2017-2018

Rapport de projet

Gravitational Search Algorithm

AGUDO-PEREZ Olivier, GHERARDI Sylvain, LIEVIN Leonard
UNIVERSITE DE HAUTE-ALSACE

Sommaire

I. Introduction

Dans le cadre du projet d'intelligence artificielle nous avons été amenés à recréer un algorithme de recherche par essaim particulaire, qui s'inspire du mouvement des planètes dans l'espace.

Il existe un grand nombre de méthodes d'optimisations heuristiques, beaucoup d'entre elles se basent sur des comportements d'essaims (swarm) et sont largement inspirés par les mêmes comportements que l'on peut retrouver dans la nature.

Cette idée de métaheuristique met en avant le fait qu'un groupe d'individus peu intelligents peut posséder une organisation globale complexe, et, qu'avec des règles de déplacement très simples, les individus vont converger vers un minimum.

L'algorithme sur lequel nous avons travaillé est le "Gravitational Search Algorithm", abrégé "GSA", initialement imaginé par Esmat Rashedi, docteur à la Graduate University of Advanced Technology en Iran. L'algorithme se présente donc sous la forme d'une population de 30 individus représentés par des planètes, le but est de mouvoir les planètes en calculant différentes données tel que la masse et l'accélération des solutions en utilisant les lois de Newton et les lois du mouvement. L'ensemble des solutions vont donc s'attirer et converger vers un point.

II. Répartitions des tâches

Olivier

- MyAlgorithm::afficher_best()
- MyAlgorithm::afficher_all()
- MyAlgorithm::change_parameters()
- MyAlgorithm::best_cost_overall()
- Solution::mass_calculation()
- Solution::update_solution()
- Doxygen

Sylvain

- afficheur::init()
- afficheur::afficher_pbm()
- afficheur::afficher_menu_principal()
- afficheur::start_resolution()
- MyAlgorithm::MyAlgorithm()
- MyAlgorithm::evolution()
- MyAlgorithm::spaceBound()
- MyAlgorithm::g_evolution()
- Solution::Solution()
- Solution::acceleration_calculation()
- PowerPoint

Léonard

- afficheur::afficher_menu_edit()
- MyAlgorithm::~MyAlgorithm()
- MyAlgorithm::initalize()
- MyAlgorithm::upper_cost()
- MyAlgorithm::lower_cost()
- Solution::fitness()
- Solution::initialize()
- PowerPoint

Commun

- Getters, setters et opérateurs de flux
- Test des fonctions avec différents paramètres
- Rapport

III. Fonctions principales

a. MyAlgorithm::evolution

La méthode “évolution” va s’occuper de faire “avancer” les solutions de l’algorithme.

En premier lieu on procède à l’évaluation des solutions t-1, on calcule ensuite les masses gravitationnelles de chaque solution ainsi que les masses inertielles, on fait évoluer la constante g, puis on calcule la nouvelle accélération pour chaque individu, enfin, on procède à la mise à jour des positions qui fait office de mouvement des individus.

b. Solution::fitness

Elle calcule la fitness d’une solution en fonction du numéro du problème.

La fitness est la capacité d’une solution à répondre à un problème donné.

Le calcul de la fitness varie en fonction de la fonction objective, pour la calculer on utilise, pour chaque dimension, la position de la solution dans la fonction de benchmark correspondante, puis on les somme afin d’obtenir une valeur. Dans le cas d’une minimisation, plus la valeur est proche de 0, meilleure est la solution.

c. Solution::mass_calculation

Chaque individu possède 2 masses, une masse inertielle et une masse gravitationnelle.

La masse inertielle est calculée par la division de la masse gravitationnelle de la solution en question sur la somme des masses de toutes les solutions de la résolution.

La masse gravitationnelle est calculée comme suit :

Masse = (fitness - pireFitness) / (meilleureFitness - pireFitness)

- fitness : la fitness de la solution courante
- meilleure Fitness : la meilleure fitness parmi les solutions de la population courante
  PireFitness : la pirefitness parmi les solutions de la population courante

d. MyAlgorithm::g_evolution

La constante de gravitation G permet de faire diminuer l'ampleur des déplacements au cours de la résolution du problème. Cette constante, qui va varier tout au long du problème, est définie par deux autres constantes, g0 et alpha, qui sont inchangées durant toute l'exécution.

g0 est la valeur de base de la constante g à la première itération.

Alpha est le "coefficient" de diminution de la constante g, plus alpha est grand, et plus la valeur de g descendra rapidement, et par extension, les mouvements des solutions seront de plus en plus réduits. Et par opposition, plus alpha sera petit et moins la constante g évoluera, les solutions auront donc des déplacements de grande amplitude même à la fin de la résolution. L'objectif est donc de trouver la bonne combinaison de g0 et alpha qui permet aux planètes de converger rapidement vers la solution optimale dès les premières itérations, puis d'affiner plus lentement ce résultat, quel que soient les limites du problème.

