Optimisation Naturelle : Une Exploration Approfondie de l'Algorithme Firefly

Anthony Wilhelm
UHA 4.0
Université de Haute Alsace
Mulhouse , France
anthony.wilhelm@uha.fr

***Abstract*—Afin d'accélérer l'exécution des divers algorithmes, les chercheurs en informatique ont trouvé inspiration dans les mécanismes naturels. Dans ce document, nous explorons une approche novatrice visant à optimiser le temps de calcul de fonctions qui, dans des conditions normales, exigeraient un temps d'exécution infini. Nous nous penchons sur la manière dont cette méthode, inspirée par les phénomènes naturels, peut considérablement améliorer l'efficacité des algorithmes, ouvrant ainsi de nouvelles perspectives dans le domaine de l'informatique et de l'optimisation.**

***Index Terms*—optimisatoion, luciole, algorithme,**

I. Introduction

Dans le domaine informatique, l'optimisation est une discipline cruciale qui permet la réalisation efficace de calculs liés aux fonctions scientifiques. Notre attention se porte ici sur une approche particulière d'optimisation, à savoir l'algorithme Firefly, qui s'inspire du comportement social des lucioles. Ces insectes communiquent entre eux en utilisant des signaux lumineux pour attirer un partenaire, et cette inspiration naturelle offre une perspective unique dans le développement d'algorithmes.

Dans la suite de cet article, nous explorerons le fonctionnement détaillé de l'algorithme Firefly, examinant comment il émule les interactions lumineuses des lucioles pour optimiser des fonctions. Ensuite, nous partagerons les résultats obtenus lors de l'exécution de cet algorithme avec différentes fonctions de test telles que la sphère, la fonction de Restrigin, la fonction de Rosenbrock et la fonction d'Ackley. Cette analyse pratique nous permettra de mieux comprendre l'efficacité et la robustesse de l'algorithme Firefly dans la résolution de problèmes d'optimisation complexes liés aux fonctions scientifiques.

II. Contextualisation de l'algorithme firefly

A. l'inspiration biologique des lucioles

L'algorithme Firefly se distingue en tant qu'algorithme d'optimisation novateur intégrée par Xin-She Yang en 2008 qui tire son inspiration des phénomènes naturels, précisément du comportement social des lucioles. Dans la nature, ces insectes nocturnes utilisent des signaux lumineux pour communiquer entre eux, établir des liens sociaux et, notamment, attirer des partenaires. Cette forme de communication lumineuse a captivé les chercheurs en informatique, qui ont transposé ces principes dans le domaine de l'optimisation.

L'idée fondamentale derrière l'algorithme Firefly est de reproduire la manière dont les lucioles s'organisent dans l'espace en fonction de l'intensité de leurs lueurs. Chaque luciole, représentant une solution potentielle dans un espace d'optimisation, ajuste sa position en réponse à la luminosité des lucioles voisines. Cette interactivité est utilisée pour guider la convergence du groupe de lucioles vers une solution optimale.

B. Algorithme "Firefly" en Action

En intégrant ces phénomènes naturels dans le processus d'optimisation, l'algorithme Firefly propose une approche bio-inspirée, où la recherche de solutions se déroule de manière collective et adaptative. Cette conceptualisation unique offre un pont entre les stratégies d'optimisation informatique et les mécanismes observés dans la nature, ouvrant ainsi de nouvelles perspectives pour la résolution de problèmes complexes. En explorant les comportements sociaux des lucioles, l'algorithme Firefly devient un exemple concret de la manière dont les principes naturels peuvent être exploitées pour améliorer les performances des algorithmes d'optimisation.

III. Manière dont les lucioles communiquent entre elles pour converger

Les lucioles, agissant comme agents dans l'algorithme Firefly, communiquent entre elles par le biais d'interactions lumineuses pour converger vers une solution optimale. Ce mécanisme de communication simulé s'inspire du comportement naturel des lucioles, qui utilisent leurs signaux lumineux pour coordonner leurs actions dans un environnement nocturne. Chaque luciole émet une lumière dont l'intensité est proportionnelle à la qualité de la solution qu'elle représente. Ainsi, une luciole avec une luminosité plus élevée indique une solution de meilleure qualité.

