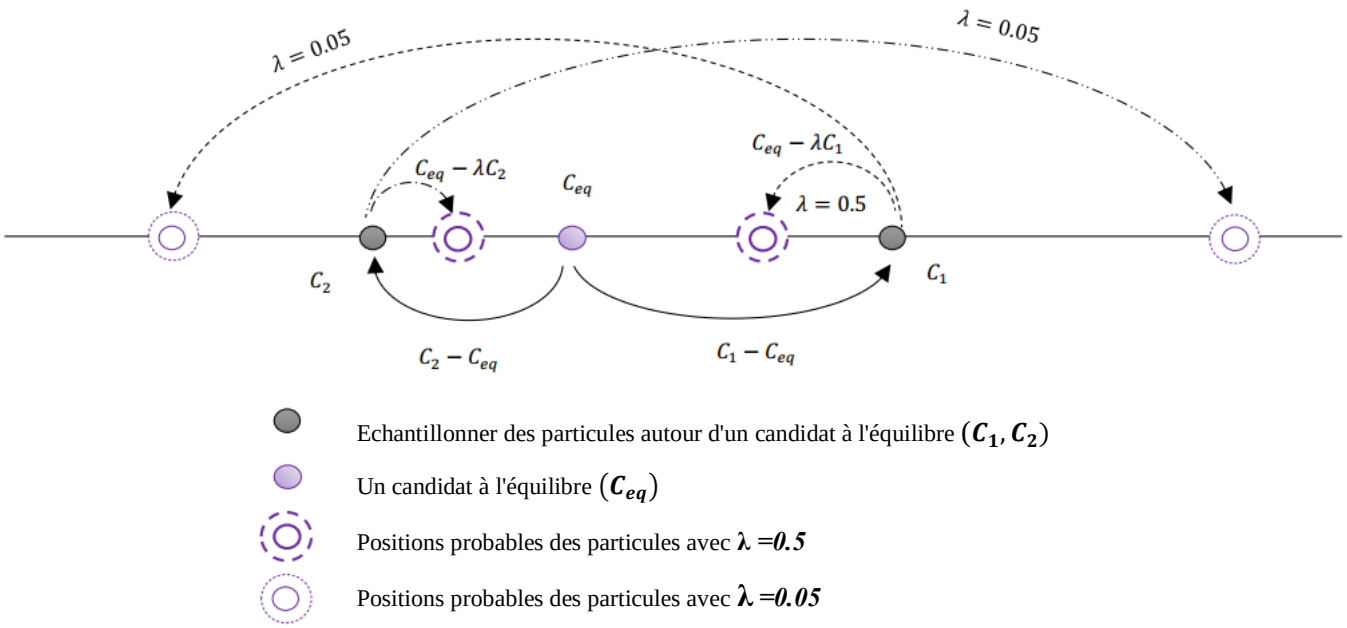


Figure III.2 : Présentation 1-D des concentrations mise à jour des aides à l'exploration et à l'exploitation

La figure (***III.3***) montre une esquisse conceptuelle de la collaboration de tous les candidats à l'équilibre sur un échantillon les particules et comment elles affectent la mise à jour de la concentration, l'une après l'autre, dans l'algorithme proposé. Étant donné que les positions topologiques des candidats à l'équilibre sont diverses dans les itérations initiales, et le terme exponentiel génère de grands nombres aléatoires, ce processus de mise à jour étape par étape aide les particules pour couvrir l'ensemble du domaine dans leur recherche. Un scénario opposé se produit dans le dernier lorsque les candidats entourent le point optimal par des configurations similaires. À ces temps, le terme exponentiel génère de petits nombres aléatoires, ce qui aide à affiner les solutions en fournissant des tailles de pas plus petites. Ce concept peut également être étendu à des dimensions plus hyperespace où la concentration sera mise à jour avec le mouvement de la particule dans l'espace dimensionnel.

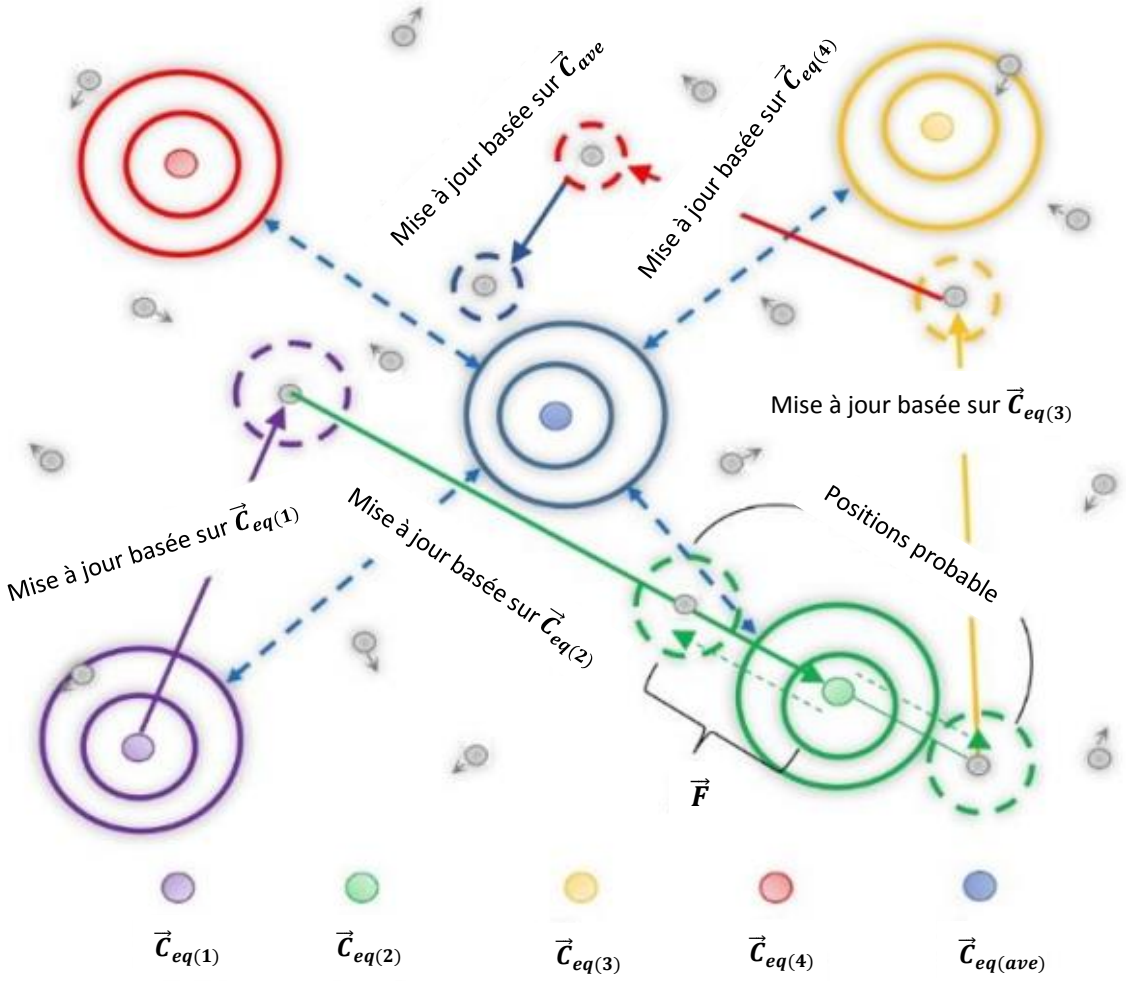


Figure III.3 : Collaboration des candidats a l'équilibre dans la mise à jour de la concentration d'une particule en dimensions 2D

Le pseudo code de l'algorithme d'optimisation d'équilibre qui a été implémenté sous Matlab et exécuté sur un PC peut se montre comme suit :