Le calcul de la constante g se fait donc de la sorte : $g0 * e^{-alpha * itération / nb\_runs}$

e. Solution::acceleration_calculation

Calcule l'accélération d'une planète en fonction des autres planètes.

Une planète possède une composante d'accélération par dimension, donc pour calculer l'accélération globale d'une planète il faut calculer l'accélération de chaque dimension qui dépend de la masse de chaque planète et de la position sur cette dimension de chaque planète.

Cette accélération sur une dimension se calcule de la manière suivante :

Accélération = (r * G * inertia_mass * (xi - xj)) / (distance + e)

- r : une valeur aléatoire entre 0 et 1 qui permet d'introduire une mutation des solutions
- G : la constante gravitationnelle
- distance : distance euclidienne classiquement utilisée dans la formule de Newton
- (xi - xj) : distance entre les deux positions correspondant à la dimension, cette valeur n'est pas utilisée dans la formule classique du calcul de l'accélération, mais elle est nécessaire dans le cadre de l'algorithme pour prendre en compte les différentes dimensions

f. Solution::update_solution

Mise à jour de la vélocité et de la position de la solution

La nouvelle vélocité est calculée en fonction de l'ancienne vélocité multipliée par un nombre aléatoire entre 0 et 1 afin de faire peser plus ou moins l'ancienne vitesse dans le calcul, auquel on additionne l'accélération de la solution.
La nouvelle position est calculée en additionnant l'ancienne position et la vitesse.

IV. Nos résultats

Sauf mention contraire tous les tests sont réalisés avec les paramètres suivants :

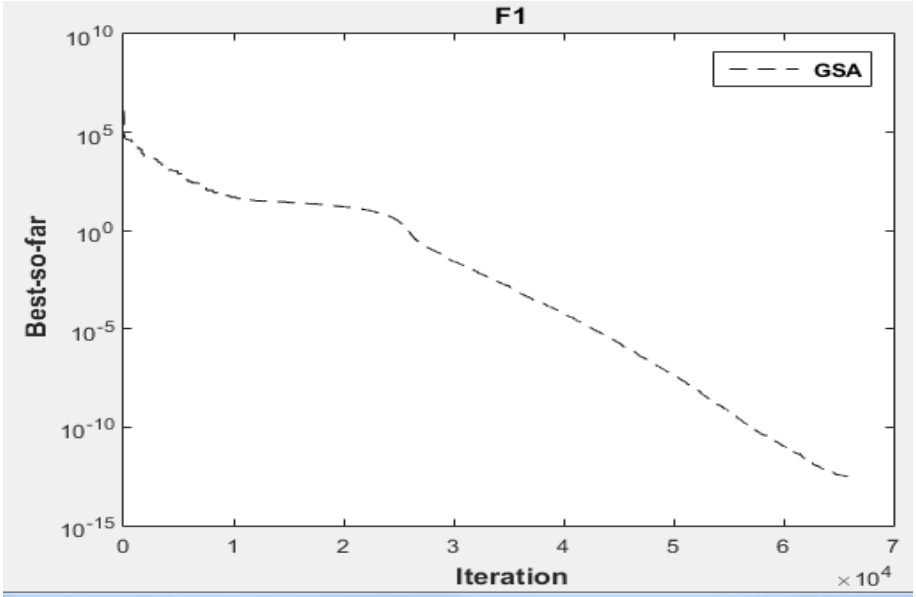
nombre d'appels de la fonction objectif	2e6
dimensions	30
individus	30
g0	100
alpha	20

A. Rosenbrock

```
Rosenbrock, premiere solution :
Fitness actuelle : 2.63859e+008

0                                                                                100%
--------------------------------------------------------------------------------

meilleure solution :
Fitness actuelle : 27.7646
```




```
Fbest =

   3.0951e-13
```

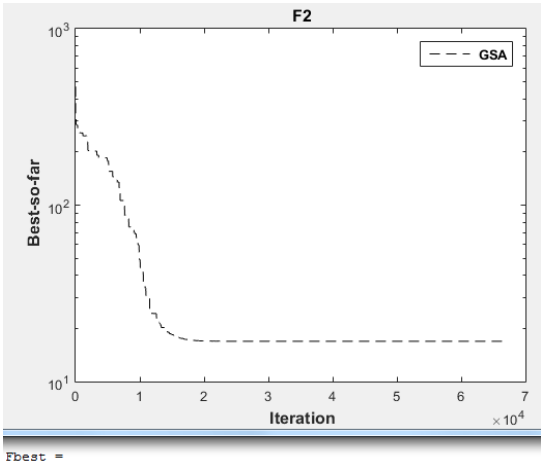
Pour cette fonction le code Matlab proposé par Esmat Rashedi est bien plus performant que notre algorithme. Une raison potentielle pour cette différence est que notre algorithme est moins performant pour explorer l'espace et sort plus difficilement d'un minimum local. Ce problème serait en plus accentué par le fait que cette fonction possède une grande zone de faibles valeurs où le minimum est difficile à distinguer.