Lors de chaque itération de l'algorithme, chaque luciole évalue l'attraction qu'elle ressent envers les autres lucioles en fonction de leur luminosité relative. Une luciole sera davantage attirée par celles émettant une lumière plus intense. Cette attraction est calculée en utilisant une fonction d'attraction, déterminée par le facteur d'absorption lumineuse et la distance entre les lucioles.

Le processus de communication entre les lucioles se traduit par des ajustements dynamiques de leurs positions dans

l'espace de recherche. Les lucioles se déplacent vers des positions qui maximisent leur attraction mutuelle, favorisant ainsi la convergence du groupe vers des configurations spatiales où la qualité globale des solutions est optimale.

La communication lumineuse entre les lucioles crée une dynamique collective, simulant la capacité du groupe à s'auto-organiser vers des solutions de qualité supérieure. Ce mécanisme favorise une exploration efficace de l'espace des solutions, les lucioles convergeant progressivement vers des régions où la concentration de solutions optimales est maximale. En résumé, la communication lumineuse des lucioles dans l'algorithme Firefly crée une synergie bio-inspirée qui guide le groupe vers une convergence collective, conduisant à la découverte de solutions optimales dans des problèmes d'optimisation complexes.

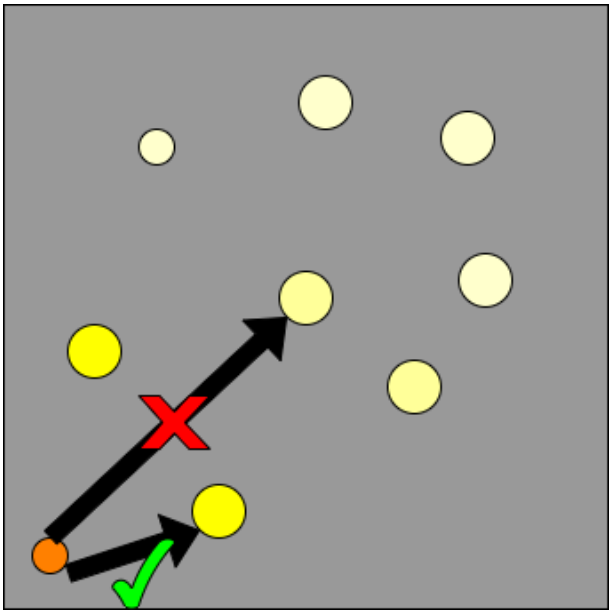

Fig. 1. Attirance des lucioles

Cette illustration s'inspire d'une images des recherches menées par Andrey Dik (https://www.mql5.com/fr/articles/11873) sur l'algorithme des lucioles, également connu sous le nom de Firefly Algorithm (FA), telles que présentées dans son article intitulé "Algorithmes d'optimisation de la population : Algorithme des Lucioles". Dans ce travail, l'image met en lumière le concept fondamental d'attraction entre les lucioles, soulignant leur propension à être attirées les unes par les autres en fonction de leur proximité et de leur luminosité. L'analogie avec le comportement naturel des lucioles est utilisée pour illustrer la manière dont ces créatures virtuelles ajustent leur position dans l'espace de recherche, s'orientant vers des solutions potentiellement optimales à mesure qu'elles interagissent les unes avec les autres. Cette visualisation offre une représentation graphique captivante des principes bio-inspirés sous-jacents à l'algorithme des lucioles.

IV. Explication détaillée du fonctionnement de l'algorithme

Nous venons de voir que l'algorithme Firefly, inspiré du comportement social des lucioles. Explorons en détail le fonctionnement de cet algorithme, mettant en avant ses principaux concepts et mécanismes.

A. Initialisation

L'algorithme débute par la création d'une population de lucioles, chacune représentant une solution possible à un problème d'optimisation. Les positions initiales des lucioles sont générées de manière aléatoire dans l'espace de recherche, et leur qualité est évaluée en fonction de la fonction objectif.

B. Attraction et Mouvement

Le concept central de l'algorithme est l'attraction entre les lucioles. À chaque itération, chaque luciole évalue l'attraction qu'elle ressent envers les autres lucioles, en se basant sur la luminosité relative. Les lucioles sont attirées vers celles qui émettent une lumière plus intense. La mise à jour de la position d'une luciole est effectuée en fonction de cette attraction, ce qui favorise le mouvement vers des positions plus lumineuses dans l'espace de recherche.

