Fabien PETIT

Résolution de problèmes complexes

TP Scorpion

I4
Session 2016 - 2017

EPSI l'École d'ingénierie informatique

1. Mise à disposition du code - GitHub

Mon code est disponible sur la plateforme GitHub au lien suivant :
https://github.com/fab37100/TP-scorpion

2. Choix du langage

Le python est utilisé car il est performant dans l'exécution d'algorithme, dispose d'une syntaxe simple et qu'il était imposé dans ce TP.

3. Structure de donnée

Une structure de données indexées est utilisée dans mon code. Cela m'a semblé plus simple à mettre en œuvre et moins gourmand en termes de ressources.

J'utilise des vecteurs, vecteurs multidimensionnels et des tableaux associatifs (Dictionnaires) dans les traitements.

4. Composition du script

Le script Python comprend 4 fichiers :

- Main.py
- Config.py
- Maths.py
- Fonctions.py

a) Fichier « main.py »

Il s'agit ni plus ni moins du fichier maitre a exécuté pour lancer le script. Il comprend la méthodologie d'exécutions du script et fera appelle aux autres fichiers.

b) Fichier « config.py »

La configuration s'effectue dans le fichier « config.py » qui comprend toutes les constantes de paramétrage du script. A chaque exécution, elles seront chargées dans le script et utilisées dans son traitement, il est donc important d'y définir des valeurs plausibles afin d'obtenir des résultats.

Nom constante	Valeurs par défaut	Description de l'utilisation
NB_POPULATION	2000	Nombre d'individu par génération
NB_GEN	200	Nombre de génération maximum
DISTANCE_CIBLE	301	Position de la cible à toucher
COUPE_CROISEMENT	4	Chance de croisement autre que la moitié. *Valeur max à ne pas dépasser : 10 (nombre de gêne)*
MAX_SCORE_FITNESS	1000	Score maximal pour le calcul de la portée
POURCENTAGE_MUTATION	0.5	Pourcentage de mutation par individu

MIN_ANGLE_HAUSSE	1	Permet de générer les valeurs aléatoires de l'angle de hausse pour chaque individu de la première génération ou des mutations entre la valeur minimale et maximale.
MAX_ANGLE_HAUSSE	85	
MIN_LONGUEUR_BRAS	1	Permet de générer les valeurs aléatoires de la longueur de bras pour chaque individu de la première génération ou des mutations entre la valeur minimale et maximale.
MAX_LONGUEUR_BRAS	60	
MIN_SECTION_BRAS	0.01	Permet de générer les valeurs aléatoires de la base de la section du bras pour chaque individu de la première génération ou des mutations entre la valeur minimale et maximale.
MAX_SECTION_BRAS	10	
MIN_HAUTEUR_SECTION	0.01	Permet de générer les valeurs aléatoires de la hauteur de la section du bras pour chaque individu de la première génération ou des mutations entre la valeur minimale et maximale.
MAX_HAUTEUR_SECTION	10	
MIN_LONGUEUR_CORDE	5	Permet de générer les valeurs aléatoires de la longueur de la corde (en mètre) pour chaque individu de la première génération ou des mutations entre la valeur minimale et maximale.
MAX_LONGUEUR_CORDE	60	
MIN_LONGUEUR_FLECHE	5	Permet de générer les valeurs aléatoires de la longueur de la flèche (en mètre) pour chaque individu de la première génération ou des mutations entre la valeur minimale et maximale.
MAX_LONGUEUR_FLECHE	60	
MIN_MASSE_VOL_FLECHE	600	Permet de générer les valeurs aléatoires de la masse volumique de la flèche pour chaque individu de la première génération ou des mutations entre la valeur minimale et maximale. *Les valeurs correspondent à du bois.*
MAX_MASSE_VOL_FLECHE	1000	
MIN_YOUNG_MATERIAU	0.15	Permet de générer les valeurs aléatoires du module de Young du matériau de l'arc pour chaque individu de la première génération ou des mutations entre la valeur minimale et maximale. *Les valeurs correspondent à du bois.*
MAX_YOUNG_MATERIAU	650	
MIN_COEF_POISSON	0.03	Permet de générer les valeurs aléatoires du coefficient de Poisson pour chaque individu de la première génération ou des mutations entre la valeur minimale et maximale. *Les valeurs correspondent à du bois.*
MAX_COEF_POISSON	0.44	
MIN_BASE_FLECHE	0.05	Permet de générer les valeurs aléatoires de la base de la flèche pour chaque individu de la première génération ou des mutations entre la valeur minimale et maximale.
MAX_BASE_FLECHE	2	
MIN_HAUTEUR_FLECHE	1	Permet de générer les valeurs aléatoires de la hauteur de la flèche pour chaque individu
MAX_HAUTEUR_FLECHE	3	