Initialiser les populations des particules i=1,…,n
Affecter aux candidats l'équilibre fitness un grand nombre
Affecter des paramètres libres a_1*=2;* a_2*=1; GP=0.5;*
Tant que *Iter < Max_iter*
  Pour *i=1: nombre de particules (n)*
    Calculer la fitness de la $i^{éme}$ *particule*
    Si $fit(\vec{C}_i) < fit(\vec{C}_{eq1})$
      Remplacer $\vec{C}_{eq1}$ *par* $\vec{C}_i$ *et fit* $(\vec{C}_{eq1})$ *par* $fit(\vec{C}_i)$
    Sinon $fit(\vec{C}_i) > fit(\vec{C}_{eq1})$ & $fit(\vec{C}_i) < fit(\vec{C}_{eq2})$
      Remplacer $\vec{C}_{eq2}$ *par* $\vec{C}_i$ *et fit* $(\vec{C}_{eq2})$ *par* $fit(\vec{C}_i)$
    Sinon $fit(\vec{C}_i) > fit(\vec{C}_{eq1})$ & $fit(\vec{C}_i) > fit(\vec{C}_{eq2})$ & $fit(\vec{C}_i) < fit(\vec{C}_{eq3})$

Remplacer $\vec{C}_{eq3}$ *par* $\vec{C}_i$ *et fit* ($\vec{C}_{eq3}$) *par* $fit(\vec{C}_i)$

Sinon $fit(\vec{C}_i) > fit(\vec{C}_{eq1})$ & $fit(\vec{C}_i) > fit(\vec{C}_{eq2})$ & $fit(\vec{C}_i) > fit(\vec{C}_{eq3})$ & $fit(\vec{C}_i) < fit(\vec{C}_{eq4})$

Remplacer $\vec{C}_{eq4}$ *par* $\vec{C}_i$ *et fit* ($\vec{C}_{eq4}$) *par* $fit(\vec{C}_i)$

Fin *(Si)*

Fin *(Pour)*

$\vec{C}_{ave} = (\vec{C}_{eq1} + \vec{C}_{eq2} + \vec{C}_{eq3} + \vec{C}_{eq4})/4$

Construire le pool d'équilibre $\vec{C}_{eq,pool} = \{\vec{C}_{eq(1)}, \vec{C}_{eq(2)}, \vec{C}_{eq(3)}, \vec{C}_{eq(4)}, \vec{C}_{eq(ave)}\}$

Réaliser un économie de mémoire (if Iter > 1)

Affecter $t = \left(1 - \frac{Iter}{Max_{iter}}\right)^{\left(a_2 - \frac{Iter}{Max_{iter}}\right)}$ *Eq (10)*

Pour *i=1: nombre de particules (n)*

Choisir au hasard un candidat dans le pool d'équilibre (vecteur)

générer des vecteurs aléatoires de $\vec{\lambda}$, $\vec{r}$ *à partir de Eq (12)*

Construire $\vec{F} = a_1 sign(\vec{r} - 0.5)\left[e^{-\vec{\lambda}t} - 1\right]$ *Eq (12)*

Construire $\overrightarrow{GCP} = \begin{cases} 0.5r_1 & r_2 \geq GP \\ 0 & r_2 < GP \end{cases}$ *Eq (16)*

Construire $\vec{G}_0 = \overrightarrow{GCP}(\vec{C}_{eq} - \vec{\lambda}\vec{C})$ *Eq (15)*

Construire $\vec{G} = \vec{G}_0 . \vec{F}$ *Eq (14)*

Mettre à jour les concentrations $\vec{C} = \vec{C}_{eq} + (\vec{C} - \vec{C}_{eq})\vec{F} + \frac{\vec{G}}{\vec{\lambda} V}(1 - \vec{F})$ *Eq(17)*

Fin *(Pour)*

Iter=Iter+1

Fin *(Tant que)*

III.4.Capacité d'exploration de l'EO

Pour résumer ces termes, il y a plusieurs paramètres et mécanismes dans EO qui mènent à exploration, comme suit :

- $\boldsymbol{a_1}$**:** contrôle la quantité d'exploration (magnitude) de l'algorithme. Il détermine la distance de la nouvelle position par rapport au candidat à l'équilibre. Plus la valeur $\boldsymbol{a_1}$est élevée, plus la capacité d'exploration est élevée. à noter que des nombres supérieurs à trois diminueraient considérablement la performance de l'exploration. Puisque $\boldsymbol{a_1}$peut augmenter la variation de la concentration, elle devrait être assez grande pour élargir la capacité d'exploration. Cependant, sur la base de tests empiriques, il a été constaté que les valeurs supérieures à trois poussent les agents à chercher sur les limites. Cette recommandation est similaire à la recommandation pour les paramètres libres dans d'autres algorithmes. Par exemple, dans PSO, il est recommandé que la somme des paramètres sociaux et cognitifs soit inférieure ou égale à quatre.
- $\boldsymbol{sign(r - 0.5)}$:contrôle la direction de l'exploration. Comme ***r*** est en **[0,1]** avec une distribution uniforme, il y a une probabilité égale de signes négatifs et positifs
- ***Probabilité de production (GP)*** **:** contrôle la probabilité de participation de la mise à jour de la concentration par le taux de production. ***GP = 1*** signifié qu'aucun terme de taux de production ne participera au processus d'optimisation. Cet État met l'accent sur une capacité d'exploration élevée et mène souvent à des solutions non récurrentes. ***GP = 0*** signifié que le terme du taux de production participera toujours au processus, ce qui augmente la probabilité de stagnation dans les optima locaux.

Sur la base de tests empiriques, ***GP = 0,5*** offre un bon équilibre entre les phases d'exploration et d'exploitation.

- ***Pool d'équilibre :*** Ce vecteur se compose de cinq particules. La sélection de cinq particules est un peu arbitraire, mais a été choisi sur la base de tests empiriques. Dans les itérations initiales, les candidats sont tous éloignés les uns des autres à distance. Mise à jour des sur ces candidats améliore la capacité de l'algorithme à rechercher globalement l'espace. La particule moyenne permet également de découvrir des espaces de recherche inconnues lors des itérations initiales lorsque les particules sont éloignées les unes des autres.

III.5.Capacité d'exploitation de l'EO

Les principaux paramètres et mécanismes d'exploitation et de recherche locale dans EO sont les suivants :

- $\mathbf{a_2}$**:** ce paramètre est similaire à $\boldsymbol{a_1}$, mais contrôle la fonction d'exploitation. Il détermine la quantité (ampleur) d'exploitation en creusant autour de la meilleure solution.
- $\boldsymbol{sign}(\boldsymbol{r} - \mathbf{0.5})$***:*** contrôle également la qualité d'exploitation (direction). Il spécifie la direction d'une recherche locale.
- ***Enregistrement de mémoire :*** enregistrement de mémoire, enregistre un certain nombre de particules et de substituts Cette fonction améliore directement la capacité d'exploitation de l'EO.
- ***pool d'équilibre :*** par lapse d'itération, l'exploration s'estompe et l'exploitation s'estompe. Ainsi, dans les dernières itérations, où les candidats à l'équilibre sont proches les uns des autres, le processus de mise à jour de la concentration facilitera la recherche locale autour des candidats, menant à exploitation.

III.6.Les caractéristiques d'EO

III.6.1.Les avantages

- Peut être simple et robuste.
- Peut converger rapidement.
- A une probabilité et une efficacité plus élevées.
- Testé EO par rapport à des repères mathématiques et d'ingénierie bien étudiés.
- Comparé l'algorithme à d'autres métaheuristiques bien connues.
- Efficacité et supériorité démontrées de la méthode proposée.

III.6.2.Les inconvénients

- Difficile de définir les paramètres initiaux.
- Ne peut pas résoudre le problème de la diffusion.

III.7.Résultats sur les fonctions de benchmark

La présente section démontre l'efficacité de l'algorithme proposé pour un ensemble de **58** fonctions de tests de benchmark, dont **23** fonctions unimodales, multimodales. Cette étude utilise les deux mesures de validation quantitatives et qualitatives. Les mesures quantitatives comprennent les valeurs d'écart-type pour différentes fonctions de tests et les mesures qualitatives comprennent la trajectoire, recherche, optimisation et historique moyen de conditionnement physique.