C. Facteur d'absorption lumineuse

Le facteur d'absorption lumineuse β joue un rôle crucial dans la modulation de l'attraction entre les lucioles. Il est initialisé à $\beta 0$ et subit une décroissance au fil des itérations, ce qui représente une adaptation dynamique de la visibilité des lucioles. Ce mécanisme permet de réguler l'intensité des signaux lumineux et d'ajuster l'exploration de l'espace des solutions.

D. Exploration et Convergence

Les lucioles explorent l'espace de recherche de manière collaborative, se déplaçant vers des positions plus lumineuses. Ce processus simule la recherche collective de solutions optimales. Au fil des itérations, l'algorithme converge vers des configurations spatiales où les lucioles représentent des solutions de haute qualité.

E. Mise à jour et évaluation

Après chaque déplacement, la qualité des solutions est réévaluée, ajustant ainsi la luminosité des lucioles. La meilleure solution identifiée est conservée, et le processus itératif continue jusqu'à atteindre un critère d'arrêt prédéfini.

En résumé, l'algorithme Firefly combine l'idée d'une attraction sociale entre les lucioles, basée sur leur luminosité respective, avec des mécanismes d'adaptation dynamique (facteur d'absorption lumineuse) pour guider la recherche vers des solutions optimales. Ce processus bio-inspiré offre une approche puissante pour résoudre divers problèmes d'optimisation en exploitant des principes de la nature.

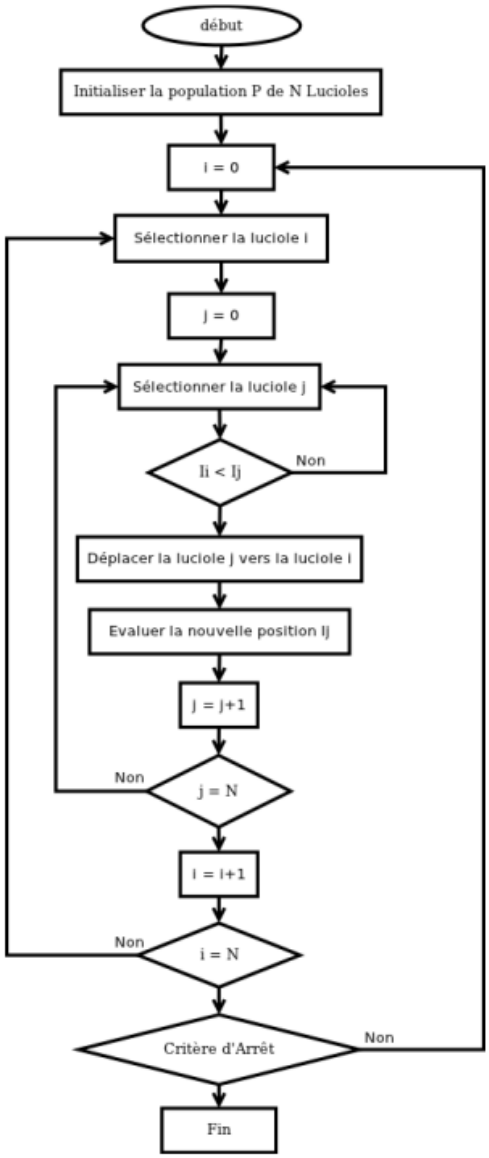

Fig. 2. Diagramme d'activité de l'algorithme des lucioles
[1]

V. Modélisation Mathématique

A. Attraction entre deux lucioles

L'attraction entre deux lucioles i et j est exprimée mathématiquement par une formule d'intensité lumineuse qui est la suivante:

$$\beta i, j = \beta_0 e^{-\gamma \mathrm{r}^2 i,j}(Zi - Zj)$$

[2]

β : C'est l'attraction entre les lucioles i et j . C'est la quantité par laquelle la position de la luciole i est ajustée enréponse à la luciole j.
β_0 : Il s'agit du facteur d'absorption lumineuse initial.Il représente la luminosité maximale initile d'une luciole.
γ : C'est le coefficient de la décroissance de l'absorption lumineuse. Il détermine la vitesse à laquelle l'attraction décroit avec la distance.
r : C'est la distance euclidienne entre les positions des lucioles i et j dans l'espace de recherche, Cette distance est au carré dans l'exponentiel , ce qui signifie que les lucioles sont plus fortement attiré lorsque elles sont les plus proches.
Zi et Zj : Sont les distances entre les lucioles i et j

B. Génération des populations initial

Pour débuter l'algorithme il faut créer une population de base de taille désirée avec des placement aléatoire on utilisera une attirance pour β_0 pour que toute les luciole est une luminosité de départ égale :

$$\mathrm{x}t + 1 = xt + \beta_0 e^{-\gamma \mathrm{r}^2} + a\epsilon$$

[2]

xt+1 : C'est la composante de la position de la luciole t
aϵ : Est un terme de randomisation avec ces paramètre de randomisation.