		de la première génération ou des mutations entre la valeur minimale et maximale.
GRAVITE_TERRE	9.81	Correspond à la gravité terrestre

c) Fichier « maths.py »

Ce fichier comprend toutes les contraintes métiers pour effectuer un tir (limites) ainsi que toutes les formules mathématiques nécessaire au calcul de la portée et de la puissance du tir. Il contient aussi la fonction « Normale » qui est utilisé pour le calcul de la fitness et ainsi que le calcul de la variance.

d) Fichier « fonctions.py »

Ce fichier est très important, car, il se compose de toutes les fonctions qui assureront le traitement du script.

Dedans, on peut retrouver les fonctions de création des gènes, la génération de la première population, la sélection des parents, la création des enfants, etc.

5. Fonctionnement des fonctions

a) Création de la première génération

Un individu est composé de 10 gènes à savoir :

- L'angle de hausse
- La longueur de bras
- La base de la section du bras
- La hauteur de la section du bras
- La longueur de corde
- La longueur de flèche
- La masse volumique des flèches
- Le module de Young
- Le coefficient de Poisson
- La base de la flèche
- La hauteur de la flèche

Chacun de ces gênes est défini de façon aléatoire dans l'intervalle des valeurs définies dans le fichier des paramètres de configuration.

b) Calcul de la Fitness

Lors de la réflexion sur la création de la fonction Fitness, je me suis penché sur un système de pourcentage linéaire pour la définition du score des individus pour la portée du tir. Cependant, cela n'était pas adapté car pour les individus disposant d'un tir proche de la cible, ils ne se démarqués pas assez des mauvais. J'ai donc pensé à une fonction exponentielle.

Mes recherches mon amené sur la fonction « Normale » qui dispose d'une belle courbe exponentielle vers le point maximal (cible à atteindre) et qui se désagrège quand on s'en éloigne et cela de n'importe quel côté du point. (Tir plus loin ou moins loin)

$$f(x) = e^{\frac{-1}{2}\left(\frac{(x-\mu)}{\sigma}\right)^2} 1000 + 0{,}05$$

μ = Distance de la cible
σ = μ / 7
x = la portée de l'individu

Sigma *σ* est divisé par 7 afin d'obtenir une courbe avec une forte exponentielle et donc privilégier les individus proche de la cible en leur fournissant un score élevé. L'ensemble est multiplié par 1000 afin d'avoir un score max de 1000 pour la distance. 0,05 est également ajouté pour éviter d'attribuer la note de 0 aux individus tirant loin de la cible.

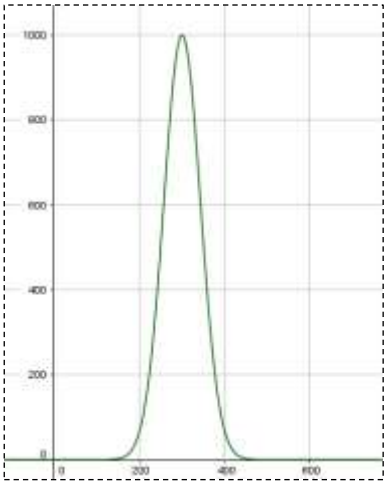

Figure 1 : Représentation graphique de la fonction

c) Génération du score par individus

Les individus sont évalués en fonction des résultats qui sont ressortis des règles métiers et des limites. Une fois cela fait, la fonction fitness est utilisée afin d'attribuer un note à chaque individu par rapport à sa portée en fonction de la distance cible. Ensuite, cette note se voit agrémentée de la puissance en TNT de l'individu se qui forme le score finale de l'individu.

