République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOHAMED KHIDER, BISKRA

FACULTÉ des SCIENCES EXACTES et des SCIENCES de la NATURE et de la VIE

DÉPARTEMENT DE MATHÉMATIQUES

Mémoire présenté en vue de l'obtention du Diplôme **:**

MASTER en Mathématiques

Option : **Analyse**

Par

BENNADJI Djihane

Titre :

L'algorithme de Pollinisation des Fleurs

Membres du Comité d'Examen :

Dr.	**CHEMCHAM Madani**	UMKB	Président
Dr.	**GHEDJEMIS Fatiha**	UMKB	Encadreur
Dr.	**SOLTANI Siham**	UMKB	Examinateur

Septembre 2020

Dédicace

Je dédie ce modeste travail :

À ceux qui m'ont encouragé par leurs prières et qui m'ont donné l'espoir et la valonté d'atteindre mes rêves, aux deux plus chers du monde : À mes parents .

À mes sœurs et mon frère.

REMERCIEMENTS

En premier lieu, je remercie Dieu de m'avoir donnée le courage pour réaliser ce modeste travail.

Je tiens à remercier ma directrice de recherche Mme **GHEDJEMIS FATIHA**, pour sa générosité et ses précieux conseils, ses encouragements, ses orientations et sa disponibilité au cours de notre recherche.

Mes remerciements vont également au membre de jury : Dr. Chemcham Madani et Dr. Soltani Siham pour avoir accepté juger ce mémoire.

Je remercie : Mes chers parents de leurs soutiens morals et financiers, mes sœurs : Merieme et Aridje, mon petit frère Said Amir et enfin mes copines : Nessrine, Isra, Lamia et Sana.

J'adresse aussi mes remerciements à tous les enseignants (es) du département de mathématique qui ont assuré ma formation.

Table des matières

Table des figures

Liste des tableaux

Introduction

L'être humain affrante chaque jours beaucoup de problème et veut les résoudre, depuis sont n'existance et l'essaye d'améliorer les méthodes utilisées pour le faire. Un des plus fréquente méthode est l'optimisation.

L'optimisation est une technique mathématique qui concerne la recherche de maximum ou minimum de fonctions dans une région réalisable. Il n'y a aucune entreprise ou industrie qui n'est pas impliquée dans la résolution des problèmes d'optimisation. Une variété de techniques d'optimisation rivalisent pour la meilleure solution. L'optimisation d'essaim des particules (OPS) est une méthode d'optimisation relativement nouvelle, moderne et puissante qui a été empiriquement montrée pour fonctionner bien sur beaucoup de ces problèmes d'optimisation. L'optimisation détermine la solution la mieux adaptée à un problème donné circonstancés. Par exemple, un gestionnaire doit prendre de nombreux plans technologiques et managériaux à plusieurs reprises. L'objectif final des plans est soit de minimiser l'effort requis, soit de maximiser les avantages souhaités. L'optimisation fait référence aux tâches de minimisation et de maximisation. Puisque la maximisation de toute fonction f est mathématiquement équivalente à la minimisation de son inverse additif $-f$, le terme minimisation et optimisation sont utilisés de manière interchangeable. Pour cette raison, de nos jours, il est très important dans de nombreuses professions. Il ya pleins de méthodes pour résoudre les problèmes d'optimisation, parmi eux des méthodes métaheuristiques.

Les "métaheuristiques" sont des procédures conçues pour résoudre des problèmes d'optimisation dits difficiles. Ce sont en général des problèmes aux données incomplètes, incertaines, bruitées ou confrontés à une capacité de calcul limitée. Les métaheuristiques ont connu le succès dans une large variété de domaines. Ce la découle du fait qu'elles peuvent être appliquées à tout problème pouvant

être exprimé sous la forme d'un problème d'optimisation de critère(s). Ces méthodes sont, pour la plupart, inspirées de la physique (recuit simulé), de la biologie (algorithmes évolutionnaires) ou de l'éthologie (essaims particulaires, colonies de fourmis).

De nombreux phénomènes dans la nature ont des caractéristiques uniques qui peuvent être utilisées et converties en un modèle mathématique ou même un algorithme pour résoudre des problèmes réels.

Au cours des dernières décennies, les chercheurs ont développé de nombreux algorithmes inspirés de la nature afin de tenter de trouver les meilleures solutions pour divers problèmes d'optimisation. Les exemples sont l'algorithme génétique (AGs), l'optimisation des essaims de particules (OEP), les algorithmes de colonies des fourmis. Ces algorithmes ont été appliqués avec succès à un large éventail de problèmes d'optimisation et sont largement utilisés dans la littérature de la métaheuristique au cours des deux dernières décennies. D'autre part, la nature a encore bien d'autres phénomènes qui peuvent être utilisés pour résoudre différents types de problèmes. Un phénomène est la stratégie de reproduction des plantes à fleurs par pollinisation, qui a inspiré $Yang$ en 2012 à proposer un nouvel algorithme appelé l'algorithme de pollinisation des fleurs (APF).

APF est une technique d'optimisation basée sur les essaims qui a attiré l'attention de nombreux chercheurs dans plusieurs domaines d'optimisation en raison de ses caractéristiques impressionnantes.

Dans ce mémoire on va étudier comment optimiser un problème a valeur intiale dans une équation différentielle ordinaire linéaire du premier ordre en basant sur l'algorithme de pollinisation des fleurs (APF).

Notre objectif est d'évaluer les performances d'(APF) et d'étudier ses propriétés. Jusqu'à présent, le créateur de l'algorithme n'a fait que contribuer à ce domaine. Les chercheurs établissent la forme de base d'(APF) et décrivent sa variante à plusieurs objectifs.

Ce mémoire est organisé de la manière suivante :

Chapitre 1 : Le premier chapitre présente une étude bibliographique où on expose le problème d'optimisation et les différentes métaheuristiques.

Chapitre 2 : Nous allons d'abord passer en revue brièvement les principales caractéristiques de la pollinisation des fleurs, et donc idéaliser ces caractéristiques en quatre règles. Nous allons ensuite les utiliser pour développer un algorithme de pollinisation des fleurs (APF). Ensuite, nous le validons en utilisant un ensemble de fonctions de test bien connues.

Chapitre 3 : Nous décrivons brièvement la fonction logarithme népérien, leurs définitions et leurs propriétés principales en plus les résultats d'expériences numériques à travers une étude de simulation d'un problème à valeur initiale (PVI) comme un exemple d'application dont les résultats obtenus par l'algorithme de pollinisation des fleurs (APF).

Chapitre 1

Métaheuristiques pour l'optimisation difficile

Une métaheuristique est un algorithme d'optimisation visant à résoudre des problèmes d'optimisation difficiles pour lesquels on ne connait pas de méthode classique plus efficace.

1.1 Problème d'optimisation

Un problème d'optimisation se définit comme la recherche, parmi un ensemble de solutions possibles S (appelé aussi espace de décision ou espace de recherche), de la (ou des) solution(s) x^* qui rend(ent) une fonction minimale (ou maximale) mesurant la qualité de cette solution. Cette fonction est appelée fonction objectif ou fonction coût. Si on pose $f : S \mapsto \mathbb{R}$ la fonction objectif à minimiser (respectivement à maximiser) à valeurs dans $\mathbb{R}$, le problème revient alors à trouver l'optimum $x^* \in S$ tel que $f(x^*)$ soit minimale (respectivement maximale).

Lorsqu'on veut résoudre un problème d'optimisation, on recherche la meilleure solution possible à ce problème, c'est-à-dire l'optimum global. Cependant, il peut exister des solutions intermédiaires, qui sont également des optimums, mais uniquement pour un sous-espace restreint de l'espace de recherche : on parle alors d'optimums locaux. La seule hypothèse faite sur S est qu'il s'agit d'un espace topologique, c'est à dire sur lequel est définie une notion de voisinage. Cette hypothèse est

nécessaire pour définir la notion de solutions locales du problème d'optimisation. On peut alors définir un optimum local (relativement au voisinage V) comme la solution x^* de S telle que $f(x^*) \leq f(x)\,;\, \forall\, x \in V\ (x^*)$ [4].

1.1.1 Optimisation globale

Un minimiseur global est défini comme x^* tel que

$$f\left(x^*\right) \leq f\left(x\right), \forall x \in S \tag{1.1}$$

où S est l'espace de recherche et $S = \mathbb{R}^n$ pour des problèmes sans contraintes.

Ici, le terme minimum global fait référence à la valeur $f\left(x^*\right)$, et x^* est appelé le minimiseur global. Certaines méthodes d'optimisation globale nécessitent un point de départ $z_0 \in S$ et il pourra trouver le minimiseur global x^* si $z_0 \in S$ [12].

1.1.2 Optimisation locale

Un minimiseur local x_L^* de la région L, est défini comme

$$f\left(x_L^*\right) \leq f\left(x\right), \forall x \in L \tag{1.2}$$

tel que $L \subseteq \mathbb{R}^n$.

Ici, une méthode d'optimisation locale devrait garantir qu'un minimiseur local x_L^* de l'ensemble L est trouvé.

Enfin, les techniques d'optimisation locale tentent de trouver un minimum local $f\left(x_L^*\right)$ et son minimiseur local x_L^* correspondant, tandis que les techniques d'optimisation globale tentent de trouver une valeur minimale ou minimale globale de fonction et son minimiseur global x^* correspondant [12].