C. Mouvement des lucioles les moins lumineuses vers les plus lumineuses

Les lucioles sont attiré par les autres qui sont plus lumineuses ce qui est déterminé par :

$$\mathrm{x}i = xi + \beta_0 e^{-\gamma \mathrm{r}^2 i,j}(xj - xi) + a\epsilon$$

[2]

xi : position d'une luciole
xj : position d'une seconde luciole

En réorganisant le classement, la meilleure solution peut être identifiée parmi l'ensemble des lucioles, permettant ainsi de déterminer la solution optimale du problème d'optimisation considéré.
Le paramètre γ joue un rôle crucial dans la modélisation de la variation attractivité entre les lucioles. Sa valeur revêt une importance particulière dans la détermination de la vitesse à laquelle l'algorithme converge ainsi que dans le comportement global de l'algorithme. En effet, γ influence directement l'intensité de l'attraction entre les lucioles, impactant ainsi la dynamique du processus d'optimisation. La sélection judicieuse de la valeur de γ est donc essentielle pour ajuster les performances de l'algorithme. [3]
Ces formulations mathématiques encapsulent les règles fondamentales de l'algorithme des lucioles, permettant une représentation précise de la manière dont les lucioles interagissent et comment leurs positions évoluent au fil du temps. Ces équations définissent les mécanismes essentiels qui guident la recherche vers des solutions optimales dans l'espace de recherche défini par le problème d'optimisation.

VI. Méthodologie Expériementale

A. Installtions

Pour manipuler cet algorithme, l'utilisation de la bibliothèque Python "mealpy" est demandée. Cette bibliothèque propose une implémentation intégrée et encapsuler de l'algorithme FireFly (FFA dans la librairie).

Des évaluations de performance on été faite à partir de deux langage :

- Python : C'est un langage qui dispose d'un ecosytème de de bibliothèque plutot puissant telles que Numpy, SciPy et Matplotlib (affichage de graphe) simplifiant les opérations numérique les calculs scientifique et les visualisations des résultats.
- C++ : C++ est réputé pour ses performances élevées. Si votre algorithme des lucioles doit être exécuté de manière intensive ou sur des ensembles de données volumineux, la performance accrue de C++ peut être un avantage.C'est aussi un langage très utilisée dans le domaine de la recherche en informatique.

La bibliothèque "mealpy" étant initialement conçue en langage Python, sa réutilisation pour les évaluations s'est avérée impossible. Avant de pouvoir procéder aux évaluations de performance, une conversion de la bibliothèque "mealpy" vers le langage C++ a été nécessaire (j'aurais malheureusement un biais dans mes expérimentation car je n'ai pas réussi à traduire l'algorithme en C++ en respectant la structure de la bibliothèque dans le temp imparti). Les mesures de performances ont étés effectué sur des fonctions souvent utilisée pour ça dans le milieu de l'optimisation.

B. Les fonctions tester

- La fonctions sphère : La fonction sphère vise à évaluer la capacité d'un algorithme d'optimisation à trouver le minimum global. La fonction sphère a une forme géométrique qui ressemble à une sphère dans un espace multidimensionnel. Chaque composante contribue à la distance au carré par rapport à l'origine, et la somme totale de ces distances donne la valeur de la fonction. Représentation mathématique :

$$f(x) = \sum_{i=1}^{D} x_i^2$$

- La fonction Ackley : La fonction d'Ackley est connue pour sa complexité et son comportement multimodal, ce qui signifie qu'elle possède plusieurs optima locaux. Elle est souvent utilisée pour tester la capacité d'un algorithme d'optimisation à échapper aux minima locaux et à converger vers le minimum global. Représentation mathématique :

$$f(x) = -a \cdot \exp\left(-b\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(c \cdot x_i)\right) + a + \exp(1)$$

- La fonction Rastrigin : La fonction de Rastrigin est caractérisée par sa nature multimodale, ce qui signifie qu'elle présente plusieurs optima locaux, créant des difficultés pour les algorithmes d'optimisation qui peuvent être piégés dans des minimums locaux. Représentation mathématique :

$$f(x) = A \cdot D + \sum_{i=1}^{D} \left(x_i^2 - A \cdot \cos(2\pi x_i)\right)$$

