



# PROJET IA

## Gravitational Search Algorithm

Olivier Agudo-Perez  
Sylvain Gherardi  
Léonard Liévin

L3 - MIAE - 2017 / 2018

# Origine de l'algorithme

- Le GSA est proposé par **Esmat Rashedi** en 2009
- Esmat Rashedi : professeur à l'université de Kerman en Iran
- Basé sur la **loi de gravitation** et les **interactions entre les masses**
- La “population” est un ensemble de **planètes** qui interagissent entre elles en fonction de la **force gravitationnelle**
- La performance est mesurée par la masse de la planète



# Principe du GSA

- Une planète est un **objet**, ces derniers s'attirent grâce à la force gravitationnelle
- Une planète qui se déplace représente l'**étape d'exécution** du GSA
- La **meilleure solution** (*best fitness*) est la solution avec la masse la plus lourde





# Masse des planètes

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^N m_j(t)}$$

$m_i$  = masse de la planète i

$M_i$  = masse inertielle de la planète i

$fit_i$  = fitness de la planète i

$worst$  = plus mauvaise fitness de la population

$best$  = meilleure fitness de la population



# La constante gravitationnelle G

$$G(t) = G_0 e^{-\alpha \frac{t}{T}}$$

$G_0 = 100$

$\alpha = 20$

$T$  = nombre d'exécutions

$t$  = itération courante

t	G
0	100.0000000
1	10.8368023
2	1.1743628
3	0.1272634
4	0.0137913
5	0.0014945
6	0.0001620
7	0.0000176
8	0.0000019
9	0.0000002

# Force gravitationnelle des planètes

$$F_{ij}^d = G(t) \frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij} + \varepsilon} (x_j^d(t) - x_i^d(t))$$

$$F_i^d(t) = \sum_{j \in Kbest, j \neq i}^N rand_j F_{ij}^d(t)$$

$F_{ij}^d$  = force entre la masse i et la masse j dans la dimension d

$F_i^d$  = force totale appliquée à i dans la dimension d

$M_{aj}$  = masse gravitationnelle active

$M_{pi}$  = masse gravitationnelle passive

$R_{ij}$  = distance euclidienne entre i et j

$x_j^d$  = position de j dans la dimension d

$x_i^d$  = position de i dans la dimension d

# Accélération gravitationnelle

$$a_i^d = \frac{F_i^d(t)}{M_{ii}(t)}$$

$M_{ii}$  = masse inertielle

$F_i^d$  = force totale sur la dimension d





# Changement de vitesse et de position

Pendant l'algorithme, on met à jour ses valeurs :

Vitesse : 
$$v_i^d(t + 1) = rand_i v_i^d(t) + a_i^d(t)$$

Position : 
$$x_i^d(t + 1) = x_i^d(t) + v_i^d(t + 1)$$





# Répartition des tâches

## Olivier

- MyAlgorithm::afficher\_best()
- MyAlgorithm::afficher\_all()
- MyAlgorithm::change\_parameters()
- MyAlgorithm::best\_cost\_overall()
- Solution::mass\_calculation()
- Solution::update\_solution()
- Doxygen

## Sylvain

- MyAlgorithm::MyAlgorithm()
- MyAlgorithm::evolution()
- MyAlgorithm::spaceBound()
- MyAlgorithm::g\_evolution()
- Solution::Solution()
- Solution::acceleration\_calculation()
- PowerPoint

## Léonard

- MyAlgorithm::~~MyAlgorithm()
- MyAlgorithm::inititalize()
- MyAlgorithm::upper\_cost()
- MyAlgorithm::lower\_cost()
- Solution::fitness()
- Solution::initialize()
- PowerPoint



# Classe Solution

Une Solution est représentée par une planète, elle contient :

- Une **position** en  $d$  dimensions
- Une **vitesse** sur  $d$  dimensions
- Une **accélération** sur  $d$  dimensions
- Une **masse gravitationnelle**
- Une **masse inertielle**
- Une **fitness**



# Classe Problem

Un problème représente une fonction objectif, elle est composée de :

- D'un **nom** et d'un **numéro** de problème : Weierstrass, Rastrigin, ... par exemple
- D'une **direction** : maximisation ou minimisation
- De **limites inférieures** et **supérieures**
- D'un nombre de **dimensions**



# Classe MyAlgorithm

La classe MyAlgorithm représente un ensemble de Solutions ainsi que le problème associé.

Un problème contient :

- Un ensemble de planètes : la **population**
- Un problème : la **fonction à optimiser**
- La **meilleure fitness** et la **pire** de la population actuelle
- La **meilleure solution** parmi toutes les évolutions
- La constante de gravitation **G**



# Le GSA en étape

## Etape 0 : Initialisation

La **population** initiale est générée **aléatoirement**.

On **boucle** sur les **étapes 1 à 6** jusqu'à atteindre les critères souhaités.



# Le GSA en étape

## Etape 1 : Evaluation


**Evaluation** de chaque agent via les **fonctions objectif**, on cherche le meilleur et le plus petit résultat de la fitness.

## Etape 2 : Masses

Calcul des **masses** et masses **inertielle**s. (diapositive 4)

## Etape 3 : G

**Mise à jour** de la **constante gravitationnelle**. (diapositive 5)



# Le GSA en étape

## Etape 4 : Forces

Calcul des **forces** agissant sur chaque planète. (diapositive 6)

## Etape 5 : Accélération

Calcul de l'**accélération**. (diapositive 7)

## Etape 6 : Déplacement

**Mouvement** des planètes. (diapositive 8)



# Le GSA en étape

## Etape 7 : Fin

On retourne la **solution optimale** obtenue au cours de l'algorithme.







## Nos résultats

Rosenbrock : 27.7646

Rastrigin : 13.9294

Ackley :  $1.20123 \times 10^{-8}$

Schwefel : 9811.29

Schaffer :  $2.07182 \times 10^{-5}$

Weierstrass : -3.33298

## Résultats Matlab

$3.095 \times 10^{-13}$

16.9143

$3.8778 \times 10^{-9}$

-2680



Merci de votre attention