III.7.1.Problème de test d'optimisation mathématique

La figure (**III.4**) montre une version bidimensionnelle des trois catégories de fonctions mathématiques que cette étude utilise pour évaluer l'EO. La première catégorie comprend les fonctions unimodales (***F1-F7***) qui ont une seule solution optimale, ce qui défie délibérément la capacité d'exploitation de l'algorithme. La deuxième catégorie comprend les fonctions multimodales (***F8-F13***) qui ont plus d'une solution. Solutions optimales locales dans ces fonctions évaluaient la performance d'exploration de l'algorithme, alors qu'un algorithme doit être capable de rechercher globalement l'espace et d'éviter d'être piégé sur le système Optima local afin de trouver le système. La troisième catégorie comprend les fonctions multimodales (***F14-F23***), qui sont similaires aux fonctions multimodales, mais en faible valeur et fixent dimensions. Ces fonctions sont présentées dans le tableau suivant :

Tableau 1 : Les fonctions de Benchmark

Fonction	***Formule***	***Dimension***	***Domaine***
F1	$f_1(x) = \sum_{i=1}^{n} x_i^2$	*30*	[-100,100]
F2	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	*30*	[-100,100]
F3	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} x_j \right)^2$	*30*	[-100,100]
F4	$f_4(x) = max\{\lvert x_i \rvert, 1 \leqslant i \leqslant n\}$	*30*	[-100,100]
F5	$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	*30*	[-30,30]
F6	$f_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	*30*	[-100,100]
F7	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1]$	*30*	[-1.28,1.28]
F8	$f_8(x) = \sum_{i=1}^{n} -x_i sin\left(\sqrt{\lvert x_i \rvert}\right)$	*30*	[-500,500]
F9	$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	*30*	[-5.12,5.12]

F10	$$f_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right)$$	*30*	[-32,32]
F11	$$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$	*30*	[-600,600]
F12	$$f_{12}(x) = \frac{\pi}{n}\left\{10sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} U(x_i, 10, 100, 4)$$ $$y_1 = \frac{1}{4}(x_i + 1)$$ $$U(x_i, a, k, m) = k((x_i - a)^m)(x > a) + k\left((-x_i - a)^m(x < (-a))\right)$$	*30*	[-50,50]
F13	$$f_{13}(x) = 0.1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i)] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} U(x_i, 5, 100, 4)$$ $$U(x_i, a, k, m) = k((x_i - a)^m)(x > a) + k\left((-x_i - a)^m(x < (-a))\right)$$	30	[-50,50]
F14	$$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{n}(x_i - a_{ij})^6}\right)^{-1}$$	*2*	[-65.536, 65.536]
F15	$$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$$	*4*	[-5,5]
F16	$$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$	*2*	[-5,5]
F17	$$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosx_1 + 10$$	*2*	lb=[-5,0] ub=[10,15]

F18	$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3{x_1}^2 - 14x_2 + 6x_1x_2 + 3{x_2}^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12{x_1}^2 + 48x_2 - 36x_1x_2 + 27{x_2}^2)]$	*2*	[-2,2]
F19	$f_{19}(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	*3*	[0,1]
F20	$f_{20}(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	*6*	[0,1]
F21	$f_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	*4*	[0,10]
F22	$f_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	*4*	[0,10]
F23	$f_{21}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	*10*	[0,10]

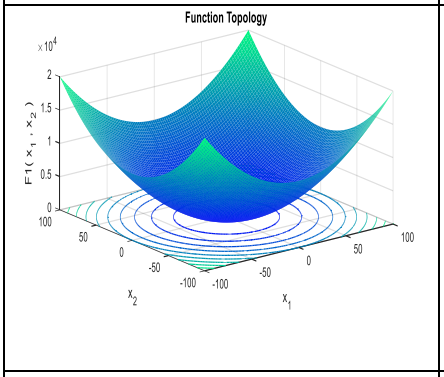
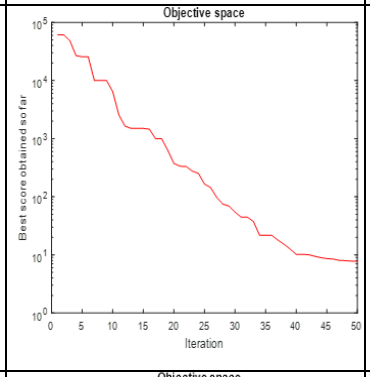
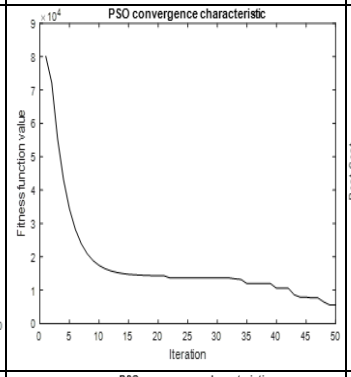
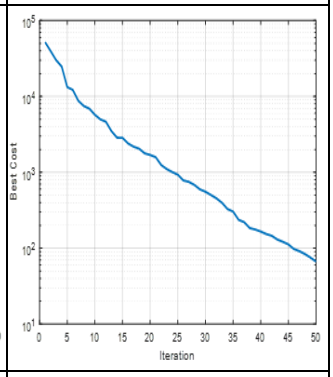
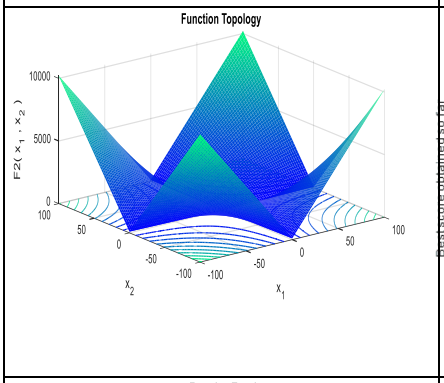
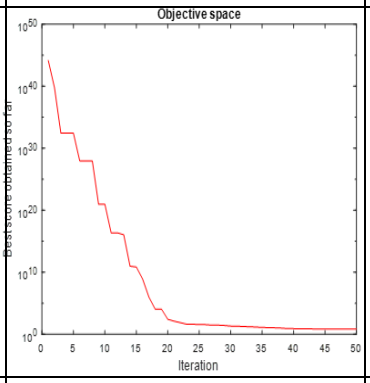
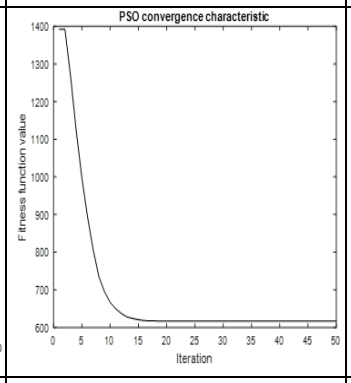
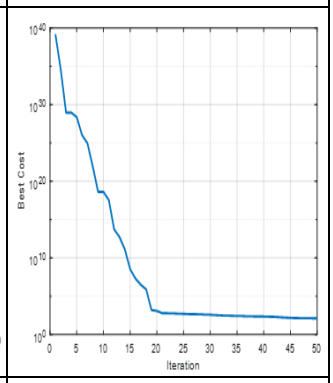
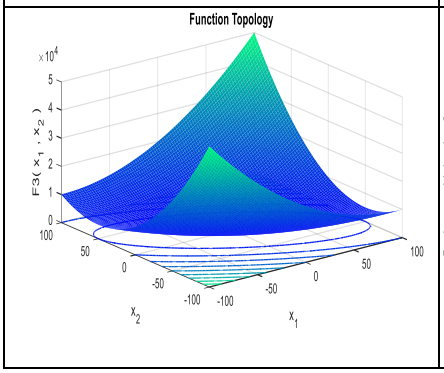
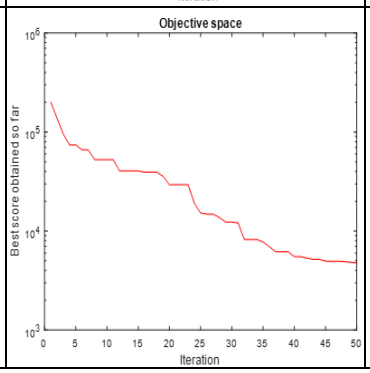
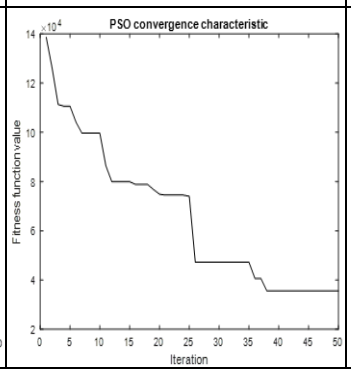
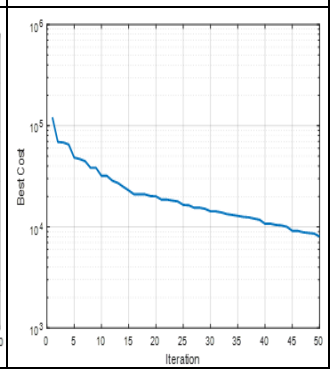
Pour toutes les catégories de tests, EO utilisent **20** particules avec **50** itérations (**1000** fonctions maximales évaluations). De même, pour fournir une comparaison équitable. Les autres méthodes utilisent également un maximum de **1000** évaluations fonctionnelles. Par exemple, PSO et FA utilisent **20** particules avec **50** itérations. ***Le tableau 2*** indique le réglage des paramètres pour chaque algorithme :