Aucune limite n'est attribuée aux scores des individus.

$$ScoreIndividu = fitness(portée\ individu) + TNT$$

d) Sélections des parents

Pour la sélection des parents, j'ai utilisé la méthode proportionnelle. Plus précisément la « Roulette Wheel Selection ». Elle est légèrement différente de la « Stochastic Universal Sampling » qui définit un offset aléatoire en début de lecture du tableau et saute d'individus en individus en fonction de la valeur de l'offset jusqu'à la fin du tableau. Cette méthode est efficace, peu couteuse en ressource et propose une faible variance.

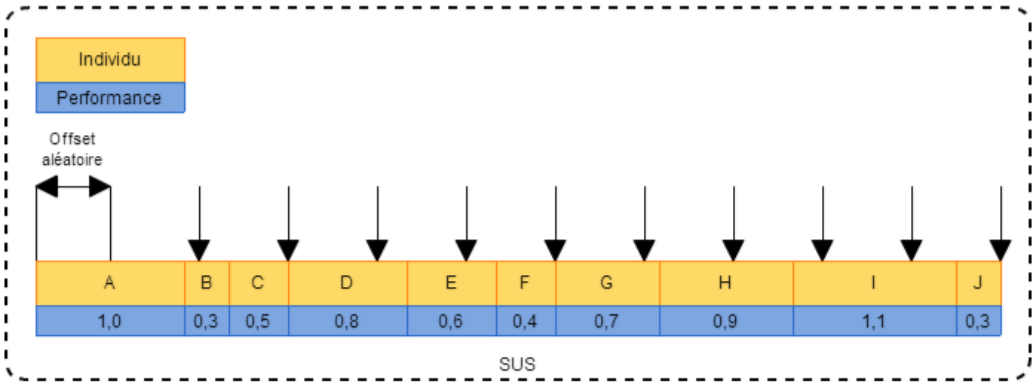

Figure 2 : Sélection « Stochastic Universal Sampling »

J'ai préféré utilisé la sélection RWS, car, elle était utilisée à l'origine des algorithmes génétiques. Cependant, le temps de traitement est plus long, du fait qu'elle définit un offset aléatoire à chaque sélection ce qui oblige de relire le tableau du début à chaque fois et la variance est plus importante (Affecte la variance pour chaque génération)

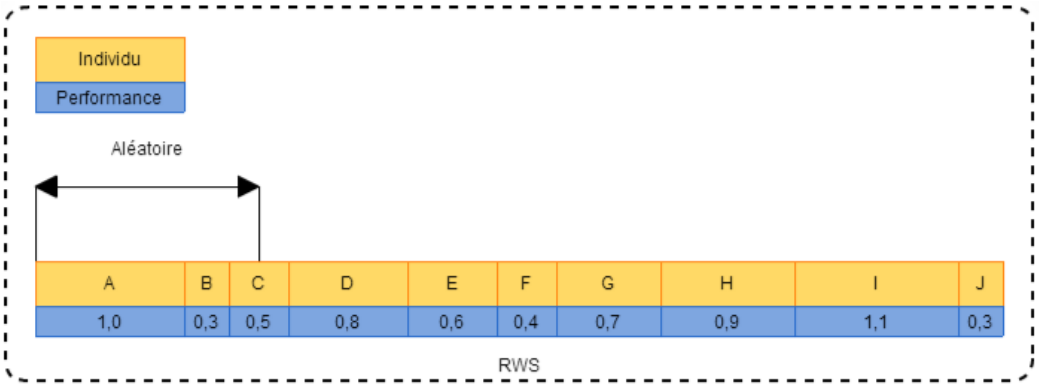

Figure 3 : Méthode de sélection utilisée « Roulette Wheel Selection »

Une contrainte est survenue lors des premiers tests. Il arrivait que le même parent soit en couple avec lui-même. Cet obstacle a été contourné en bouclant sur le deuxième parent jusqu'à qu'il ne soit plus le même. L'offset est bien entendu réinitialiser à chaque occurrence sinon cela ne respecterait pas le principe de la sélection RWS et on tomberait dans une boucle infinie (La valeur RWS désigne un parent).

e) Création des enfants

Lorsque que la sélection des parents et terminée, chaque couple forme 2 enfants qui disposent chacun de la moitié des gènes de chaque parents.