Exemple 1.1.1 *: considérant une fonction* $y = f\left(x\right) = x^4 - 12x^3 + 47x^2 - 75x + 10$*, la figure*

suivante illustre la différence entre le minimiseur global x^* et le minimiseur local x_L^*

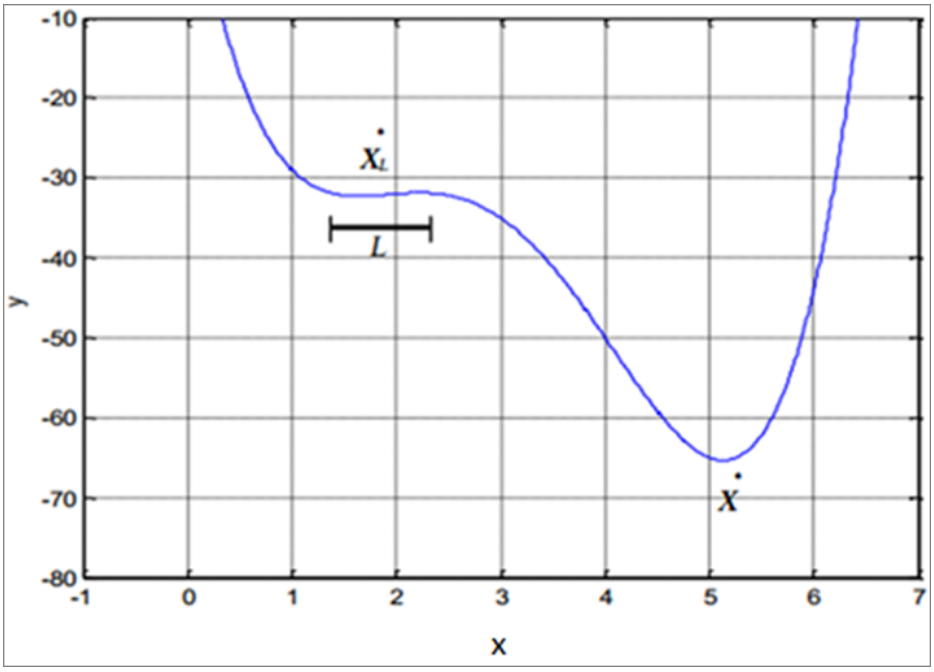


FIG. 1.1 – La différence entre le minimiseur global x^* et le minimiseur local x_L^*

1.2 Les types d'optimisation

On distingue trois types d'optimisation : avec contrainte, sans contrainte et dynamique.

1.2.1 Optimisation sans contrainte

Nous allons étudier le problème d'optimisation sans contraintes où on effectue la minimisation de la fonction $f : \mathbb{R}^n \to \mathbb{R}$ sur tout l'espace $\mathbb{R}^n$. Nous considérons donc le problème formulé de la façon suivante :

$$\{\min f(x)\,;\ x \in \mathbb{R}^n\} \tag{1.3}$$

qui peut réécrire sous la forme

{trouver $x^* \in \mathbb{R}^n$ tel que $f(x^*) \leq f(x)\,, \forall x \in \mathbb{R}^n$} [3].

1.2.2 Optimisation sous contrainte

Le problème d'optimisation avec contraintes s'écrit sous la forme :

$$\{\min f(x) \; ; \; x \in S\} \tag{1.4}$$

$\{$trouver $x^* \in S$ tel que $f(x^*) \leq f(x) \;\; , \forall x \in S\}$

S est l'ensemble des contraintes sous forme d'égalité et d'inégalité tel que pour (p, q) de $\mathbb{N}$.

$$S = \left\{ \begin{array}{ll} \forall x \in \mathbb{R}^n, & g_i(x) = 0, \quad i \in \{1, \ldots, p\} \\ & h_j(x) \leq 0, \quad j \in \{1, \ldots, q\} \end{array} \right\}, \tag{1.5}$$

tel que :

•La fonction $g(x) = (g_1(x), g_2(x), \ldots\ldots, g_p(x))$ est une fonction de plusieurs variables x de $\mathbb{R}^n$ à valeur dans $\mathbb{R}$, qui représente les contraintes en égalités.

•La fonction $h(x) = (h_1(x), h_2(x), \ldots\ldots, h_q(x))$ est une fonction de plusieurs variables x de $\mathbb{R}^n$ à valeur dans $\mathbb{R}$, qui représente les contraintes en inégalités [3].

Nous noterons ces types de problème ainsi :

-(PCE) Problème avec contraintes d'égalité.

-(PCI) Problème avec contraintes d'inégalité.

Exemple 1.2.1 *1. Quelle est la valeur maximale de la fonction f*

$$(A, B, C) = \sin A \times \sin B \times \sin C$$

sous la contrainte :

$$A + B + C = \pi$$

C'est un problème d'optimisation avec contraintes d'égalité

2. Maximiser la fonction objective

$$f(x_1, x_2) = 1200x_1 + 1000x_2$$

sous les contraintes :

$$3x_1 + 4x_2 \leq 160 \ ,$$

$$6x_1 + 3x_2 \leq 180,$$

$$x_1 \geq 0, x_2 \geq 0 \ ,$$

C'est un problème d'optimisation avec contraintes d'inégalité.

Exemple 1.2.2 *Minimiser la fonction*

$$f(x) = 100\,(x_2 - x_1)^2 + (1 - x_1)^2$$

sous les contraintes :

$$x_1 + x_2^2 \geq 0$$

$$x_1^2 + x_2 \geq 0$$

$$\text{avec } x_1 \in [-0.5, 0.5] \text{ et } x_2 \leq 1.0$$

Alors, l'optimum global est $x^* = (0.5, 0.25)$, avec $f(x^*) = 0.25$.

Remarque 1.2.1

1. *Plus généralement, on peut remplacer l'espace $\mathbb{R}^n$ par un espace vectoriel topologique sur $\mathbb{R}$ de dimension finie.*
2. *Pour la maximisation, on fait la minimisation de la fonction $(-f)$:*

$$\max f(x) = -\min f(-x)$$

1.2.3 Optimisation dynamique

De nombreux problèmes d'optimisation ont des fonctions objectives qui changent avec le temps et de tels changements dans la fonction objective entraînent des changements dans la position des optimums. Ces types de problèmes seraient des problèmes d'optimisation dynamique et définis comme minimisant $f(x, \varpi(t))$, $x = (x_1, x_2, \ldots, x_n)$, $\varpi(t) = (\varpi_1(t), \varpi_2(t), \ldots, \varpi_{n_\varpi}(t))$

sous les contraintes

$$\left\{ \begin{array}{ll} \forall x \in \mathbb{R}^n, & g_i(x) = 0, \quad i \in \{1, \ldots, p\} \\ & h_j(x) \leq 0, \quad j \in \{1, \ldots, q\} \end{array} \right\} \forall p, q \in \mathbb{N} \tag{1.6}$$

où $\varpi(t)$ est un vecteur de paramètres de contrôle de fonction objective dépendant du temps t, et $x^*(t)$ est l'optimum trouvé au pas de temps [12].

1.3 Les métaheuristiques d'optimisation

Le mot "métaheuristique" est dérivé de la composition de deux mots grecs :

- Heuristique qui vient du verbe heuriskein (euriskein) et qui signifie trouver.
- Méta qui est un suffixe signifiant au-delà, dans un niveau supérieur.

Les métaheuristiques d'optimisation sont des algorithmes généraux d'optimisation applicables à une grande variété de problèmes. Elles sont apparues à partir des années 80, dans le but de résoudre au mieux des problèmes d'optimisation difficile [7]. Les métaheuristiques ont rencontré le succés dans une large variété de domaines. Cela découle du fait qu'elles peuvent être appliquées à tout problème pouvant être exprimé sous la forme d'un problème d'optimisation de critère(s). Ces méthodes sont, pour la plupart, inspirées de la physique (récuit simulé), de la biologie (algorithme évolutionnaires) ou de l'éthologie (essaims particulaires, colonies de fourmis).

Dans ce qui suit, nous présentons brièvement les principales métaheuristiques d'optimisation. Nous les classons en deux catégories : métaheuristiques à solution unique et métaheuristiques à population de solutions [2].

1.3.1 Les métaheuristiques à solution unique

Les méthodes à solution unique, plus connues sous le nom de méthodes de recherche locale ou encore méthodes de trajectoire, sont basées sur l'évolution d'une seule solution dans l'espace de recherche. Typiquement, les méthodes de recherche locale démarrent d'une solution unique, puis à chaque itération, la solution courante est déplacée dans un voisinage local en espérant améliorer la fonction objectif. Les méthodes à solution unique englobent principalement la méthode de descente, la recherche locale itérée, la recherche à voisinage variable, la méthode de récuit simulé et la rechrche tabou [2].

1.3.2 Les métaheuristiques à population de solutions

À l'inverse des méthodes de recherche à solution unique, les métaheuristiques à population de solution sont des méthodes qui font évoluer simultanément un ensemble d'individus (solutions) dans l'espace de recherche, où chacun profite de l'expérience du groupe, de manière directe ou indirecte. Ces méthodes sont principalement inspirées du vivant. On peut distinguer deux catégories de métaheuristiques à population : les algorithmes évolutionnaires et les algorithmes d'intelligence en essaim [2].

Les algorithmes évolutionnaires

Les algorithmes évolutionnaires (AEs) sont des techniques de recherche inspirées par l'évolution biologique des espèces, apparues à la fin des années 1995. Les méthodes évolutionnaires ont d'abord suscité un intérêt limité, du fait de leur important coût d'exécution. Mais elles connaissent, depuis dix ans, un développement considérable, grâce à l'augmentation vertigineuse de la puissance des calculateurs, et notamment suite à l'apparition des architectures massivement parallèles, qui exploitent leur "parallélisme intrinsèque". Le principe d'un algorithme évolutionnaire se décrit simplement. Un ensemble de N points dans un espace de recherche, choisis a priori au hasard, constitue la population initial ; chaque individu x de la population possède une certaine performance, qui mesure son degré d'adaptation à l'objectif visé : dans le cas de la minimisation d'une

fonction objectif f, x est d'autant plus performant que $f(x)$ est plus petit. Nous avons représenté en figure 1.2 le principe d'un algorithme évolutionnaire [5].