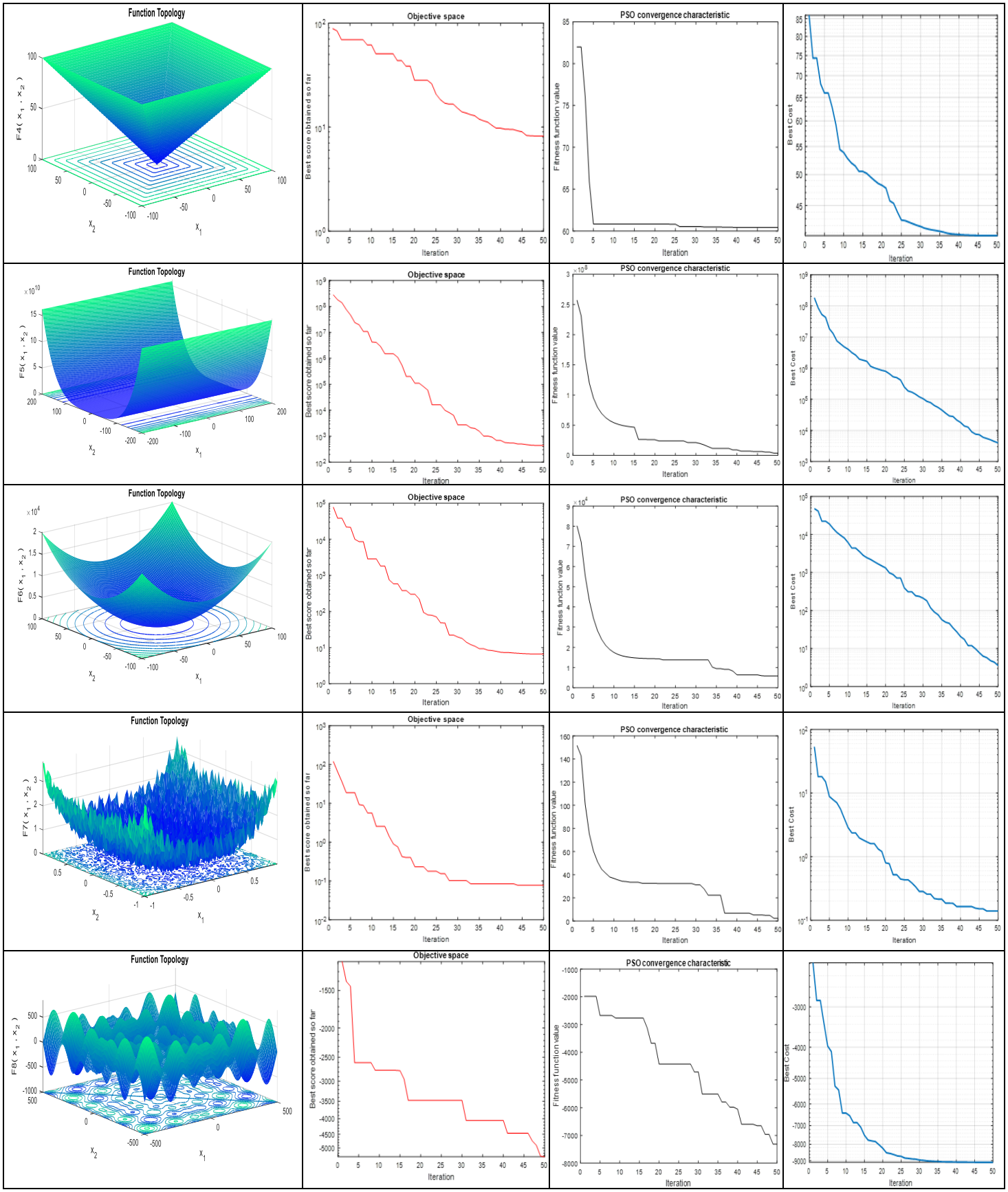
Tableau 2 : Les Paramètres des algorithmes (EO, PSO, FA)