B. Rastrigin

```
Rastrigin, premiere solution :
Fitness actuelle : 478.033

0                                                                                                    100%
_____________________________________________________________________________________________________

meilleure solution :
Fitness actuelle : 13.9294
```




```
Fbest =

    16.9143
```

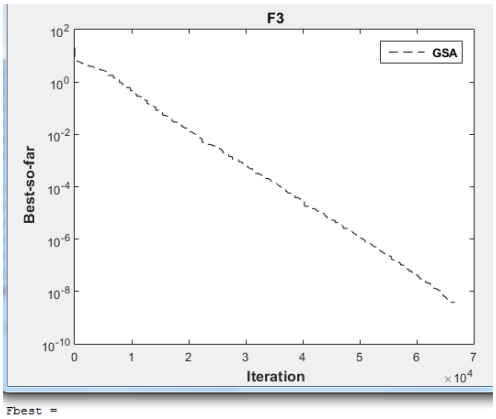
Pour cette fonction le résultat trouvé est légèrement meilleur que le résultat trouvé via Matlab, cela correspond à nos attentes.

C. Ackley

```
Ackley, premiere solution :
Fitness actuelle : 21.5

0                                                                                                    100%
_____________________________________________________________________________________________________

meilleure solution :
Fitness actuelle : 1.20123e-008
```




```
Fbest =

    3.8778e-09
```

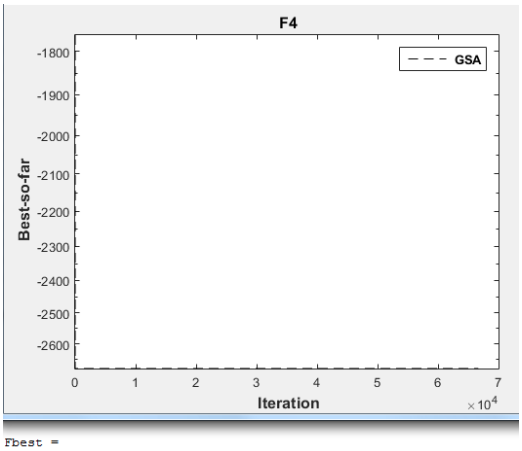
Pour cette fonction aussi le résultat trouvé correspond à nos attentes, mais on remarque que notre solution est légèrement moins bonne que l'algorithme Matlab.

D. Schwefel

```
Schwefel, premiere solution :
Fitness actuelle : 13171.1

0                                                                                                    100%
_____________________________________________________________________________________________________

meilleure solution :
Fitness actuelle : 9811.29
```




```
Fbest =

  -2.6810e+03
```

Pour cette fonction, comme pour la fonction Rosenbrock, notre algorithme a beaucoup plus de mal à converger vers la solution attendue.

E. Schaffer 2

```
Schaffer, premiere solution :
Fitness actuelle : 0.524719

0                                                                                                    100%
_____________________________________________________________________________________________________

meilleure solution :
Fitness actuelle : 2.07182e-005
```

Nous n'avons pas de test via Matlab pour comparer à ce résultat, mais on peut tout de même noter que l'amélioration de la fitness est significative.

F. Weierstrass

```
Weierstrass, premiere solution :
Fitness actuelle : 0.0763407

0                                                                                                    100%
_____________________________________________________________________________________________________

meilleure solution :
Fitness actuelle : -3.33298
```

V. Difficultés

Pendant les phases de test nous avons remarqué que notre algorithme converge parfois vers le résultat attendu mais converge très mal d'autre fois. L'algorithme GSA utilisé seul est connu pour avoir tendance à rester bloquer dans des minimums locaux, mais cela ne suffit pas à expliquer les fitness très grandes que nous trouvons parfois.

Malheureusement nous n'avons pas réussi à trouver de solution à cette divergence de la qualité de nos résultats, en effet l'algorithme proposé par Esmat Rashedi arrive à trouver de bons résultats sans changer ses paramètres sur certaines fonctions où notre algorithme n'y parvient pas.

VI. Conclusion

Pour conclure, on peut dire que le projet est principalement une réussite. Nous avons su gérer le temps et finir l'algorithme dans le temps imparti, de plus l'algorithme est entièrement fonctionnel. Nous avons su intégrer toutes les fonctions de benchmark ainsi que toutes les étapes d'évolution. Nos résultats sont très similaires avec ceux trouvés sur Matlab sur 2 fonctions objectives. Nous avons su comprendre le fonctionnement de l'algorithme malgré le peu de documentation sur le sujet du fait de sa relative jeunesse.