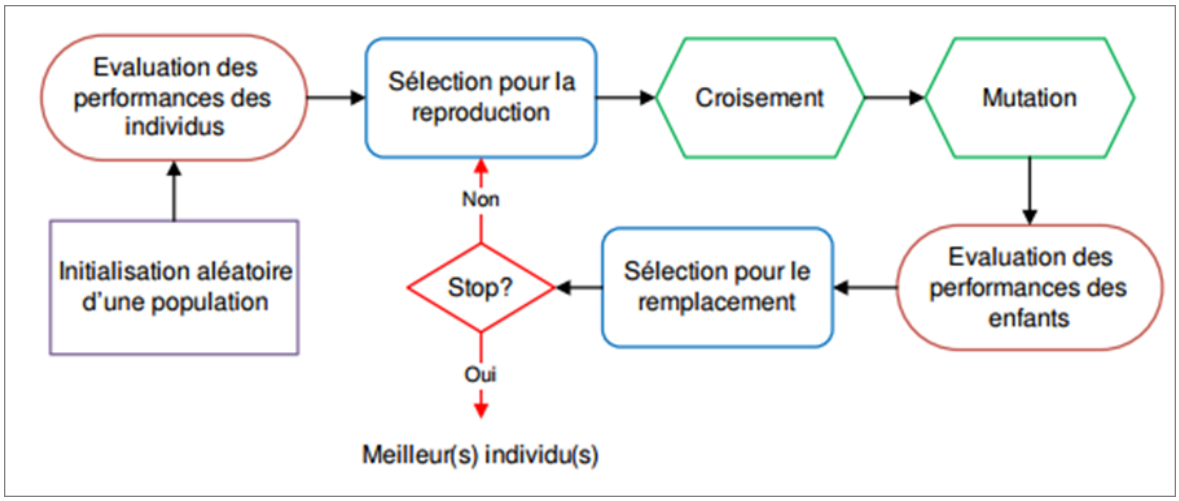


FIG. 1.2 – Le principe d'un algorithme évolutionnaire

Les algorithmes génétiques Les algorithmes génétiques (AGs) sont probablement les algorithmes évolutionnaires les plus populaires avec une gamme varié d'applications. La grande majorité des problèmes d'optimisation bien connus ont été résolus par des algorithmes génétiques. De plus, les algorithmes génétiques sont basés sur la population et de nombreux algorithmes évolutionnaires modernes sont directement basés sur ou ont de fortes similitudes avec les algorithmes génétiques.

Les algorithmes génétiques développés par *John Holland* et ses collaborateurs dans les années 1960 et 1970, sont un modèle ou une abstraction de l'évolution biologique basée sur la théorie de *Charles Darwin* de la selection naturelle [13].

Les algorithmes d'intelligence en essaim

Les comportements collectifs de certaines espèces, telles que les fourmis, les abeilles, les poissons , les oiseaux, ...etc. pour résoudre certains problèmes très complexes en effectuant de simples tâches individuelles, ont inspiré plusieurs chercheurs pour transposer ces comportements en algorithmes d'optimisation. Ces derniers sont nés de la modélisation mathématique des phénomènes éthologiques et biologiques. Les algorithmes d'intelligence en essaim qui ont connu le plus de succès sont

l'optimisation par essaim particulaire et les algorithmes de colonies de fourmis, que nous décrivons dans les sections ci-dessous [2].

La méthode des essaims particulaires L'optimisation par essaim particulaire (OEP) (Particle Swarm Optimization (PSO), en anglais) est une métaheuristique d'optimisation proposée pour la première fois par $J.Kennedy$ et $R.Eberhart$ en 1995 en s'inspirant des modélisations statistiques développées par [$Reynolds$, 1987] et [$Heppner\&Grenander$, 1990], qui permettent de simuler le déplacement de volées d'oiseaux et de bancs de poissons [2].

Dans l'algorithme d'essaim particulaire, la recherche s'effectue par une population d'individus appelés particules. Chaque particule survolant l'espace de recherche en quête de l'optimum global est considérée comme solution potentielle du problème. Afin de définir sa direction de vol, une particule se base sur deux types d'informations : une information tirée de sa propre expérience et une information tirée de l'expérience de l'essaim. Le mouvement des particules est régi par les équations suivantes :

$$V^{(k+1)} = w.V^{(k)} + c_1.\,\mathrm{rand}_1\,.(Pbest^{(k)} - X^{(k)}) + c_2.\,\mathrm{rand}_2\,.\left(Gbest^{(k)} - X^{(k)}\right) \tag{1.7}$$

$$X^{(k+1)} = X^{(k)} + V^{(k+1)} \tag{1.8}$$

où :

X : est la position des particules,

V : est la vitesse des particules,

w : est le paramètre d'inertie,

$Pbest$: est la meilleure position personnelle,

$Gbest$: est la meilleure solution de l'essaim,

rand_1, rand_2 : sont des variables aléatoires entre 0 et 1,

c_1, c_2 : sont des constantes positives,

k : est la variable d'itération.

Les trois termes de l'équation de vitesse peuvent être traduits comme suit :

1. $w.V^{(k)}$: représente une composante physique d'inertie, qui incite chaque particule à suivre sa direction courante de déplacement,
2. $c_1.\,\mathrm{rand}_1\,.\left(Pbest^{(k)} - X^{(k)}\right)$: représente une composante cognitive, qui incite la particule à revenir vers le meilleur site qu'elle a déjà visité,
3. $c_2.\,\mathrm{rand}_2\,.\left(Gbest^{(k)} - X^{(k)}\right)$: représente une composante sociale, qui incite la particule à se diriger vers le meilleur site trouvé par ses congénères.

La méthode des colonies de fourmis Les algorithmes de colonies de fourmis sont des métaheuristiques d'optimisation inspirées du comportement collectif des fourmis dans leur processus de recherche de nourriture et d'optimisation du chemin entre leur nid et la source de nourriture trouvée. L'optimisation des colonies de fourmis a été lancée par *Marco Dorigo* en 1992 et elle est basée sur le comportement de recherche de fourmis socials. Beaucoup d'insectes tels que les fourmis utilisent la phéromone comme messager chimique. Les fourmis sont des insectes sociaux et vivent ensemble dans des colonies organisées d'environ 2 à 25 *millions* de personnes lors de la recherche de nourriture, un essaim de fourmis ou d'agents mobiles interagit ou communique dans leur environnement local. Chaque fourmi dépose des produits chimiques odorants ou des phéromones pour communiquer avec les autres. Chaque fourmi est également en mesure de suivre l'itinéraire marqué par des phéromones posées par d'autres fourmis. Une fourmi trouve une source de nourriture, elle la marquera avec la phéromone et marquera également le chemin vers et depuis elle [13].

Cependant, la concentration de phéromones ϕ décroît ou s'évapore à un taux constant γ c'est-à-dire :

$$\phi\,(t) = \phi_0 \exp\left[-\gamma t\right], \tag{1.9}$$

où ϕ_0 est la concentration initiale à $t = 0$. Ici, l'évaporation est importante, car elle garantit la possibilité de convergence et d'auto-organisation.

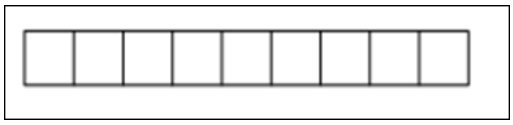
1.4 La distribution d'optimisation

Les distributions sont un nouveau consept mathématique généralisant la notion de fonctions, découvert par *Laurent Shwartz* en 1945. Dans notre travail on va utiliser deux distributions nécessaires qui sont : distribution uniforme et distribution de *Lévy*.

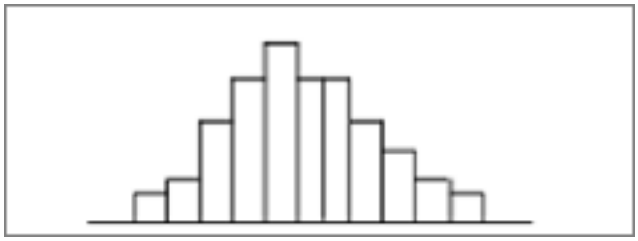
1.4.1 Distribution uniforme

Une distribution uniforme, parfois appelée rectangulaire, est une distribution où la probabilité d'occurrence est la même pour tous les valeurs de x , c'est-à-dire qu'elle a une probabilié constante. Par exemple, si un dé est lancé, la probabilité d'obtenir l'un des six résultats possibles est de 1/6. Maintenant, comme tous les résultats sont également probables, la distribution est uniforme.

Par conséquent, si une distribution uniforme est divisée en intervalles également espacés, il y aura un nombre égal de membres de la population dans chaque intervalle. La distribution est définie par $U(a, b)$, où a et b sont ses valeurs minimale et maximale respectivement [12].



Une distribution uniforme



Une distribution non uniforme

1.4.2 Distribution de Lévy

Nous utilisons le terme vol parce que nous sommes entrain de traiter des algorithmes qui sont basés sur le comportement des oiseaux. la nature n'a pas une distribution uniforme des sources. les butineuses, par le passé, ont besoin d'une stratégie pour trouver ces ressources non uniformément réparties de la manière la plus rapide pour leur survie. En général, le temps de recherche est un facteur limitant qui doit être optimisé pour la survie au quotidien de ces butineuses. la question

fondamentale des butineuses est quel est le moyen le plus rapide de trouver des ressources non uniformément réparties ? afin de répondre à cette importante question dans le comportement ecologique, une rafale d'études expérimentales et théoriques a été déclenchée. Ces études ont montré que le comportement de recherche de nombreux butineuses est généralement sporadique, comprenant des phases de recherche actives et qu'il alterne aléatoirement avec une phase de mouvement balistique rapide. L'éventail des stratégies de recherche varie de la stratégie de croisière à l'embuscade ou à la recherche assise et d'attente, où un butineur reste immobile pendant une longue période.

On considère N variables aléatoires $(X_1, X_2, ..., X_N)$ qui sont distribuées identiquement et qui ont la même fonction de densité. On appelle un processus stable un ensemble de variables aléatoires qui ont tous une fonction densité égale à la fonction densité de leur somme. Un processus Gaussien est un exemple de processus stable, et la distribution Gaussien a un deuxième moment fini. Il existe des classes de distribution de probabilité avec un deuxième moment infini et qui sont stables. Spécialement la distribution de probabilité de *Lévy* qui appartient à une classe spéciale des distributions α-stable et symétrique [15].

Une variable aléatoire α-stable symétrique est definie par leur fonction caractéristique suivante :

$$\phi_{\alpha,\sigma} = E\left[\exp\left(izS\right)\right] = \exp\left(-\sigma^{\alpha}\left|z\right|^{\alpha}\right) \tag{1.10}$$

Avec $E[]$ est l'opérateur d'attente, i est un nombre complexe, $z \in \mathbb{R}$, $\sigma \geq 0$ et $\alpha \in (0, 2]$ est le facteur escalier.