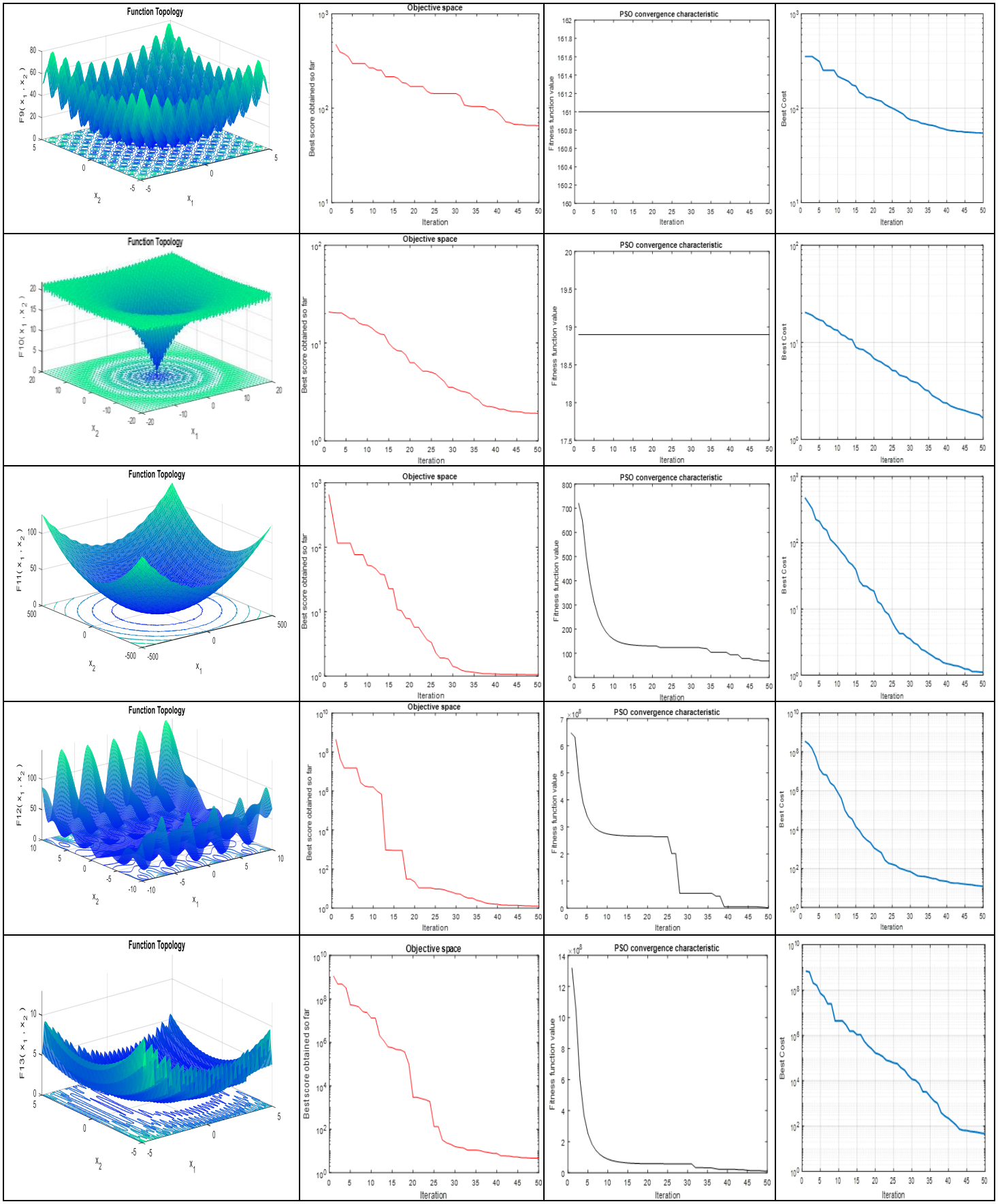
Algorithme	Paramètres	Valeurs
EO	a_1 valeur constante qui contrôle la capacité d'exploration a_2 valeur constante qui contrôle la capacité d'exploitation GP probabilité de génération	2 1 0.5
PSO	***Wmax*** Poids d'inertie ***Wmin*** Poids d'inertie Facteur d'accélération (c_1 , c_2)	0.9 0.4 (2.05, 2.05)
FA	***gamma*** Coefficient d'absorbation lumineuse ***beta*** Valeur de base du coefficient d'attraction ***alpha*** Coefficient de mutation ***alpha damp*** Taux d'amortissement du coefficient de mutation ***delta*** Plage de la mutation uniforme	1 2 0.2 0.98 delta=0.05*(VarMax-VarMin)

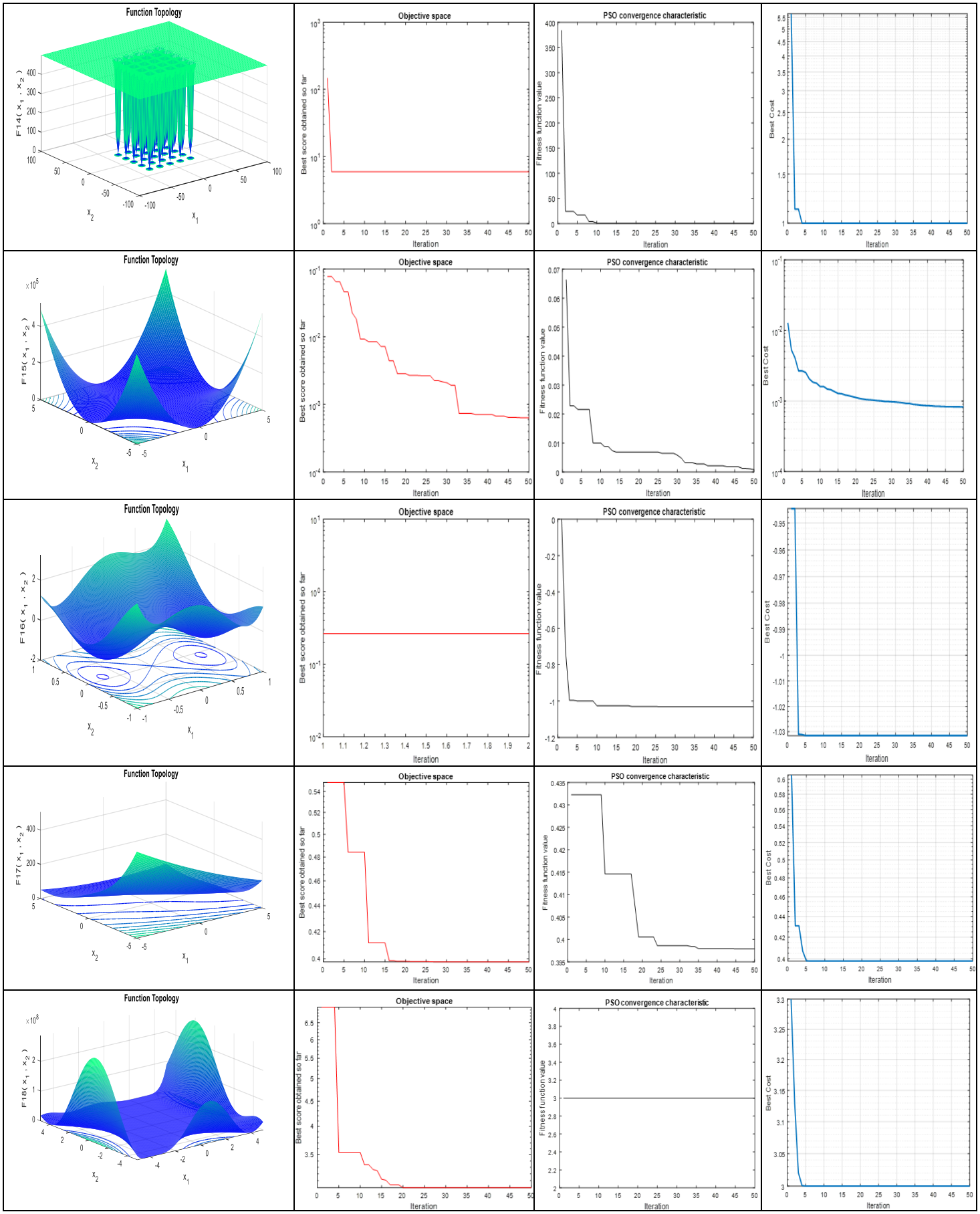
Dans les fonctions ***F1, F5, F6*** et ***F7*** la convergence rapide est dans le PSO jusqu'à l'itération 10, puis elle est lente, par contre la solution optimale on se trouve dans l'EO. Pour la fonction ***F3*** dans l'EO, PSO et FA la convergence est lente d'où la solution optimale est sur l'EO. Sur les fonctions ***F9, F10*** dans l'EO et FA la convergence est lente où la solution optimale est dans l'EO. Dans les fonctions ***F12, F13*** et ***F15*** la convergence rapide est sur le PSO jusqu'à l'itération 5, puis elle devient lente, d'où la solution optimale est dans l'EO, et dans les fonctions ***F17, F19*** et ***F20*** la convergence rapide est sur FA jusqu'à l'itération 5, ensuite elle devient stable où on trouve la solution optimale dans l'EO. Et pour les fonctions ***F2, F4, F11*** la convergence rapide est dans le PSO, D'où la solution optimale est sur le PSO. Ensuite les fonctions ***F21, F22*** et ***F23*** la convergence rapide est dans le FA jusqu'à l'itération 5 donc la solution optimale dans le PSO. Si on parle sur les fonctions ***F8, F16*** et ***F18*** on voit que la convergence rapide est dans le FA autant que l'itération 5 après cela elle devient stable alors la solution optimale est dans le FA. Par la suite dans la fonction ***F14*** la convergence rapide est sur le PSO et le FA avant que l'itération 5, au bout que la solution optimale est dans le FA.

Par conséquent, il convient de noter que, bien qu'il existe une convergence rapide en PSO ou FA, la solution optimale est en EO.