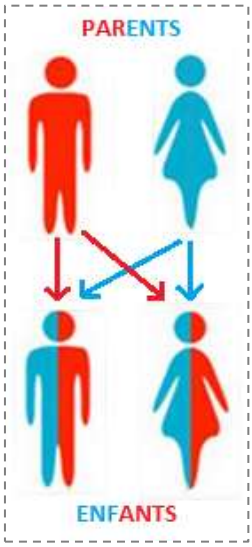

Figure 4 : Représentation de la répartition des gènes aux enfants

La gestion d'enjambement (Coupe un endroit différent que la moitié) est prise en compte avec un système aléatoire en fonction d'un pourcentage d'enjambement.

La mutation est aussi implémentée fonctionnant sur le même principe que l'enjambement en générant un nouveau gène si l'individu est sélectionné en fonction d'un pourcentage donné.

6. Jeux de données

Le script a été exécuté plusieurs fois afin de ressortir des graphes. Ils représentent :

- Le score moyen pour chaque génération
- La variance par rapport au score de chaque génération
- La portée moyenne pour chaque génération
- La moyenne de TNT par génération

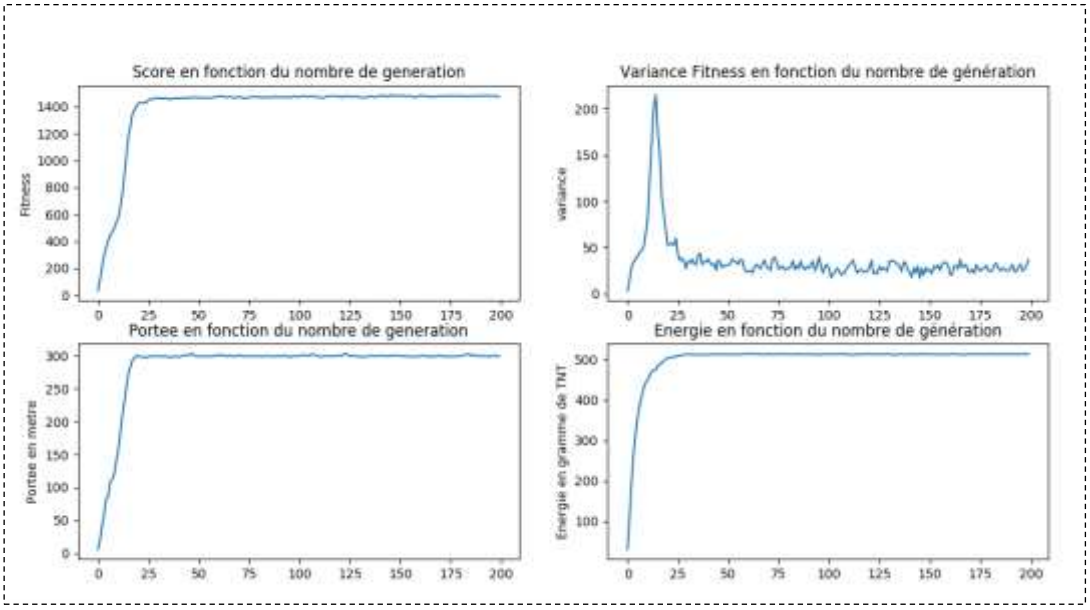

Figure 5 : Rendu sur 200 générations de 2000 individus

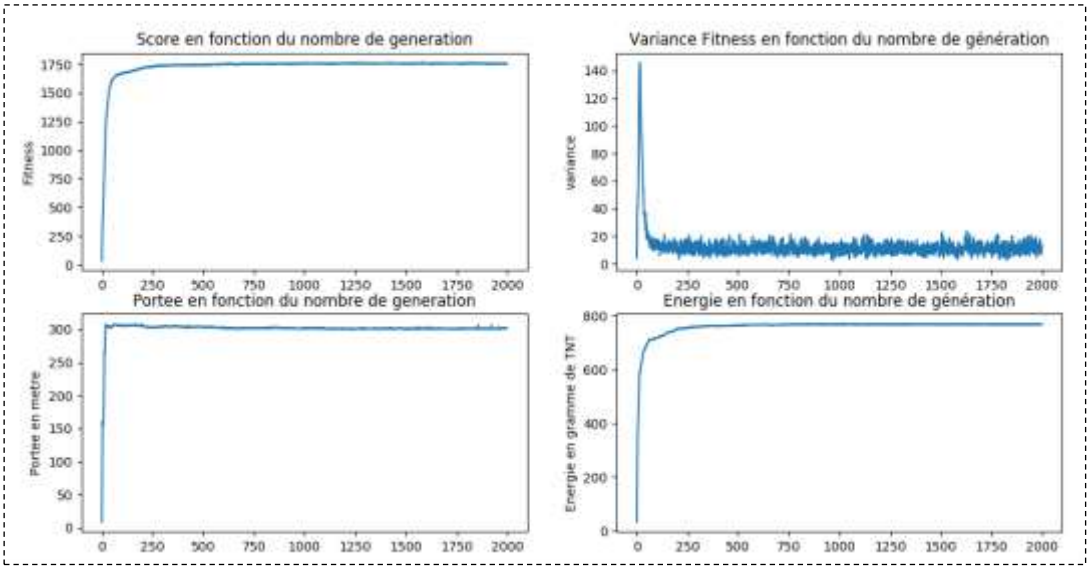

Figure 6 : Représentation sur 2000 générations de 1000 individus

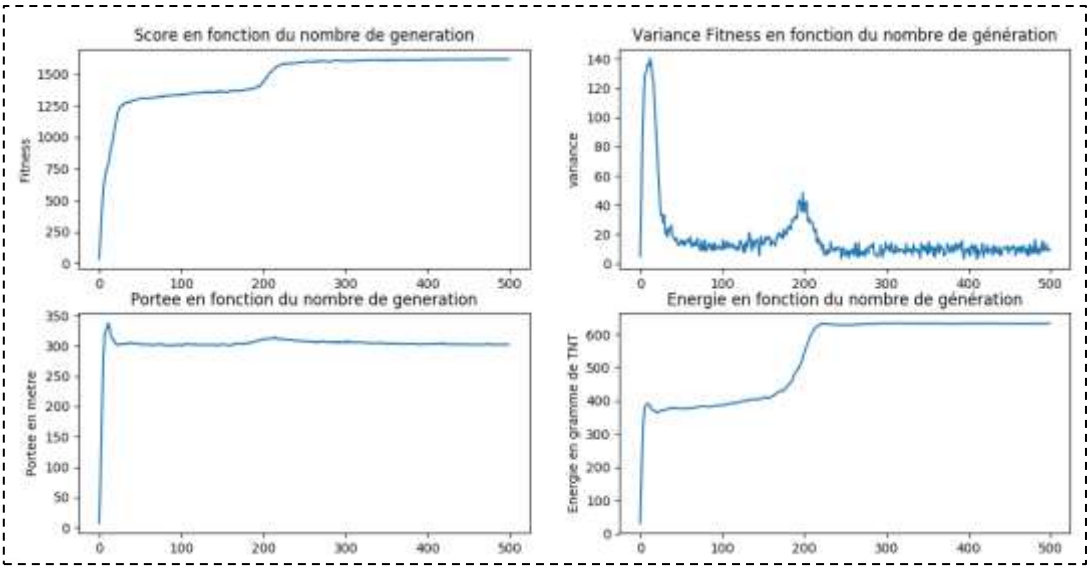

Figure 7 : Rendu sur 500 générations de 2000 individus