La distribution de probabilité de *Lévy* est donnée par la transformation de *Fourier* inverse de l'équation 1.10 comme :

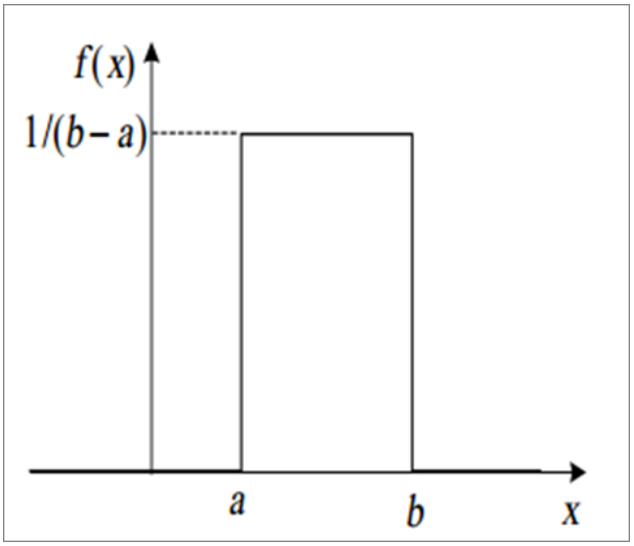
$$L_{\alpha,\gamma}\left(S\right) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\phi_{\alpha,\gamma}\left(z\right)\exp\left(izS\right)dz = \frac{1}{\pi}\int_{0}^{\infty}\exp\left(-\gamma z^{\alpha}\right)\cos\left(zS\right)dz$$

La fonction de densité de probabilité (FDP) et la fonction de distribution cumulative (FDC) pour une distribution uniforme continue sur l'intervalle $[a, b]$ sont respectivement

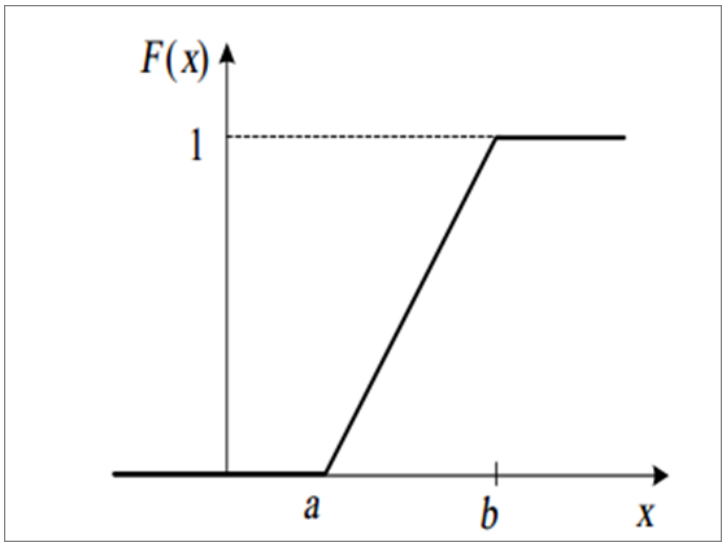
$$f(x) = \begin{cases} 0 & pour \quad x < a \\ \frac{1}{b-a} & pour \quad a \leq x \leq b \\ 0 & pour \quad x > b \end{cases} \tag{1.11}$$

et

$$F(x) = \begin{cases} 0 & pour \quad x < a \\ \frac{x-a}{b-a} & pour \quad a \leq x \leq b \\ 1 & pour \quad x > b \end{cases} \tag{1.12}$$



FDP uniforme



FDC uniforme

$U(0,1)$ est appelé une distribution uniforme standard.

Chapitre 2

L'algorithme de pollinisation des fleurs

L'algorithme de pollinisation des fleurs (APF) est l'un des algorithmes d'optimisation méta-heuristique inspirés de la nature basés sur la population les plus efficaces et sur le processus de pollinisation des plantes à fleurs. Avec la distribution de *Lévy*, l'APF peut contrôler l'équilibre des propriétés d'exploration et d'exploitation avec une probabilité de changement proposée. Cela conduit l'APF à échapper efficacement au piégeage local et à atteindre rapidement l'optimum global.

2.1 La pollinisation des fleurs dans la nature

La majorité des plantes sont des plantes à fleurs et il existe plus de 250000 espèces de plantes à fleurs dans la nature, où la pollinisation représente la principale stratégie de reproduction des plantes. La pollinisation est un processus de transfert de pollen d'une fleur à une autre par le vent ou des pollinisateurs tels que les insectes, les papillons, les abeilles, les oiseaux et les chauves-souris. Les plantes à fleurs ont évolué pour produire du nectar afin d'attirer les pollinisateurs et d'assurer la pollinisation. De plus, certains pollinisateurs et espèces végétales tels que les colibris et les plantes à fleurs ornithophiles forment une constance florale coévolutive, par exemple, certaines fleurs ne peuvent qu'attirer et ne peuvent dépendre que d'une espèce spécifique d'insectes pour une pollinisation réussie. Sur la base des principales caractéristiques de la pollinisation, l'algorithme

de pollinisation des fleurs a été développé [1].

2.2 Caractéristiques de la pollinisation des fleurs

Avant de décrire en détail l'algorithme de pollinisation des fleurs, examinant brièvement la forme de base de la pollinisation chez les plantes à fleurs. La pollinisation prend deux formes fondamentales : biotique ou abiotique [1].

2.2.1 Pollinisation biotique et croisée

La principale forme de pollinisation est la pollinisation biotique, et qui se fait avec la pollinisation croisée qui se forme, par des pollinisateurs tels que les insectes et les oiseaux et autres. Près de 90% des plantes à fleurs utilisent cette forme de pollinisation. Comme les pollinisateurs se déplacent et volent même à des rythmes et des vitesses variés, le mouvement du pollen peut être assez long. Une telle pollinisation peut également être considérée comme une pollinisation globale avec des propriétés potentielles des vols de *Lévy*. Si le pollen est codé comme vecteur de solution, cette action peut être équivalente à une recherche globale [1].

Pollinisation biotique : les pollens se transfert avec des pollinisateurs.

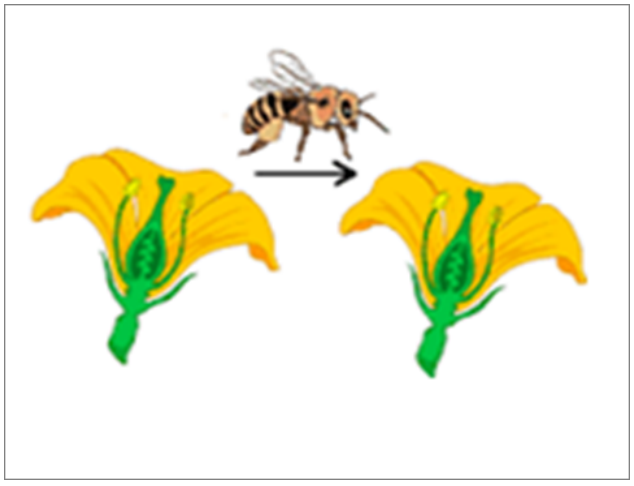


FIG. 2.1 – Pollinisation croisée

2.2.2 Pollinisation abiotique, auto-pollinisation

Une autre forme de pollinisation est la pollinisation abiotique, qui se fait généralement avec l'autofécondation, c'est-à-dire la fécondation de la même fleur, par les pollens d'elle même ou d'une autre fleur de la même plante, qui ne nécessite pas de pollinisateurs. On estime qu'environ 10% des plantes florales adoptent cette forme de pollinisation. Comme la pollinisation a tendance à être locale et à auto-pollinisation, elle peut être obtenue par le vent et la diffusion. La distance parcourue par un tel mouvement local est généralement courte, et une telle action peut donc être considérée comme une recherche locale [1].

Pollinisation abiotique : le pollen se transfert avec l'aide du vent ou d'eau.

FIG. 2.2 – Auto-pollinisation

2.2.3 Constance des fleurs

Parfois, il est avantageux pour les plantes et les pollinisateurs comme les colibris de former un partenariat pour économiser de l'énergie avec un succès garanti. Par conséquent, la constance des fleurs a évolué. Dans ce cas, les pollinisateurs ne visitent qu'un ensemble fixe de types de fleurs sans gaspiller d'énergie pour explorer de nouveaux types de fleurs, tandis que les plantes à fleurs évoluent pour fournir une récompense en nectar suffisante aux pollinisateurs afin d'encourager les visites fréquentes des pollinisateurs et ainsi maximiser leur succès de reproduction [1].

Les caractéristiques ci-dessus ont été utilisées pour concevoir un algorithme d'optimisation, appelé algorithme de pollinisation des fleurs (APF). Les principales caractéristiques et les composants de

Pollinisation des fleurs	Composants d'optimisation (en APF)
Pollinisateurs (insectes, papillons, oiseaux)	Déplacements / modification de variables
Biotique	Recherche globale
Abiotique	Recherche locale
Vols de Lévy	Tailles de pas (obéissant à une loi de puissance)
Pollen / fleurs	Vecteurs de solution
Constance des fleurs	Similitude dans les vecteurs de solution
Evolution des fleurs	Evolution itérative des solutions
Reproduction optimale des fleurs	Ensemble de solutions optimales

TAB. 2.1 – Pollinisation et ses composants d'optimisation

l'algorithme d'APF peuvent être résumés dans Table 2.1, qui montre la relation ou l'équivalence entre les termes d'optimisation et le contexte de la fleur. Avec ces composants et caractéristiques, nous pouvons maintenant décrire en détail l'algorithme de pollinisation des fleurs standard [1].

2.3 Algorithme de pollinisation des fleurs

L'APF est un algorithme inspiré de la nature qui imite le principal comportement de pollinisation des plantes à fleurs. Les quatres règles d'idéalisation ont été utilisées par $Yang$ en 2012 et peuvent être résumées comme suit :

Règle 1 La pollinisation globale implique une pollinisation biotique et croisée où les pollinisateurs transportent le pollen sur la base des vols de *Lévy*.

Règle 2 La pollinisation locale implique abiotique et auto-pollinisation.

Règle 3 La constance des fleurs peut être considérée comme une probabilité de reproduction proportionnelle à la similitude entre deux fleurs quelconques.

Règle 4 La probabilité de commutation $p \in [0, 1]$ peut être contrôlée entre la pollinisation locale et la pollinisation globale en raison de certains facteurs externes, tel que le vent. La pollinisation locale a une fraction significative p dans l'ensemble des activités de pollinisation.