Fonction	EO	PSO	FA
			
			
			


Function Topology
Objective space
PSO convergence characteristic
Best score obtained so far
Fitness function value
Best Cost
Iteration
F4(x1 , x2)
F5(x1 , x2)
F6(x1 , x2)
F7(x1 , x2)
F8(x1 , x2)
x1
x2


Function Topology
F9(x1 , x2)
Objective space
Best score obtained so far
Iteration
PSO convergence characteristic
Fitness function value
Best Cost
F10(x1 , x2)
F11(x1 , x2)
F12(x1 , x2)
F13(x1 , x2)


Function Topology
Objective space
PSO convergence characteristic
Best score obtained so far
Fitness function value
Best Cost
Iteration
F14(x1 , x2)
F15(x1 , x2)
F16(x1 , x2)
F17(x1 , x2)
F18(x1 , x2)
x1
x2

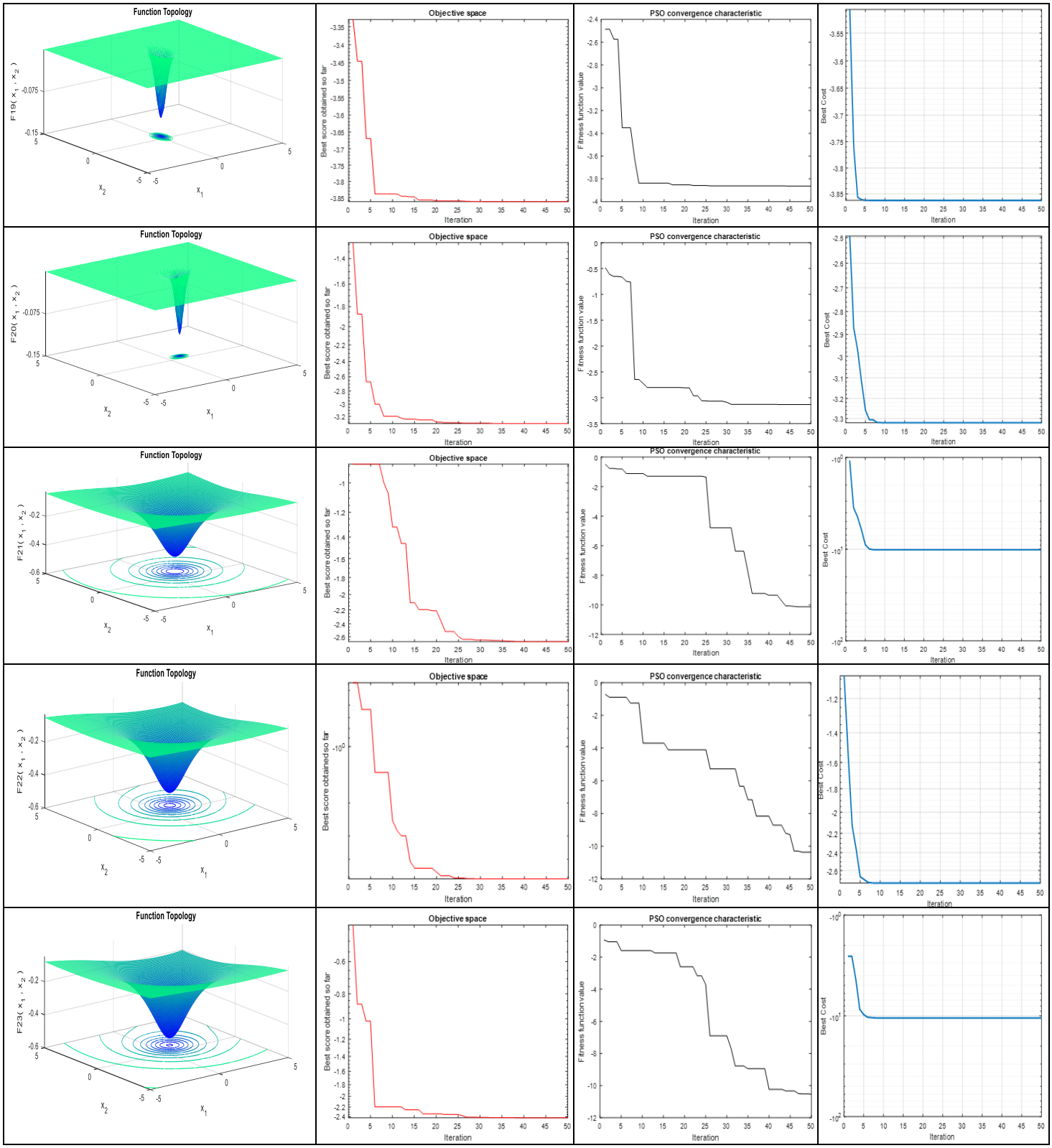


Figure III.4: Une vue en perspective bidimensionnelle des fonctions mathématiques de benchmark avec leurs courbes de convergences des algorithmes (EO, PSO, FA)

Tableau 3 : Résultats de l'optimisation et comparaison des fonctions

	Fonctions	Ave / Std temps	EO	PSO	FA
Fonctions Unimodales	F1	Ave	5.169638	23198.86	19.35984
		Std	2.649505	13076.45	14.5528
		Temps	0.966750 sec	2.355153 sec	18.392717 sec
	F2	Ave	9.111494	23198.86	108.3398
		Std	2.44102	13076.45	93.24715
		Temps	0.904097 sec	0.566523 sec	12.229624 sec
	F3	Ave	2331.724	65149.73	12217.52
		Std	1506.912	17405.51	5122.821
		Temps	1.774327 sec	1.772764 sec	18.834004 sec
	F4	Ave	6.263748	75.95218	41.10902
		Std	2.822619	9.245533	8.65218
		Temps	0.716928 sec	1.692278 sec	10.538231 sec
	F5	Ave	219.5263	3.418689e+07	5286.203
		Std	137.3678	2.718497e+07	4575.749
		Temps	0.774814 sec	0.685665 sec	10.622822 sec
	F6	Ave	12.98092	21965.02	35.90945
		Std	7.218266	11153.06	63.83359
		Temps	0.752518 sec	0.603138 sec	10.490704 sec
	F7	Ave	0.02339636	26.38723	0.1872303
		Std	0.01074385	18.83285	0.1135141
		Temps	0.901630 sec	0.752603 sec	11.573916 sec
Fonctions Multimodales (Dimension élevée)	F8	Ave	-5441.28	-6190.293	-8198.079
		Std	842.8608	705.9372	809.6488
		Temps	0.844418 sec	0.716292 sec	10.658139 sec
	F9	Ave	37.73968	195.9395	79.15112
		Std	20.43818	19.70677	24.50843
		Temps	0.778373 sec	0.626547 sec	9.888719 sec
	F10	Ave	1.442388	19.15423	2.447163
		Std	0.6129137	0.1301311	1.018192
		Temps	1.593979 sec	0.701377 sec	10.636160 sec
	F11	Ave	1.040546	196.0664	1.215959
		Std	0.04883541	96.36842	0.1423454
		Temps	0.908853 sec	0.751436 sec	15.223370 sec
	F12	Ave	0.7405778	17.70678	53.08226
		Std	0.5811345	6.95212	102.4844
		Temps	1.400427 sec	1.816016 sec	19.331462 sec
	F13	Ave	3.387255	17.70678	1071.098
		Std	1.182377	6.95212	1711.425
		Temps	1.292663 sec	0.782597 sec	18.519157 sec
	F14	Ave	2.678045	1.643236	1.492657
		Std	1.969052	1.125555	1.220029
		Temps	2.198007 sec	2.275627 sec	19.651870 sec
	F15	Ave	0.004648992	0.01221682	0.0007571264
		Std	0.008064598	0.009567974	0.0002320226
		Temps	0.750346 sec	0.484790 sec	9.986290 sec

