|  |  |
| --- | --- |
| Résultat de recherche d'images pour "bee colony"  Intelligence artificielle  Artificial Bee Colony Algorithm | Olivier DELIERRE  Michael GERWILL  Ilyasse TISSAFI IDRISSI  Antonin LECLERC |

ENSEIGNANT : L.IDOUMGHAR

Table des matières

[I) Introduction 2](#_Toc502434498)

[II) L’algorithme ABC 2](#_Toc502434499)

[III) Organisation 2](#_Toc502434500)

[IV) Description du programme 3](#_Toc502434501)

[V) Benchmarks 4](#_Toc502434502)

[VI) Conclusions 8](#_Toc502434503)

# Introduction

Dans la nature, différentes espèces vivent en groupe : les bancs de poissons, les nuées d’oiseaux, les troupeaux, les abeilles… Ces dernières sont très bien organisées et très rigoureuses dans leur travail. Des chercheurs se sont donc intéressé au comportement des abeilles pour créer une métaheuristique.

Une métaheuristique est un algorithme d’optimisation permettant de résoudre des problèmes d’optimisation difficile pour lesquels on ne connaît pas de méthode classique plus efficaces. Nous allons donc dans ce rapport parler de l’algorithme ABC (Artificial Bee Colony).

Dans cet algorithme on distingue trois types d’abeilles : les employées, les éclaireuses et les spectatrices qui ont chacune une tâche bien définie.

# L’algorithme ABC

L’algorithme simule le comportement d’une colonie d’abeilles, il y a donc trois types d’abeilles :

* Les employées : cherchent de la nourriture autour de la source de nourriture qu’elles ont en mémoire et partagent ces informations avec les abeilles spectatrices.
* Les spectatrices : choisissent les meilleures sources de nourriture parmi celles que les abeilles employées leur ont transmises.
* Les éclaireuses : anciennes abeilles employées dont la source de nourriture n’a pas été retenue, elles cherchent donc aléatoirement une nouvelle source de nourriture.

# Organisation

# Description du programme

Le programme est composé de cinq classes, quatre imposées et une dernière créée par nous-même afin de mieux pouvoir décomposer le programme qui sont :

* MyAlgorithm
* Solution
* Problem
* Benchmark
* SetUpParams

Toute ces classes disposant de getter, setter ainsi que de surcharge d’opérateurs pour certains cas afin de faciliter les manipulations lors de l’exécution de l’algorithme.

MyAlgorithm :

C’est la classe principale qui contient l’algorithme, ainsi que les paramètres et les solutions.

Fonctions principales :

-initialize(), fonction qui initialise l’algorithme, avec toute ses solutions, et rempli le vecteur de fitness une première fois utilisation la fonction evaluate().

-evaluate(), fonction qui évalue pour l’algorithme, elle calcule les fitness.

-evolution(), elle commence l’algorithme avec ses plusieurs étapes, initialisation, puis pour chaque étape elle : évalue, transmets les informations, envoie les abeilles employées, puis les spectatrices et enfin les éclaireuses. Elle répète ces actions un nombre de fois définis dans les paramètres.

-send\_bees(), fonction qui effectue les mutations en fonction des paramètres reçus, on utilise donc cette fonction pour les trois types d’abeilles car chaque abeille a un rôle particulier

-send\_employed\_bees(), fonction qui fais des mutations avec un paramètre aléatoire et les change en utilisant send\_bees().

-send\_onlooker\_bees(), fonction qui fais des mutations en utilisant un paramètre choisi par calculate\_probabilities() et les change en utilisant send\_bees().

-send\_scout\_bees(), fonction qui regarde si la solutions a passé le maximum d’essais et le réinitialise si c’est le cas afin de ne pas rester coincé dans des extremums locaux.

-calculate\_probabilities(), fonction qui calcul les probabilités pour chaque solutions choisis par les abeilles spectatrices utilisant une fonction mathématique simple.

Solution :

Classe qui contient la solution, la meilleur fitness calculée par la solution, la meilleure calculée par l’algorithme, ainsi que diverses fonctions qui par exemple renvoie les fitness.