Pour illustrer le mécanisme de l'APF basé sur ces quatre règles, trois étapes clés peuvent être décrites dans les trois sous-sections suivantes [1].

2.3.1 Recherche globale d'APF

Comme mentionné ci-dessus, les pollinisateurs tels que les oiseaux et les chauves-souris peuvent transférer le pollen sur de longues distances pendant la pollinisation biotique, garantissant ainsi la diversité et la pollinisation les plus adaptées à la reproduction. Par conséquent, la première (Règle 1) et la troisième (Règle 3) règle d'APF peuvent être formulées mathématiquement comme suit :

$$x_i^{t+1} = x_i^t + L\left(x_i^t - g_*\right) \tag{2.1}$$

où x_i^t est le pollen ou vecteur de solution à l'itération t et g_* est la meilleure solution trouvée parmi toutes les solutions à l'itération actuelle. Le paramètre L est la force de pollinisation, qui est essentiellement une taille de pas. Parce que les pollinisateurs se déplacent sur de longues distances avec différents intervalles de distance, le vol de *Lévy* peut être un simulateur efficace pour cette caractéristique ; c'est-à-dire que L peut être tiré d'une distribution de *Lévy* comme suit :

$$L \sim \frac{\lambda \Gamma(\lambda) \sin(\lambda\pi/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0) \tag{2.2}$$

où $\Gamma(\lambda)$ désigne la fonction gamma standard et cette distribution est valide pour les grands pas $s > 0$. Normalement, il est recommandé d'utiliser $\lambda = 1.5$ [1].

2.3.2 Recherche locale d'APF

Comme la pollinisation abiotique se produit par le vent ou la diffusion d'eau sans aucun pollinisateur, la pollinisation locale (Règle 2) et la constance des fleurs (Règle 3) peuvent être représentées comme suit :

$$x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right) \tag{2.3}$$

où x_j^t et x_k^t sont des pollens de fleurs différentes du même type de plante. Cette équation imite essentiellement la constance des fleurs dans un voisinage limité. Mathématiquement parlant, si

x_j^t et x_k^t sont de la même espèce qui peut être sélectionnée dans la même population, l'équation devient une marche aléatoire locale si nous tirons ε d'une distribution uniforme dans $[0, 1]$, et le nouveau vecteur de solution généré ne sera pas trop loin des solutions existantes [1].

2.3.3 Probabilité de commutation dans APF

Bien que nous ayons simulé à la fois la pollinisation biotique et abiotique, nous n'avons pas pris en compte le pourcentage et la fréquence de chaque type de pollinisation. Pour imiter cette fonctionnalité, nous utilisons une probabilité de commutation (Règle 4), où la valeur de p détermine si la modification de la solution suit la pollinisation locale ou globale. Bien qu'une valeur naïve de $p = 0,5$ puisse être utilisée, lorsqu'une valeur plus réaliste et efficace de $p = 0,8$ donne de meilleures performances (que $p = 0,5$) pour la plupart des applications .

La figure suivante montre l'organigramme d'APF. Trois étapes clés peuvent être résumées dans le pseudocode APF présenté dans l'algorithme 1 [1].

Pseudo-code d'(APF)

Objectife min ou max $f(x)$, $x = (x_1, x_{2,...})$

Initialiser la population de n fleurs/pollen gamètes avec des solutions

Trouver la meilleure solution g_* dans la population initiale

Définir une probabilité de commutation $p \in [0; 1]$

Tandis que $(t < MaxGeneration)$

Pour $i = 1 : n$ (toutes les n fleurs de la population)

Si $rand < p$

Dessiner un vecteur pas L (dimensionnel d) qui obéit à une distribution de *Lévy*

Pollinisation globale via $x_i^{t+1} = x_i^t + L(x_i^t - g_*)$

Autre

Dessiner ε à partir d'une distribution uniforme dans $[0, 1]$

Choisissez au hasard j et k parmi toutes les solutions

Pollinisation locale via $x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right)$

Fin Si

Evaluer de nouvelles solutions

Si de nouvelles solutions sont meilleures, mettez-les à jour dans la population

Fin pour

Trouver la meilleure solution actuelle g_*

Fin Que

TAB. 2.2 – Pseudo code d'APF

2.4 Fonctions de test

Toute nouvelle optimisation devrait être largement validée et comparée à d'autres algorithmes. Il y a beaucoup de fonctions de test, au moins plus d'une centaine de fonctions de test bien connues cependant, il n'y a pas d'ensemble convenu de fonctions de test pour valider de nouveaux algorithmes, bien qu'il y ait une revue et de la littérature. Dans ce travail, nous choisirons un

sous-ensemble diversifié de ces fonctions de test pour valider notre algorithme de pollinisation des fleurs proposé (APF).

En outre, nous allons également comparer les performances de notre algorithme avec celle des algorithmes génétiques et de l'optimisation des essaims de particules [14].

– La fonction *Ackley* peut être écrite comme

$$f(x) = -20\exp\left[-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right] - \exp\left[\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right] + 20 + e \ , \tag{2.4}$$

qui a un minimum global $f_* = 0$ à $(0, 0, ..., 0)$.

– Fonction de *Rastrigin*

$$f(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right], \quad -5.12 \leq x_i \leq 5.12 \ , \tag{2.5}$$

dont le minimum global est $f_* = 0$ à $(0, 0, ..., 0)$.

– Fonction de *Rosenbrock*

$$f(x) = \sum_{i=1}^{n-1}\left[(x_i - 1)^2 + 100\left(x_{i+1} - x_i^2\right)^2\right] \ , \tag{2.6}$$

dont le minimum global $f_* = 0$ se produit à $x_* = (1, 1, ..., 1)$ dans le domaine $-5 \leq x_i \leq 5$ où $i = 1, 2, ..., n$. Dans le cas $2D$, il est souvent écrit comme

$$f(x, y) = (x - 1)^2 + 100\left(y - x^2\right)^2 \ . \tag{2.7}$$

Fonctions/Algorithmes	AG	PSO	APF
Ackley ($d = 128$)	$32720 \pm 3327(90\%)$	$23407 \pm 4325(92\%)$	$3357 \pm 968(100\%)$
Rastrigin	$110523 \pm 5199(77\%)$	$79491 \pm 3715(90\%)$	$10840 \pm 2689(100\%)$
Rosenbrock ($d = 16$)	$55723 \pm 8901(90\%)$	$32756 \pm 5325(98\%)$	$5532 \pm 1464(100\%)$

TAB. 2.3 – Comparaison des performances de l'algorithme en termes de nombre d'itérations

2.5 Les propriétés de l'algorithmes de pollinisation des fleurs :

- Les algorithmes metaheuristiques utilisent des stratégies pour effectuer efficacement un processus de recherche et pour explorer l'espace de recherche de manière à obtenir des solutions presque optimales ou optimales.
- Les mécanismes de recherche en algorithmes mathématiques vont du processus de recherche simple locale au processus d'apprentissage complexe.
- Ils fournissent des solutions approximatives et sont généralement non déterministes.
- Ils peuvent incorporer des mécanismes d'échappement tels que l'étirement afin d'éviter d'être piégé dans des zones confinées de l'espace de recherche.
- Ils ne sont pas spécifiques au problème.
- Efficacité.
- Temps de calcul raisonnable.
- Grande flexibilité.
- Formulation mathématique intuitive.
- Capacité de gérer des informations incertaines, stochastiques et dynamiques.
- En raison de leur nature de proposition générale, ils peuvent être appliqués à un large éventail de problèmes.
- Ils sont également appelés algorithmes de boîte noire car ils exploitent des connaissances limitées sur le problème à résoudre.
- Comme aucune information de gradient ou de matrice de Hessienne n'est requise pour leur fonctionnement, ils sont également appelés algorithmes sans dérivés ou d'ordre zéro. Le terme d'ordre zéro implique que pour établir le vecteur de recherche seules les valeurs de fonction sont utilisées. De plus, la fonction à optimiser ne doit pas nécessairement être continue ou différenciable mais peut s'accompagner d'un ensemble de contraintes.

Chapitre 3

Etude Experimentale

Dans ce chapitre on utilise le logiciel Matlab pour tester l'efficacité de l'APF à travers une étude de simulation avec une comparaison entre les résultats obtenus par l'APF, les résultats exactes et celles qui sont obtenus par l'APF puis avec les résultats obtenus par la méthode *Euler* et qui est considérée comme un outil classique de résolution numérique d'un problème de valeur initiale (PVI).

3.1 Formulation du problème

Soit $f = f(x, y)$ une fonction à valeur réelle de deux variables réelles définie pour $a \leq x \leq b$, où a et b sont finis, et pour toutes les valeurs réelles de y. Les équations

$$\begin{cases} y' &= f(x, y) \\ y(a) &= y_0 \end{cases} \tag{3.1}$$

sont appelés problème de valeur initiale (PVI) ; ils symbolisent le problème suivant :

Trouver une fonction $y(x)$ continue et différentiable pour $x \in [a; b]$ telle que $y' = f(x; y)$ à partir de $y(a) = y_0$ et pour tout $x \in [a; b]$. Ce problème possède une solution unique lorsque : f est continu sur $[a; b] \times \mathbb{R}$, et satisfait la condition de Lipschitz ; il existe une constante $k \geq 0$ comme pour tout $x \in [a; b]$ et tout couple $(\theta_1; \theta_2) \in \mathbb{R} \times \mathbb{R}$. La recherche numérique des solutions optimales d'un

problème de valeur initiale (PVI) s'obtient avec des approximations :$y(x_0+h);\ldots;y(x_0+nh)$ où $a=x_0$ et $h=\frac{(b-a)}{(n+1)}$.

Pour plus de précision sur la solution, nous devons utiliser une très petite taille de pas h, qui comprend un plus grand nombre d'étapes, donc un temps de calcul plus long qui n'est pas disponible dans les méthodes numériques utiles comme les méthodes d'$Euler$ et de $Runge-Kutta$, ce qui est insuffisant et, par conséquent, des méthodes d'optimisation globales sont utilisées [10].