F16	***Ave***	***-1.031628***	***-1.031535***	***-1.031628***
	Std	***6.58047e-11***	***0.000201378***	***6.813527e-07***
	Temps	***0.722311 sec***	***0.441983 sec***	***8.033341 sec***
F17	***Ave***	***0.3979074***	***0.475157***	***0.3978874***
	Std	***5.818618e-05***	***0.3455275***	***8.746202e-08***
	Temps	***0.795208 sec***	***0.381275 sec***	***7.850534 sec***
F18	***Ave***	***3***	***3.000031***	***3.000003***
	Std	***8.322018e-07***	***6.398396e-05***	***1.149365e-05***
	Temps	***0.704886 sec***	***0.410561 sec***	***8.944677 sec***
F19	***Ave***	***-3.862782***	***-3.345031***	***-3.862782***
	Std	***6.595446e-07***	***0.8044341***	***2.252855e-07***
	Temps	***0.798092 sec***	***0.582038 sec***	***9.127343 sec***
F20	***Ave***	***-3.230748***	***-1.170928***	***-3.274434***
	Std	***0.07300321***	***0.7775027***	***0.05975501***
	Temps	***0.870396 sec***	***0.552072 sec***	***10.457101 sec***
F21	***Ave***	***-7.126694***	***-5.929812***	***-7.033216***
	Std	***3.483608***	***3.003331***	***3.60577***
	Temps	***1.508642 sec***	***0.754120 sec***	***11.421180 sec***
F22	***Ave***	***-6.83958***	***-6.371771***	***-9.352942***
	Std	***3.699412***	***3.503384***	***2.569896***
	Temps	***1.030905 sec***	***0.877269 sec***	***11.735578 sec***
F23	***Ave***	***-8.715582***	***-6.441643***	***-10.53611***
	Std	***-8.715582***	***3.577949***	***0.001156579***
	Temps	***1.196765 sec***	***1.835269 sec***	***14.144086 sec***

Fonctions Multimodales (Dimension fixée)

Tel que :

Std **:** l'écart-type (Standard deviation en Anglais).

Ave **:** la moyenne (Average).

III.7.2.Discussion et résultats

Les fonctions unimodales sont conçues pour tester la capacité d'exploitation d'une méthode. D'après les résultats sur les fonctions unimodales du ***tableau 3*** (***F1-F7***), il est évident que l'EO a surpassé presque toutes les méthodes sur la majorité des fonctions. Comme on le voit, EO a pu atteindre le premier rang dans les fonctions de test unimodales. Cette performance supérieure se voit également sur la moyenne et l'écart-type dans toutes les fonctions. Par contre, le temps d'exécution de PSO a pris moins de temps par rapport les autres algorithmes dans les fonctions ***F2, F5, F6*** et ***F7***. Sur la base des caractéristiques des fonctions unimodales, on peut affirmer que l'EO bénéficie d'une capacité d'exploitation élevée.

Les fonctions multimodales peuvent évaluer la capacité d'exploration d'un algorithme en raison de leur nombre d'optimums locaux. Le nombre de solutions localement optimales dans ces types de fonctions augmentation exponentielle du nombre de dimensions. Les résultats d'EO sur les fonctions multimodales sont donnés dans ***le tableau 3*** pour les fonctions à dimensions élevées (***F8-F13***) et pour les dimensions fixes fonctions (***F14-F23***).

Le tableau montre que l'EO a surpassé les autres méthodes pour les problèmes de dimensions élevées ***F11, F12*** et ***F13*** sur la moyenne et l'écart type, et la durée minimale d'exécution est dans PSO pour ***F11*** et ***F13***. Par contre les fonctions ***F8, F9*** et ***F10*** EO ont affiché des résultats concurrentiels.

Pour les problèmes de dimensions fixées ***F16, F18*** et ***F19***, l'EO a surpassé les autres méthodes sur la moyenne et l'écart type et le temps d'exécution minimale sont dans le PSO. Pour ***F15, F17, F20*** et ***F22*** l'EO a atteint le deuxième meilleur rendement après celui de FA tel que l'EO ont affiché des résultats concurrentiels sur la moyenne et l'écart type et la durée minimale d'exécution sont dans PSO. Les fonctions ***F14, F21*** et ***F23*** ont affiché des résultats concurrentiels.

Dans les fonctions de test unimodales et multimodales nous remarquons que la durée d'exécution de FA est plus longue que les deux algorithmes, d'après les résultats des fonctions de benchmark l'EO est plus proche de la solution optimale par rapport aux PSO et FA.

Toutes ces fonctions nous avons l'étudier sous Matlab

III.8.Conclusion

Dans ce chapitre, nous avons examiné un nouveau algorithme d'amélioration c'est l'algorithme d'optimisation d'équilibre. Ou nous avons testé les fonctions de benchmark unimodales et multimodales au plus on a comparé là sur les trois algorithmes : EO, PSO et FA, puis nous avons analysé la convergence de chaque algorithme.

Enfin, nous avons constaté que le meilleur algorithme pour l'optimisation est l'EO.

Conclusion générale

Les approches d'optimisation inspirées par la nature récemment développée sont de bonnes techniques pour trouver des solutions globales pour des applications d'optimisation de la vie réelle.

L'objectif de ce travail a été l'étude des approches de résolution des problèmes d'optimisation et plus particulièrement le test des fonctions de benchmark afin de proposer une approche qui peut contribuer à la résolution de ce problème.

L'accent a particulièrement été mise sur l'utilisation des algorithmes bio inspiré (l'optimisation par essaim particulier "PSO", l'algorithme des lucioles "FA" et l'algorithme d'optimisation d'équilibre "EO").

Le but de ces algorithmes bio inspiré est de produire une solution efficace (optimale), L'algorithme d'optimisation d'équilibre c'est la méthode qui se trouve la meilleure solution de problème du test des fonctions de benchmark.

Cet algorithme, basé sur une comparaison de différentes méthodes métaheuristiques, permet de trouver rapidement la solution la plus proche de la solution optimale.

Comme perspectives à ce travail, nous souhaitons :

- L'hybridation des formules de représentation avec d'autres algorithmes locaux ou globaux, tel que le ABC, colonie de fourmis, recherche tabou,...etc.
- La recherche d'algorithmes d'implémentation de l'EO.
- L'hybridation de l'EO avec des algorithmes évolutionnaires d'optimisation multi objectif.
- L'application de l'EO pour des problèmes de l'ingénierie.

Bibliographie

[1] H. Osman, G. Laporte. Metaheuristics: A bibliography. Ann. Oper. Res. Vol. 63, N° 5, pp.513-623, 1996

[2] Riccardo poli, James Kennedy, and Tim Black well, Particle swarm optimization. Swarm Intelligence 1(1):33-57, 2007

[3] Russell.c.Ebehart, yuhui Shi, and James Kennedy swarm Intelligence. The Morgan Kaufmann Séries in Artificial Intelligence, Morgan Kaufmann, San Francisco, CA,USA ,2001

[4] Craig-W- Rynold. Floks.herd, and Schools: A Distributed behavioral model. Computer Graphics, 21(4):25-34, 1987.

[5] EUDES et RIOLAND, Optimisation par essaim particulaire pour un problème d'ordonnancement et d'affectation de ressources, 2007

[6] GOURGAND et KEMMOE, Particle Swarm Optimization: A study of particle displacement for solving continuous and combinatorial optimization problems, 2009.

[7] Abbas EL DOR, Perfectionnement des algorithmes d'Optimisation par Essaim Particulaire. Applications en segmentation d'images et en électronique, Thèse de doctorat, Université Paris-Est. 5 décembre 2012.

[8] Maurice Clerc and James Kennedy. The Particle Swarm –explosion, stability, and convergence in a multidimensionnel cpmlexe space.IEEE Trans.Evolutionary Computation, 6(1):58-73, 2002.

[9] Yang X.S,"Nature-Inspired Metaheuristic Algorithms". Luniver Press, UK. 2008.