- La fonction Rosenbrock : La fonction de Rosenbrock a une forme en "vallée" allongée, ce qui la rend intéressante pour tester les algorithmes d'optimisation. Il s'agit de minimiser la fonction dans une étroite vallée avec une solution optimale à l'intérieur.RosenBrock est utilisée pour évaluer la capacité des algorithmes d'optimisation à naviguer à travers des vallées allongées et converger vers le minimum global. Elle teste la robustesse des algorithmes face à des problèmes de convergence lente et à des régions difficiles de l'espace de recherche. Représentation mathématique :

$$f(x) = \sum_{i=1}^{D-1} \left(100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right)$$

C. Protocole expérimentale

Pour chacun des langage et afin d'obtenir des résultats significatifs tout en évitant des temps d'exécutions excessivement longs, les fonctions ont été testées avec 5000 époques pour 10 itérations. Trois ensembles de tests ont été réalisés, chacun avec une population de 30, 50 et 70 individus, et pour chaque population, les tests ont été menés avec 10, 30 et 50 dimensions toutes les fonctions ont été testée avec des bornes de [-10,10] .

Les résultats ont été enregistrés dans un fichier Excel pour faciliter le calcul des moyennes, des écarts-types, et l'identification de tendances significatives.

VII. Les résultats

Les différents graphiques ci-dessous montrent les résultats obtenus pour les neuf cas possibles qui ont été étudiés. L'ordre des dimensions est croissant, donc les trois premiers résultats concernent une taille de population de 30, les trois suivants pour 50, et les trois derniers pour une population de 70. Les dimensions sont toujours répertoriées dans le même ordre : 10, 30, 50.

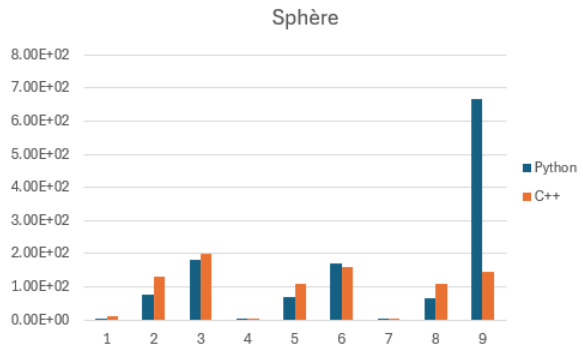

Fig. 3. **Moyenne des 1O meilleurs résultat obtenue pour chacun des cas étudier avec la fonction Sphère**

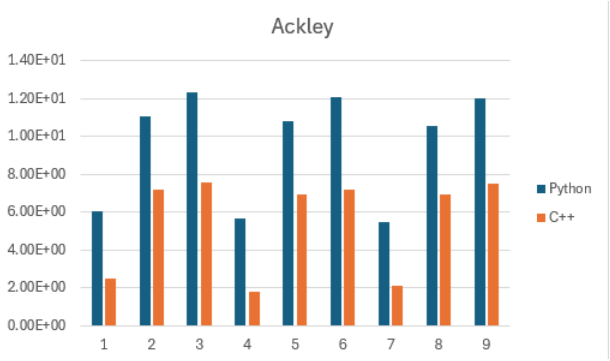

Fig. 4. **Moyenne des 1O meilleurs résultat obtenue pour chacun des cas étudier avec la fonction Ackley**

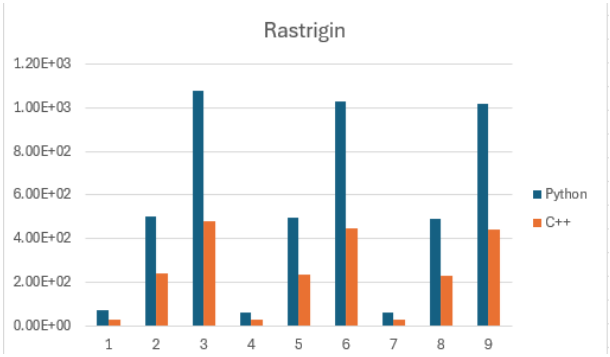

Fig. 5. **Moyenne des 1O meilleurs résultat obtenue pour chacun des cas étudier avec la fonction Rastrigin**