Problem :

Classe qui contient la dimension du problème, les limites du problème (qui diffèrent en fonction du benchmark utilisés) ainsi qu’une variable du benchmark choisi.

Benchmark :

Classe qui contient toutes les fonctions de benchmark, séparées du reste du programme afin de permettre un code plus propre et une utilisation plus fluide.

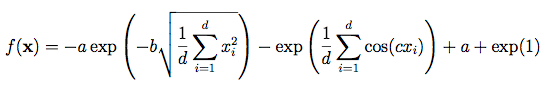
SetUpParams :

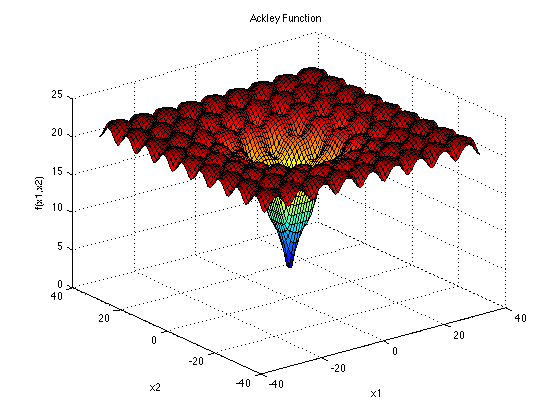
Classe qui contient les données essentielles à la résolution du problème telles que le nombre d’exécutions indépendantes, le nombre d’évolution, la taille des populations, la taille des solutions et enfin le nombre d’appel de la fonction.

# Benchmarks

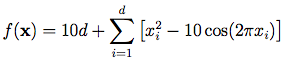
Pour ce projet il nous a été demandé d’utiliser six fonctions à optimiser :