[10] Ali Saoucha, N, K. Ghanem, and B. Benmammar. "On applying firefly algorithm for cognitive radio networks." Communications and Vehicular Technology in the Benelux (SCVT), 2014 IEEE 21st Symposium on. IEEE, 2014.

[11] Romana CAPOR-HROSIK, Adis ALIHODZIC, Milan TUBA, Mirjana VUKOVIC, Milenko PIKULA, "Firefly Algorithm for Constrained Optimization Problems". ISBN: 978 960-474-330-8.

[12] N. A. Saoucha, "Paramétrage des algorithmes génétiques pour l'optimisation de la QoS dans les réseaux radios cognitifs", thèse de magistère, Université de M'sila, mars 2013.

[13] Satyobroto Talukder, Mathematical Modelling and Applications of Particle Swarm Optimization.2010.

[14] Ghorpade-Aher J, Metre VA. PSO based multidimensional data clustering: a survey. Int J Comput Appl.2014; 87(16):41–48

[15] Dudeja C. Fuzzy-based modified particle swarm optimization algorithm for shortest path problems. Soft Comput. 2019; 23(17):8321.

[16] Mahesa R, Wibowo EP. Optimization of fuzzy c-means clustering using particle swarm optimization in brain tumor image segmentation. J Theor Appl Inf Technol. 2020; 98; 19.

[17] Samma, Hussein, Chee Peng Lim, and Junita Mohamad Saleh. "A New Reinforcement Learning-Based Memetic Particle Swarm Optimizer." Applied Soft Computing Journal 43 (Jun.): 276–297. doi:10.1016/j.asoc.2016.01.006

[18] Li, Ke, and Jitendra Malik. "Learning to Optimize Neural Nets." arXiv Preprint arXiv: 2017, 1703.00441.

[19] Xu, Yue, and Dechang Pi. "A Reinforcement Learning-Based Communication Topology in Particle Swarm Optimization." Neural Computing and Applications 32 (14): 10007–10032. Doi:.10.1007/s00521-019-04527-9, 2020.

[20] Samma, Hussein, Junita Mohamad-Saleh, Shahrel Azmin Suandi, and Badr Lahasan "Q-Learning-Based Simulated Annealing Algorithm for Constrained Engineering Design Problems." Neural Computing and Applications 32 (9): 5147–5161. doi:10.1007/s00521-019-04008-z. . 2020.

[21] Dominik Weikert , Sebastian Mai et Sanaz Mostaghim, (http://creativecommons.org/licenses/by/4.0/). 2020

[22] Xin-She Yang and Xingshi He. 'Firefly Algorithm: Recent Advances and Applications', Int. J. Swarm Intelligence, Vol. 1, No. 1, pp. 36–50. DOI: 10.1504/IJSI.2013.055801, 2013

[23] A. Y. Abdelaziz, S. F. Mekhamer, M.A.L. Badr, M. A. Algabalawy, The Firefly Meta-Heuristic Algorithms: Developments and Applications, International Electrical Engineering Journal (IEEJ) Vol. 6 (2015) No.7, pp. 1945-1952

[24] U. Singh and M. Rattan, "Design of thinned concentric circular antenna arrays using firefly algorithm", IET Microwaves, Antennas & Propagation, 2014.

[25] M.M. Othman, W. El-Khattam, A.Y. Abdelaziz, and Y.G. Hegazy, "A firefly based optimization algorithm for optimal planning of voltage controlled distributed generators", Engineering Optimization IV – Rodrigues et al. (Eds), 2015.

[26] X.-S. Yang, X.S. He, Why the firefly algorithm works? in: Nature-Inspired Algorithms and Applied Optimization (Edited by X.-S. Yang), Springer, pp. 245-259 (2018).

[27] Kaur, M., Ghosh, S., Network reconfiguration of unbalanced distribution networks using fuzzy-firefly algorithm, Applied Soft Computing, 49, 868–886 (2016).

[28] Singh, S.K., Sinha, N., Goswami, A.K., Sinha, N., Optimal estimation of power system harmonics using a hybrid firefly algorithm-based least square method, Soft Computing, 21(7), 1721–1734 (2017).

[29] Satapathy, P., Dhar, S., Dash, P.K., Stability improvement of PV-BESS diesel generatorbased microgrid with a new modified harmony search-based hybrid firefly algorithm, IET Renewable Power Generation, 11(5), 566–577 (2017).

[30] Kaushik, A., Tayal, D.K., Yadav, K., Kaur, A., Integrating firefly algorithm in artificial neural network models for accurate software cost predictions, Journal of Softwware: Evolution and Process, 28(8), 665–688 (2016).

[31] Ao Liu, Peng Li, Xudong Deng & Liang Ren ,A sigmoid attractiveness based improved firefly algorithm and its applications in IIR filter design, Connection Science, 33:1,1-25, DOI: 10.1080/09540091.2020.1742660, (2021)

[32] Singh, U., & Salgotra, R. Synthesis of linear antenna arrays using enhanced firefly algorithm. Arabian Journal for Science and Engineering, 44(3), 1961–1976. https://doi.org/10.1007/s13369-018-3214-2, (2019).

[33] Xin-She Yang, in Nature-Inspired Optimization Algorithms (second Edition), , Chapter 9 (firefly Algorithms). www.sciencedirect.com, (2021)

[34] A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, Knowledge-Based Systems (2019) 105190. https://doi.org/10.1016/j.knosys.2019.105190

Résumé

De nombreux algorithmes et métaheuristiques pour l'optimisation s'inspirent de la nature, par exemple l'algorithme d'optimisation de l'essaim de particules et l'algorithme des lucioles. Ces algorithmes permettent souvent de résoudre des problèmes d'optimisation de nature très diverse. Dans ce mémoire, nous proposons un nouvel algorithme métaheuristique, c'est l'algorithme d'optimisation d'équilibre inspiré par les modèles de bilan massique de volume de contrôle. Cet algorithme est très simple, il donne de bons résultats et peut facilement s'adapter à des problèmes d'optimisation divers. Nous l'appliquons le test des fonctions de benchmark pour les fonctions unimodales et multimodales. Enfin, nous discuterons des différences entre les trois algorithmes.

Mots clés : Métaheuristique, Optimisation par essaims particulaires, Algorithme des lucioles, Optimisation d'équilibre.

Abstract

Many algorithms and metaheuristics for optimization are inspired by nature, for example the optimization algorithm of the particle swarm and the firefly algorithm. These algorithms often solve optimization problems of a very diverse nature. In this paper, we propose a new metaheuristic algorithm, which is the equilibrium optimization algorithm inspired by the volume control mass balance models. This algorithm is very simple, it gives good results and can easily adapt to various optimization problems. We apply the benchmark function test for unimodal and multimodal functions. Finally, we will discuss the differences between the three algorithms.

Keywords : metaheuristics, the particle swarm optimization, the firefly algorithm, the equilibrium optimization.

ملخص

العديد من الخوارزميات والميتافيزيقا لتحقيق المستوى الأمثل مستوحاة من الطبيعة، على سبيل المثال خوارزمية الاستخدام الأمثل لسرب الجسيمات وخوارزمية الذبابة النارية. وغالبا ما تحل هذه الخوارزميات المشاكل المثلى ذات الطابع المتنوع جدا. في هذه المذكرة، نقترح خوارزمية جديدة للتوازن، وهي خوارزمية التوازن الأمثل المستوحاة من نماذج توازن كتلة التحكم في الحجم. هذه الخوارزمية بسيطة جدا، وتعطي نتائج جيدة ويمكن أن تتكيف بسهولة مع مختلف المشاكل المثلى. ونطبق اختبار الدالة المرجعية للوظائف الأحادية الواسطة والوسائط المتعددة. وأخيرا، سنناقش الاختلافات بين الخوارزميات الثلاث.

الكلمات المفتاحية : الميتافيزيقا، تحسين سرب الجسيمات، خوارزمية اليراعات، خوارزمية تحسين التوازن.