3.1.1 Fonction objective

L'idée principale pour obtenir notre fonction objective est d'utiliser la formule de la différence finie pour f est d'utiliser la formule de la différence finie pour le premier dérivé de f et l'équation 3.1 alors

$$\frac{y\left(x_j\right)-y\left(x_{j-1}\right)}{h}\approx f\left(x_{j-1},y\left(x_{j-1}\right)\right)\ .$$

Ainsi,

$$\frac{y_{j.}-y_{j-1}}{h}\approx f\left(x_{j-1},y_{j-1}\right)\ .$$

On considère la formule d'erreur :

$$\left[\frac{y_{j.}-y_{j-1}}{h}-f\left(x_{j-1},y_{j-1}\right)\right]^2\ .$$

Par conséquent la fonction objective, associée à $Y=(y_1,y_2,\ldots,y_d)$ sera :

$$F(y)=\sum_{i=1}^{d}\left[\frac{y_{j.}-y_{j-1}}{h}-f\left(x_{j-1},y_{j-1}\right)\right]^2 \tag{3.2}$$

3.1.2 La cohérence

Nous nous intéressons au calcul de $Y = (y_1, y_2, \ldots, y_d)$ qui minimise la fonction objective dans 3.2. Nous avons la formule de *Taylor* d'ordre 1 :

$$y_j = y_{j-1} + hy'_{j-1} + O\left(h^2\right),\ j = 1, \ldots, d$$

Alors,

$$\frac{y_j - y_{j-1}}{h} = y'_{j-1} + O\left(h\right)$$

Si on soustrait $f\left(x_{j-1}, y_{j-1}\right)$ des deux côtés de la dernière équation, on obtient

$$\frac{y_j - y_{j-1}}{h} - f\left(x_{j-1}, y_{j-1}\right) = y'_{j-1} - f\left(x_{j-1}, y_{j-1}\right) + O\left(h\right),\ j = 1, \ldots, d.$$

La dernière relation montre que la valeur finale $Y = (y_1, y_2, \ldots, y_d)$ est une solution approximative de (PVI), pour une petite valeur de h.

3.2 Erreur absolue

Si x_t est la valeur exacte d'une quantité et x_a est la valeur approximative de la même quantité alors l'erreur absolue Δ_a est définie par

$$Erreur\ absolue = |valeur\ exacte - valeur\ approximative|$$

c-à-d

$$\Delta_a = |x_t - x_a|$$

L'important dans le calcul d'erreur est l'ampleur de l'erreur, pas le signe car il peut être positif ou négatif. Par conséquent, le concept d'erreur absolue entre en jeu. Mais parfois, une erreur absolue peut ne pas donner son impact correct car elle ne tient pas compte l'ordre de grandeur de la valeur

à l'étude.

Les erreurs dans la solution numérique proviennent de deux sources : les erreurs de troncature et les erreurs d'arrondi. Les erreurs d'arrondi sont celles qui se produisent car l'arithmétique n'est pas effectuée exactement. Même si l'arithmétique était exacte, des erreurs de troncature se produiraient car la méthode d'intégration elle-même n'est qu'une approximation. Toutes les méthodes discutées ici sont convergentes, ce qui signifie que (en supposant une arithmétique exacte) l'erreur peut être maintenue arbitrairement petite en choisissant des tailles de pas suffisamment petites [9].

3.2.1 L'erreur globale

L'erreur globale en un point de la solution numérique est la différence par rapport à la valeur exacte,

$$\varepsilon_j = y\left(x_j\right) - y_j \tag{3.3}$$

où $y\left(x\right)$ est la solution exacte de 3.1.

Si la solution numérique doit être une approximation utile de $y(x)$, l'erreur globale doit être maintenue dans des limites raisonnables tout au long de l'intervalle d'intégration [9].

3.2.2 L'erreur locale

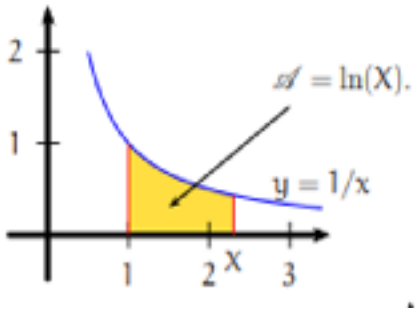
Pour comprendre la signification de l'erreur locale, considérons toutes les courbes de solution dans le plan (x, y) de l'équation différentielle dans 3.1. La condition initiale dans 3.1 sélectionne comme solution la courbe unique $y(x)$ qui contient le point (x_0, y_0). Supposons que la solution numérique soit passée à x_{j-1}.Parce que y_{j-1} contient une erreur, elle ne repose pas sur $y(x)$, mais plutôt sur la courbe de solution voisine $u(x)$ spécifiée par $u\left(x_{j-1}\right) = y_{j-1}$. En passant à x_j, la méthode numérique suivra alors $u(x)$ au lieu de $y(x)$ [9]. L'erreur locale dans l'étape de x_{j-1} à x_j est définie comme étant :

$$\eta_j = u\left(x_j\right) - y_j \tag{3.4}$$

3.3 La fonction logarithme neperien

Définition 3.3.1 *La fonction logarithme est l'unique fonction f, définie et dérivable sur $]0,+\infty[$ et vérifiant $f(1)=0$ et pour tout réel $x>0$, $f'(x)=\frac{1}{x}$.* ln *est la primitive de la fonction $x\longrightarrow\frac{1}{x}$ sur $]0,+\infty[$ qui s'annule en 1 [8]*

$$\textit{Pour tout réel } x>0,\ \ \ln(x)=\int_1^x\frac{1}{t}dt.$$



3.3.1 Propriétés analytiques

– La fonction logarithme népérien est définie et dérivable sur $]0,+\infty[$ et

$$\text{Pour tout réel } x>0,\ \ (\ln)'(x)=\frac{1}{x}.$$

– La fonction logarithme népérien est continue et strictement croissante sur $]0,+\infty[$.

– Limites aux bornes du domaine :

$$\lim_{\substack{x\to0\\x>0}}\ln(x)=-\infty,\quad\lim_{x\to+\infty}\ln(x)=+\infty.$$

– Théorème de croissances comparées :

$$\lim_{x\to+\infty}\frac{\ln(x)}{x}=0.$$

– Nombre dérivé en 1 :

$$\lim_{h\to 0} \frac{\ln(1+h)}{h} = 1.$$

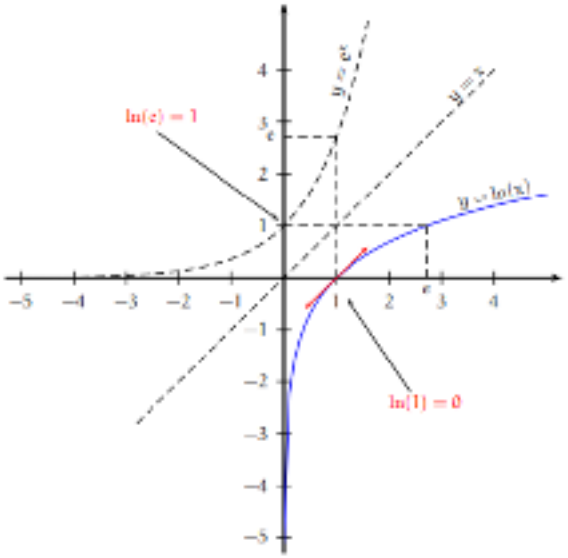


FIG. 3.1 – La fonction logarithme

3.3.2 Propriétés algébriques

Pour tous réels $x > 0$ *et* $y > 0$,

$$\ln(x \times y) = \ln(x) + \ln(y).$$

Pour tout réel $x > 0, \ln\left(\frac{1}{x}\right) = -\ln(x)$.

Pour tous réels $x > 0$ *et* $y > 0$,

$$\ln\left(\frac{x}{y}\right) = \ln(x) - \ln(y).$$

Pour tout réel $x > 0$ *et entier relatif* n,

$$\ln(x^n) = n\ln(x).$$

Pour tous réels $x > 0$ *et* $y > 0$,

$$\ln(x) + \ln(y) = \ln(x \times y).$$

Pour tout réel $x > 0, -\ln(x) = \ln\left(\frac{1}{x}\right)$.

Pour tous réels $x > 0$ *et* $y > 0$,

$$\ln(x) - \ln(y) = \ln\left(\frac{x}{y}\right).$$

Pour tout réel $x > 0$ *et entier relatif* n,

$$n\ln(x) = \ln(x^n).$$

3.3.3 Liens avec la fonction exponentielle

Pour tout réel $x, \ln(e^x) = x$. *Pour tout réel* x *strictement positif* $e^{\ln(x)} = x$.

3.3.4 Résolution d'équations et d'inéquations

– Pour tout réel a, l'équation $\ln(x) = a$ a une solution et une seule.

– Pour tous réels $x > 0$ *et* $y > 0$, $(\ln(x) = \ln(y) \Leftrightarrow x = y)$.

– Pour tout réel $x > 0$ et tout réel a, $\ln(x) = a \Leftrightarrow x = e^a$.

La fonction logarithme népérien est strictement croissante sur $]0, +\infty[$. Donc

– Pour tous réels $x > 0$ *et* $y > 0$ $(\ln(x) < \ln(y) \Leftrightarrow x < y)$.

– Pour tout réel $x > 0$ *et tout réel* $a, \ln(x) < a \Leftrightarrow x < e^a$.

3.4 Application : Le modèle logarithmique

Pour illustrer la méthode traitée et démontrer son éfficacité dans le calcul, le modèle logarithmique est considérè en prenant une taille de pas uniforme h. La principale motivation dans la sélection d'exemple d'application vient de la grande importance de modèle logarithmique dans la modélisation des problèmes de la vie réelle. Les fonctions logarithmes sont fréquemment utilisées pour modéliser la croissance ou dépréciation des investissements financiers, croissance démographique, décroissance radioactive, et les phénomènes où une quantité est autorisés à subir une croissance sans contrainte.

Soit le (PVI) suivant :

$$\begin{cases} \frac{dy}{dx} = \frac{1}{x+1} & 0 \leq x \leq 1 \, . \\ y(0) = 0 \end{cases}$$

Pour $d = 10$, $h = 0.1$, $x_0 = 0$, $y_0 = 0$.