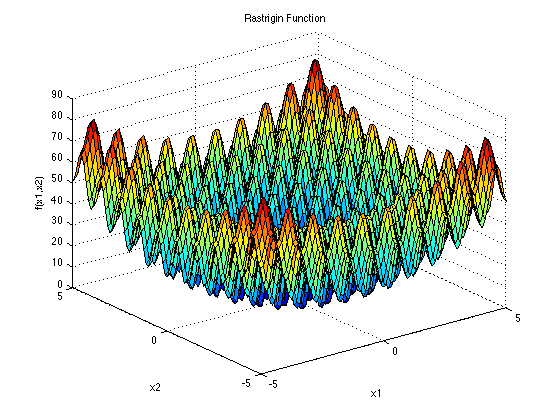
* Ackley



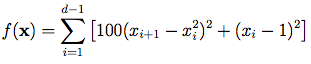


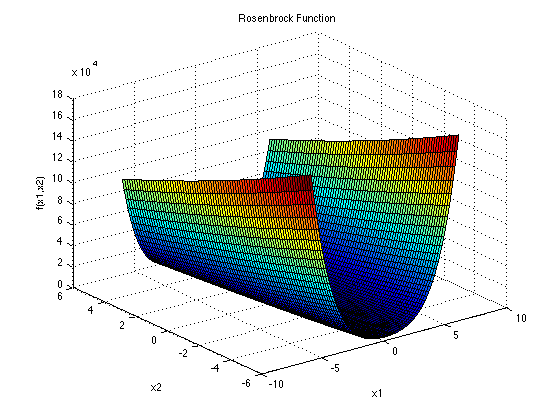
* Rastrigin





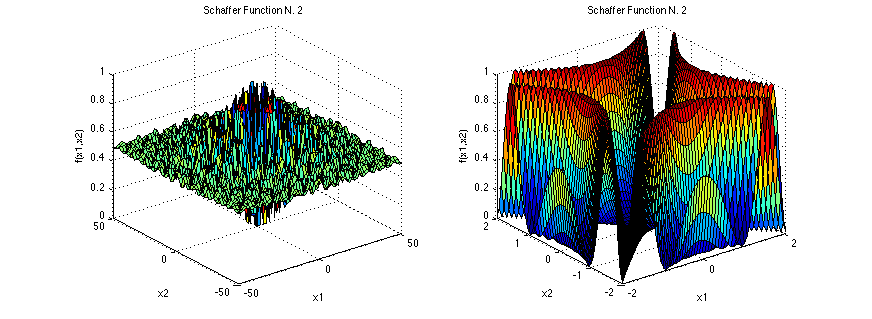
* Rosenbrock





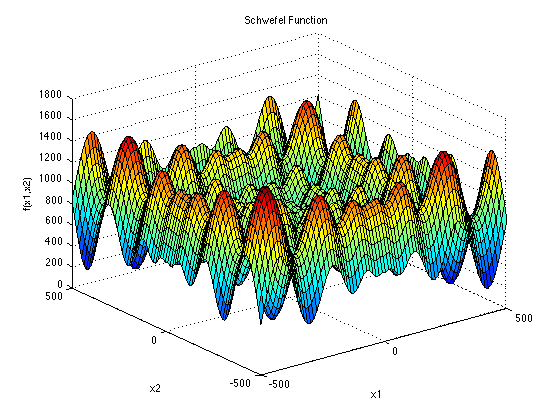
* Schaffer



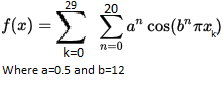


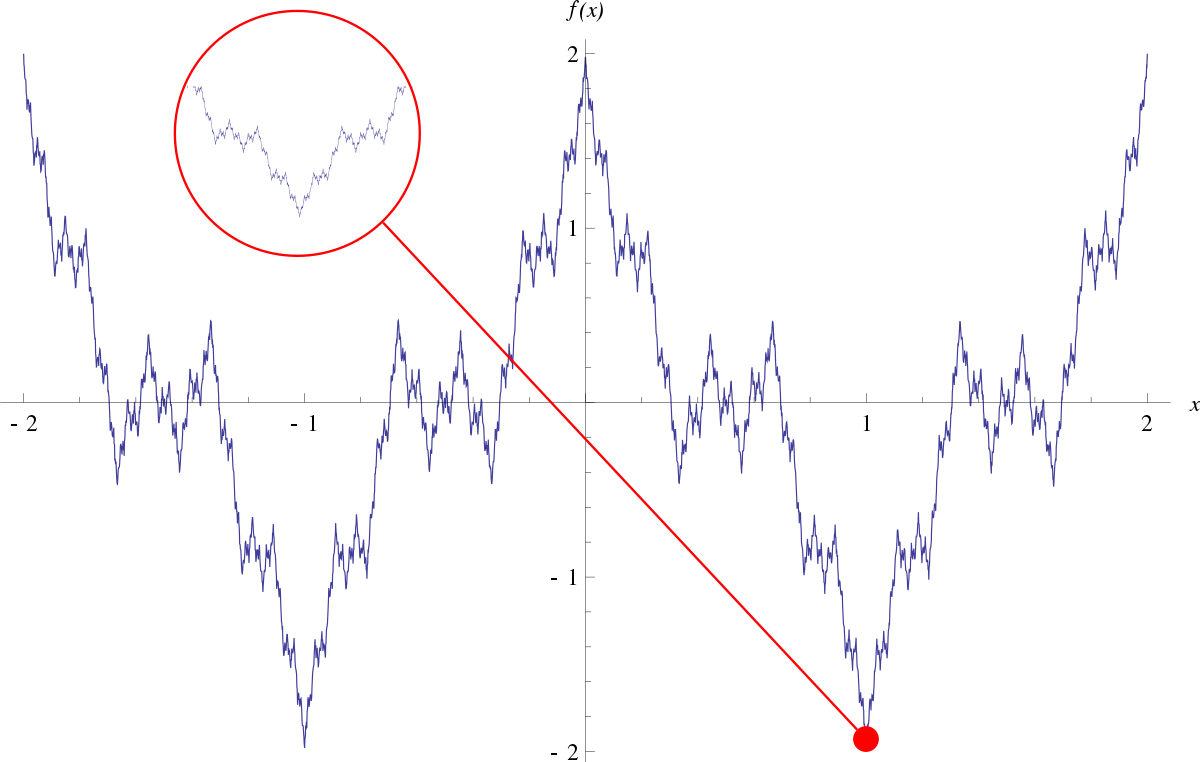
* Schwefel





* Weierstrass





# Conclusions