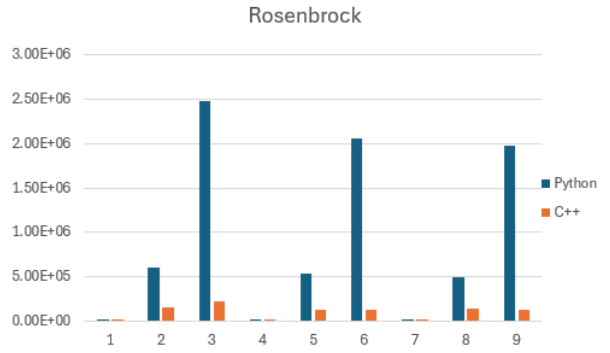

Fig. 6. **Moyenne des 1O meilleurs résultat obtenue pour chacun des cas étudier avec la fonction Rosenbrock**

A. Observation des résultats

On observe que les différentes fonctions renvoient des valeurs qui varient considérablement en termes d'ordres de grandeur. Les valeurs les plus élevées sont atteintes par la fonction Rosenbrock, tandis que les valeurs les plus basses sont données par la fonction Ackley. Il est notable que, pour les fonctions Rosenbrock, Rastrigin et Ackley, une différence significative de performance de l'algorithme des lucioles est observée. On constate une disparité importante, jusqu'à un ordre de grandeur de 10^3, entre les valeurs renvoyées par l'algorithme dans les langages Python et C++.Pour la fonction sphère, les résultats sont généralement similaires, à l'exception d'un cas particulier étudié avec une population de 70 individus et 50 dimensions, où l'on observe une nette différence avec les résultats en Python qui s'envolent.

B. Critique des résultats

La différence notable entre les deux langages, outre leur rapidité d'exécution, peut résider dans la graine utilisée pour générer les nombres aléatoires, qui n'a pas été fixée. De plus, il est important de noter que les deux programmes n'ont pas été lancés simultanément.

Il est observé que l'algorithme peut converger vers une valeur proche de zéro avec des fonctions objectifs, mais présente souvent des pics importants. Cette tendance pourrait être attribuée au nombre limité d'époques utilisé lors des expérimentations, fixé à seulement 5000. Un tel nombre d'époques peut être insuffisant pour permettre à l'algorithme d'explorer de manière approfondie l'espace de recherche et d'atteindre une convergence plus stable.

Pour obtenir une évaluation plus robuste de la convergence de l'algorithme, il pourrais être intéressant d'augmenter le nombre d'époques à un niveau plus élevé.. Une expérimentation prolongée pourrait permettre une exploration plus exhaustive de l'espace de recherche, potentiellement conduisant à une meilleure identification de la solution optimale.

Cette extension du nombre d'époques pourrait aider à atténuer les pics observés, offrant ainsi une vue plus claire de la performance de l'algorithme sur la fonction objectif. Il est conseillé de réaliser des essais avec différents niveaux d'époques pour évaluer la sensibilité de l'algorithme à ce paramètre et pour déterminer le point optimal entre le temps de calcul disponible et la qualité de la solution recherchée.

Table 1 : Benchmark :Moyennes sur 10 itérations des meilleurs des des résultats de 30,50,70 de population et 10,30,50 de dimension avec le langage C++ ainsi que les 4 fonctions tester et

	C++								
population	30			50			70		
dimension	10	30	50	10	30	50	10	30	50
Sphère	1.14E+01	1.29E+02	1.98E+02	3.60E+00	1.10E+02	1.59E+02	2.34E+00	1.09E+02	1.46E+02
Rastrigin	2.98E+01	2.41E+02	4.80E+02	3.05E+01	2.35E+02	4.44E+02	3.01E+01	2.29E+02	4.43E+02
Rosenbrock	1.51E+03	1.60E+05	2.23E+05	2.51E+03	1.31E+05	1.31E+05	3.90E+02	1.40E+05	1.22E+05
Ackley	2.51E+00	7.20E+00	7.58E+00	1.77E+00	6.92E+00	7.16E+00	2.13E+00	6.95E+00	7.53E+00

Table 2 : Benchmark : Moyennes sur 10 itérations des meilleurs des des résultats de 30,50,70 de population et 10,30,50 de dimension avec le langage Python ainsi que les 4 fonctions tester et