Alors la fonction objective

$$F(y_1, y_2, ..., y_{10}) = \sum_{i=1}^{10} \left[\frac{y_{j.} - y_{j-1}}{h} - f\left(x_{j-1}, y_{j-1}\right) \right]^2$$
$$= \sum_{i=1}^{10} \left[\frac{y_{j.} - y_{j-1}}{h} - \frac{1}{x_{j-1} + 1} \right]^2$$

La solution excate est $y = \ln(x + 1)$.

3.5 Résultats Numériques et discussions

Dans ce travail, les paramètres adoptés par (APF) pour traiter notre problème sont cités dans Table 3.1.

Les résultats obtenus et la comparaison entre la performance du (APF) et la méthode d'*Euler* sont donnés par Table 3.2.

Les paramètres	*La valeur*
Taille de la population (n)	20
Probabilité de commutateur (p)	0.8
Nombre total d'itérations (N)	2000
Dimension des variables de recherche (d)	10

TAB. 3.1 – Paramètres adoptés (APF)

j	x_j	Y_{exact}	Y_{Euler}	Y_{APF}	Err_{Euler}	Err_{APF}
0	0	0	0	0	0	0
1	0.1	0.0953	0.1000	0.1000	0.0047	0.0047
2	0.2	0.1823	0.1909	0.1900	0.0086	0.0077
3	0.3	0.2623	0.2742	0.2733	0.0119	0.0110
4	0.4	0.3364	0.3512	0.3503	0.0148	0.0139
5	0.5	0.4054	0.4226	0.4217	0.0172	0.0163
6	0.6	0.4700	0.4893	0.4884	0.0193	0.0184
7	0.7	0.5306	0.5518	0.5509	0.0212	0.0203
8	0.8	0.5877	0.6106	0.6097	0.0229	0.0220
9	0.9	0.6418	0.6661	0.6652	0.0243	0.0234
10	1	0.6931	0.7188	0.7179	0.0257	0.0248

TAB. 3.2 – Résultats excates et résultats obtenus par (APF) et par Euler avec leurs erreurs absolues pour d=10

L'erreur moyen au carré d'APF est égale à 2.7805×10^{-4} et l'erreur moyen au carré de la méthode d'$Euler$ est de 3.0453×10^{-4}.

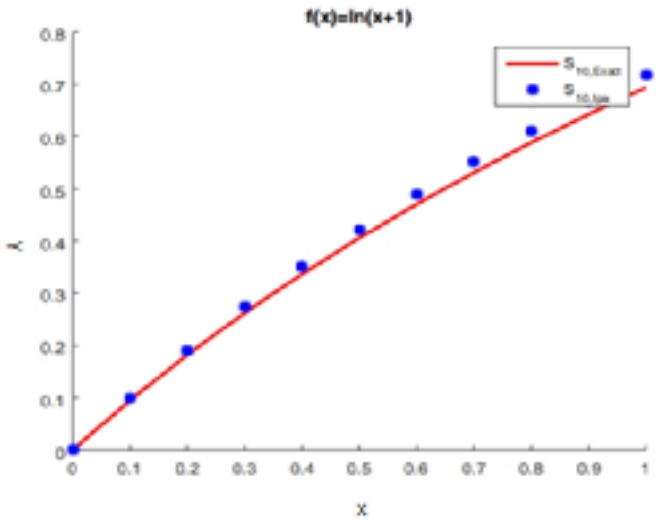


FIG. 3.2 – Graphe des résultats exactes et résultats approchés obtenus par (APF) pour d=10

Conclusion : En comparant la moyenne au carré de chaque méthode, nous constatons que L'APF donne de meilleurs résultats sur l'optimisation.

On remarque que l'erreur moyen au carré de la méthode d'$Euler$ est plus grand que l'erreur moyen au carré de l'APF et on voit aussi que les erreurs absolues d'$Euler$ sont plus grands que les erreurs absolues d'APF.

Conclusion

Les travaux de recherche présentés dans ce travail concernent le développement de nouvelles méthodes d'optimisation globale qui s'appuient sur les algorithmes métaheuristiques. Nous avons focalisé nos recherches sur un des algorithmes évolutionnaires destinés à résoudre des problèmes d'optimisation difficiles qui est l'algorithme de pollinisation des fleurs.

Comme tous les algorithmes métaheuristiques, les valeurs des paramètres d'un algorithme peuvent affecter les performances de cet algorithme. Par conséquent, certaines variantes d'APF ont tenté de régler leurs paramètres. En outre, la norme APF n'a pas été appliquée pour résoudre le problème à grande échelle. Par conséquent, plusieurs chercheurs ont tenté d'utiliser APF avec des modifications pour résoudre un large éventail de problèmes d'optimisation. Cet algorithme a une bonne capacité robuste à résoudre des problèmes d'optimisation continue, et cette capacité a été testée pour des problèmes d'optimisation discrets.

L'algorithme de pollinisation des fleurs est un algorithme d'optimisation efficace avec un large éventail d'applications. Dans cette étude, nous avons appliqué l'APF pour résoudre approximativement un problème de valeur initial, via un exemple choisi et après une comparaison entre les solutions exactes, les résultats de l'algorithme et les résultats de la méthode d'$Euler$; APF a été trouvé de façon exponentielle mieux en offrant des solutions précises avec la plus petite erreur de quantité.

En outre, de nombreux problèmes d'optimisation dans les applications du monde réel sont multiobjectifs. Ainsi, certaines variantes ont porté sur l'extension d'APF pour résoudre des problèmes multi-objectifs difficiles. Ce domaine nécessite d'autres recherches, car l'optimisation multi-objectifs peut être très étendue sur le plan informatique dans des dimensions plus élevées. Les méthodes

efficaces doivent être triées pour générer des fronts de Pareto de haute qualité pour l'optimisation multi-objective.

En fin de compte, on peut s'attendre à l'application de l'APF multi-objectif pour d'autre problèmes plus compliqués.

Bibliographie

[1] Alyasseri, Z. A. A., Khader, A. T., Al-Betar, M. A., Awadallah, M. A., & Yang, X. S. (2018). Variants of the flower pollination algorithm : a review. In Nature-Inspired Algorithms and Applied Optimization (pp. 91-118). Springer, Cham.

[2] Benaichouche, A. N. (2014). Conception de métaheuristiques d'optimisation pour la segmentation d'images : application aux images IRM du cerveau et aux images de tomographie par émission de positons (Doctoral dissertation, Université Paris-Est).

[3] Berhail, A. (2016). Optimisation sans contraintes.(Doctoral dissertation, Université 08 Mai 1945 Guelma).

[4] Boussaid, I. (2013). Perfectionnement de métaheuristiques pour l'optimisation continue (Doctoral dissertation, Université Paris Est).

[5] Dréo, J., Pétrowski, A., Siarry, P., & Taillard, E. (2003). Métaheuristiques pour l'optimisation difficile.EYROLLES, pp.356, Algorithmes, 978-2-212-11368-6. (hal-00843020).

[6] Ghoumari, A. (2018). Métaheuristiques adaptatives d'optimisation continue basées sur des méthodes d'apprentissage (Doctoral dissertation, Université Paris-Est).

[7] Hachimi, H. (2013). Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications (Doctoral dissertation, Rouen, INSA).

[8] Jean, L. R. (2012). La fonction logarithme népérien (Cours, Univertsité Paris) https ://www.maths-france.fr/Terminale/TerminaleS/FichesCours/Logarithme.pdf.

[9] Malanchuk, J., Otis, J., & Bouver, H. (1980). Efficient Algorithms for Solving Systems of Ordinary Differential Equations for Ecosystems Modeling. Environmental Research Laboratory, Office of Research and Development, US Environmental Protection Agency.

[10] Ouaar, F., & Khelil, N. (2018). Solving initial value problems by flower pollination algorithm. American Journal of Electrical and Computer Engineering, 31-36.

[11] Radi, B., & El Hami, A. (2018). Méthodes numériques avancées sous Matlab : Résolution des équations non linéaires, différentielles et aux dérivées partielles (Vol. 7). ISTE Group.

[12] Talukder, S. (2011). Mathematicle modelling and applications of particle swarm optimization.(Doctoral dissertation, Blekinge Institute of Technology)

[13] Yang, X. S. (2011). Metaheuristic optimization. Scholarpedia, 11472.

[14] Yang, X. S. (2012). Flower pollination algorithm for global optimization. In international conference on unconventional computing and natural computation (pp. 240-249). Springer, Berlin, Heidelberg.

[15] Yang, X. S., Cui, Z., Xiao, R., Gandomi, A. H., & Karamanoglu, M. (2013). Swarm intelligence and bio-inspired computation : theory and applications. Newnes.

Annexe A : Logiciel Matlab

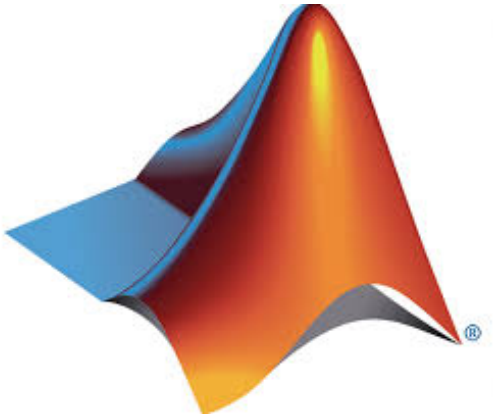


FIG. 3.3 – Logo du logiciel Matlab

– Matlab est une abréviation de Matrix Laboratory. Écrit à l'origine, en Fortran, par $C.Moler$, Matlab était destiné à faciliter l'accès au logiciel matriciel développé dans les projets LINPACK et EISPACK. La version actuelle, écrite en C par the MathWorks Inc, existe en version professionnelle et en version étudiant. Sa disponibilité est assurée sur plusieurs plateformes.

– Matlab est un environnement puissant, complet et facile à utiliser destiné au calul scientifique. Il apporte aux ingénieurs, chercheurs et à tout scientifique un système interactif intégrant calcul numérique et visualisation.

– Matlab possède son propre langage, intuitif et naturel qui permet des gains de temps de CPU spectaculaires par rapport à des langages comme le C, le TurboPascal et le Fortran. Avec Matlab, on peut faire des liaisons de façon dynamique, à des programmes C ou Fortran, écganger des données avec d'autres applications ou utiliser Matlab comme moteur d'analyse et de visualisation.

– Matlab comprend aussi un ensemble d'outils spécifiques à des domaines, appelés *toolboxes* (ou

boîtes à outils). Indispensables à la plupart des utilisateurs, les boîtes à outils sont des collections de fonctions qui étendent l'environnement Matlab pour résoudre des catégories spécifiques de problème. Les domaines couverts sont trés variés et comprennent notamment le traitement de signal. L'automatique, les réseaux neurones, le calcule de structure, les statistiques....etc.

– Matlab permet le travail interactif soit en mode commande, soit en mode programmation ; tout en ayant toujours la possibilité de faire des visualisations graphiques. Considéré comme un des meilleurs langages de programmation (C ou Fortran), Matlab possède les particularités suivantes par rapport à ces langages :

- la programmation facile ;
- la continuité parmi les valeurs entières, réelles et complexes ;
- la gamme étendue des nombres et et leurs précisions ;
- la bibiliothèque mathématique trés compréhensive ;
- l'outil graphique qui inclus les fonctions d'interface graphique et les utilitaires ;
- la possibilité liaison avec les autres langages classiques de programmation (C ou Fortran).

La fonction	Le role
Plot	permet de tracer une ou plusieure courbe sur le méme graphe
Legend	sont un moyen utile pour étiqueter les série de donnée tracées sur un graphique
hold on	permet de tracer plusieure courbe sur la figure courante
grid on	met la grille sur le graphe tracé par plot

TAB. 3.3 – Les fonctions les plus utilisées dans ce travail

Annexe B : Code Matlab des algorithmes

3.6 Code Matlab de méthode d'Euler

function E=euler(f,a,b,ya,M)

$f = @(x)exp(x)\,;$

$a = 0\,;$

$b = 1\,;$

ya= $0\,;$

$M = 10\,;$

$h = (b-a)/M\,;$

$Y = zeros(1, M+1)\,;$

$T = a : h : b\,;$

$Y(1) =$ya$\,;$

$for\ j = 1 : M$

$Y(j+1) = Y(j) + h * f(T(j))\,;$

end

$E = [T'\ Y']\,;$

3.7 Code Matlab d'algorithme de pollinisation des fleurs (APF)

```
% ------------------------------------------------------------ %
% Flower pollenation algorithm (FPA),or flower algorithm       %
% Programmed by Xin-She Yang @ May 2012                        %
% ------------------------------------------------------------ %

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %
% Notes :This demo program contains the very basic components of %
% the flower pollination algorithm (FPA), or flower algorithm (FA), %
% for single objective optimization.        It usually works well for %
% unconstrained functions only. For functions/problems with    %
% limits/bounds and constraints, constraint-handling techniques %
% should be implemented to deal with constrained problems properly. %
%                                                              %
% Citation details :                                           %
% 1)Xin-She Yang, Flower pollination algorithm for global optimization, %
% Unconventional Computation and Natural Computation,          %
% Lecture Notes in Computer Science, Vol. 7445, pp. 240-249 (2012). %
% 2)X. S. Yang, M. Karamanoglu, X. S. He, Multi-objective flower %
% algorithm for optimization, Procedia in Computer Science,    %
% vol. 18, pp. 861-868 (2013).                                 %
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %

function [best,fmin,N_iter]=fpa_demo(para)
% Default parameters
if nargin< 1,
```

```
    para= [200.8];
  end

  n=para(1);              % Population size, typically 10 to 25
  p=para(2);              % probabibility switch

% Iteration parameters
N_iter=2000;              % Total number of iterations

% Dimension of the search variables
d= 3;
Lb=-2 * ones(1, d);
Ub=2 * ones(1, d);

% Initialize the population/solutions
for i=1 :n,
    Sol(i, :)=Lb+(Ub-Lb).*rand(1,d);
    Fitness(i)=Fun(Sol(i, :));
end

% Find the current best
[fmin,I]=min(Fitness);
best=Sol(I, :);
S=Sol;

% Start the iterations – Flower Algorithm
for t=1 :N_iter,
        % Loop over all bats/solutions
        for i=1 :n,
```

```
% Pollens are carried by insects and thus can move in
% large scale, large distance.
% This L should replace by Levy flights
% Formula : x_i^{t+1} = x_i^t + L(x_i^t - gbest)
if rand>p,
%% L=rand;
L=Levy(d);
dS=L.*(Sol(i,:)-best);
S(i,:)=Sol(i,:)+dS;

% Check if the simple limits/bounds are OK
S(i,:)=simplebounds(S(i,:),Lb,Ub);

% If not, then local pollenation of neighbor flowers
else
    epsilon=rand;
    % Find random flowers in the neighbourhood
    JK=randperm(n);
    % As they are random, the first two entries also random
    % If the flower are the same or similar species, then
    % they can be pollenated, otherwise, no action.
    % Formula : x_i^{t+1}+epsilon*(x_j^t-x_k^t)
    S(i,:)=S(i,:)+epsilon*(Sol(JK(1),:)-Sol(JK(2),:));
    % Check if the simple limits/bounds are OK
    S(i,:)=simplebounds(S(i,:),Lb,Ub);
end
```

```
        % Evaluate new solutions
          Fnew=Fun(S(i, :));
        % If fitness improves (better solutions found), update then
          if (Fnew<=Fitness(i)),
              Sol(i, :)=S(i, :);
               Fitness(i)=Fnew;
        end

        % Update the current global best
        if Fnew<=fmin,
          best=S(i, :);
          fmin=Fnew;
        end
      end
        % Display results every 100 iterations
        if round(t/100)==t/100,
        best
        fmin
        end

end

% Output/display
disp(['Total number of evaluations : ',num2str(N_iter*n)]);
disp(['Best solution=',num2str(best),'      fmin=',num2str(fmin)]);

% Application of simple constraints
function s=simplebounds(s,Lb,Ub)
```

```
  % Apply the lower bound
  ns_tmp=s;
  I=ns_tmp<Lb;
  ns_tmp(I)=Lb(I);

  % Apply the upper bounds
 J=ns_tmp>Ub;
  ns_tmp(J)=Ub(J);
  % Update this new move
  s=ns_tmp;

% Draw n Levy flight sample
function L=Levy(d)
% Levy exponent and coefficient
% For details, see Chapter 11 of the following book :
% Xin-She Yang, Nature-Inspired Optimization Algorithms, Elsevier, (2014).
beta=3/2;
sigma=(gamma(1+beta)*sin(pi*beta/2)/(gamma((1+beta)/2)*beta*2^((beta-1)/2)))^(1/beta);
      u=randn(1,d)*sigma;
      v=randn(1,d);
      step=u./abs(v).^(1/beta);
L=0.01*step;

% Objective function and here we used Rosenbrock's 3D function
function z=Fun(u)
z = (1 - u(1))^2 + 100 * (u(2) - u(1)^2)^2 + 100 * (u(3) - u(2)^2)^2;
```

Annexe C : Abréviations et Notations

Les différentes abréviations et notations utilisées tout au long de ce mémoire sont expliquées ci-dessous :

Abréviations	Explication
APF	Algorithme de pollinisation des fleurs
AEs	Algorithmes évolutionnaires
AGs	Algorithmes génétiques
OEP	Optimisation par essaim particulaire
FDP	Fonction de densité de probabilité
FDC	Fonction de distribution cumulative
PVI	Problème de valeur initiale
x^*	Un minimiseur global
x_L^*	Un minimiseur local
$f(x^*)$	Un minimum global
$f(x_L^*)$	Un minimum local
$x^*(t)$	Optimum trouvé au pas de temps

Résumé :

La pollinisation des fleurs est un processus intéressant dans le monde naturel. Ses fonctionnalités évolutives peuvent être utilisées pour concevoir de nouveaux algorithmes améliorés. Dans ce travail, nous proposons un nouvel algorithme, l'algorithme de pollinisation des fleurs, inspiré par le processus de pollinisation des fleurs, et développé par Yang en 2012. Nous avons d'abord utilisé quatre fonctions de test pour vérifier le nouvel algorithme et comparer ses performances avec les algorithmes génétiques et les particules d'essaim. Nos résultats de simulation montrent que l'algorithme de fleur est plus efficace qu'AG et OPS. Nous avons ensuite étudié comment optimiser un problème de valeur initiale dans une équation différentielle ordinaire linéaire du premier ordre on se basant sur l'algorithme de pollinisation des fleurs (APF) et nous avons comparé les résultats obtenus.

Mots clés : Algorithme de pollinisation des fleurs (APF), Optimisation, Méta- heuristique, problème de valeur initiale (PVI).

Abstract :

Flower pollination is an intriguing process in the natural world. Its evolutionary characteristics can be used to design new optimization algorithms. In this work, we propose a new algorithm, the flower pollination algorithm, inspired by the flower pollination process, developed by Yang in 2012. We first used four test functions to check the new algorithm and compare its performance with genetic algorithms and swarm particles. Our simulation results show that the flower algorithm is more efficient than GA and PSO. Then we studied how to optimize an ordinary linear differential equation of the first order in the IVP based on the flower pollination algorithm (FPA) and we are compared the results.

Keywords : Flower pollination algorithm (FPA), Optimization, Metaheuristic, Initial-value problem (IVP).

الملخص:

تلقيح الزهور عملية مثيرة للاهتمام في العالم الطبيعي. يمكن استخدام ميزاته القابلة للتطوير لتصميم خوارزميات جديدة و محسنة. في هذا العمل, نقترح خوارزمية جديدة, خوارزمية تلقيح الزهور, مستوحاة من عملية تلقيح الزهور, والتي قام يونغ بتطويرها عام 2012. استخدمنا أولا أربع دوال اختبار للتحقق من الخوارزمية الجديدة ومقارنة أدائها مع أداء خوارزمية الجسيمات الجينية و خوارزمية جسيمات السرب. تظهر نتائج المحاكاة لدينا أن خوارزمية الزهرة هي الأكثر كفاءة من السابقتين. ثم درسنا كيفية تحسين مشكلة القيمة الأولية في معادلة تفاضلية خطية عادية من الدرجة الأولى باستعمال خوارزمية تلقيح الزهور. ثم قمنا بمقارنة النتائج المحصلة.

الكلمات الرئيسية: خوارزمية تلقيح الزهور,الأمثل, مشكلة القيمة الأولية.