	Python								
population	30			50			70		
dimension	10	30	50	10	30	50	10	30	50
Sphère	3.80E+00	7.63E+01	1.83E+02	3.08E+00	6.76E+01	1.70E+02	2.90E+00	6.48E+01	6.69E+02
Rastrigin	7.03E+01	5.02E+02	1.08E+03	6.35E+01	4.95E+02	1.03E+03	6.20E+01	4.90E+02	1.02E+03
Rosenbrock	4.93E+03	6.10E+05	2.48E+06	2.60E+03	5.34E+05	2.06E+06	2.57E+03	4.99E+05	1.97E+06
Ackley	6.04E+00	1.11E+01	1.23E+01	5.64E+00	1.08E+01	1.21E+01	5.49E+00	1.05E+01	1.20E+01

Table 2 : Benchmark : Moyennes sur 10 itérations des Écart-types de 30,50,70 de population et 10,30,50 de dimension avec le langage Python ainsi que les 4 fonctions tester et

	Python								
population	30			50			70		
dimension	10	30	50	10	30	50	10	30	50
Sphère	6.28E-01	6.83E+00	5.87E+00	9.65E-01	6.57E+00	4.58E+00	7.55E-01	4.53E+00	2.98E+01
Rastrigin	6.76E+00	1.46E+05	3.39E+01	9.84E+00	1.37E+01	4.75E+01	7.73E+00	2.24E+01	4.49E+01
Rosenbrock	1.78E+03	7.31E+05	1.86E+05	9.97E+02	1.11E+05	3.33E+05	9.58E+02	5.17E+04	2.65E+05
Ackley	5.11E-01	2.06E-01	1.54E-01	4.66E-01	2.72E-01	1.54E-01	4.84E-01	3.87E-01	1.96E-01

Table 2 : Benchmark : Moyennes sur 10 itérations des Écart-types de 30,50,70 de population et 10,30,50 de dimension avec le langage C++ ainsi que les 4 fonctions tester et

	C++								
population	30			50			70		
dimension	10	30	50	10	30	50	10	30	50
Sphère	1.01E+01	2.13E+01	4.34E+01	4.39E+00	5.18E+01	3.61E+01	2.62E+00	5.91E+01	5.19E+01
Rastrigin	6.00E+00	2.65E+01	5.02E+01	5.69E+00	3.88E+01	3.41E+01	5.15E+00	3.82E+01	2.56E+01
Rosenbrock	4.68E+03	6.47E+04	6.09E+04	4.32E+03	1.12E+05	4.70E+04	4.00E+02	7.23E+04	5.22E+04
Ackley	7.91E-01	7.03E-01	6.74E-01	6.24E-01	7.81E-01	2.91E-01	1.49E+00	9.76E-01	7.98E-01

C. Conclusion

En conclusion les algorithmes bio inspirée permettent de grande avancée dans le monde de l'optimisation en informatique en s'inspirant de phénomène naturel comme dans notre exemple les lucioles qui attirent des partenaires pour s'accoupler ou des victimes potentielles. Chaque phénomène peut être utile en fonction du cas on s'en ai bien rendu compte avec les expérimentations effectuer que en foncntion de la fonction objectifs les résultats étais plus ou moins bon . Ce qui montre bien la limite de l'utilisation de ces algorithme qui seront les meilleurs dans des cas d'utilisation précis l'algorithme des lucioles a par exmples été utilisé pour résoudre un problème dans le domaine du soin a domicile [4]
.

REFERENCES

[1] Krishna Durbhaka, Gopi & Selvaraj, Barani & Nayyar, Anand. (2019). Firefly Swarm: Metaheuristic Swarm Intelligence Technique for Mathematical Optimization: Proceedings of ICDMAI 2018, Volume 2. 10.1007/978-981-13-1274-8_34.

[2] Balande, U. Shrimankar, D. SRIFA: Stochastic Ranking with Improved-Firefly-Algorithm for Constrained Optimization Engineering Design Problems. Mathematics 2019, 7, 250.

[3] Beghoura Mohamed Amine, mémoire : Segmentation multi sources des images Satellitaire par l'algorithme Firefly, UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE d'ORAN, 2012.

[4] Latifa Dekhici, Rabeh Redjem, Khaled Belkadi, Abderrahman El Mhamedi. Algorithme de Lucioles pour le Soin à Domicile. Internationnal conference on complex systems and logistic; ICOSYL 2018, Apr 2018, Paris, France. ffhal-02473691f