MARS LANDER GENETICS

Delezenne Quentin – Sebastian Mejía

Description du Projet

Le projet est celui de réaliser un apprentissage par algorithme génétique pour trouver une trajectoire d'atterrissage pour un module d'atterrissage.
L'algorithme va générer des liste d'actions prises au fur et à mesure du vol, amenant à une trajectoire précise. Le but est de trouver la liste de commande, que l'on appellera des « profils de commandes » permettant un posé dans les conditions souhaitées.

Initialisation

L'initialisation constitue une première population avec des profils de commandes uniformes dont les valeurs sont aléatoires mais dont la direction sera toujours vers la zone d'atterrissage. De cette manière, des trajectoires quasi rectilignes sont générées et permettent de trouver un potentiel début de trajectoire droit et rapide vers la zone.

Évaluation

L'évaluation d'un individu, donc d'un profil de commandes, s'effectue en faisant une simulation physique complète de son profil. L'évaluation sera simultanée avec tout sa population. Le score de l'évaluation consiste en la somme de la distance à parcourir en suivant le sol pour arriver à la zone d'atterrissage avec les valeurs absolues de vitesse horizontale, vitesse verticale et angle du module. De cette manière, le score est minimal si le module arrive à faible vitesse et bien droit sur la zone, ce qui est notre souhait.

Critère d'arrêt

Le critère d'arrêt consiste en la détection d'un individu dont le profil a permis de poser un module sur la zone d'atterrissage, dans les paramètres définis sur le site CodinGames.

Sélection

Pour la sélection, nous avons opté pour un tournoi à 2 : deux membres de la population sont choisis au hasard, qui vont être comparés et le meilleur est choisi en fonction de son fitness (c'est-à-dire le plus bas, car il s'agit d'un problème de minimisation). On constate ici une intensification, car l'individu ayant le meilleur score sera toujours choisi. Cependant, il existe une certaine incertitude car il n'est pas possible de savoir si les individus choisis au début de la méthode représentent les meilleurs de la population.

Croisement

Tout d'abord, on définit un taux de croisement, qui est comparé à un nombre aléatoire pour déterminer s'il faut appliquer la méthode aux deux parents de la population. Si cette condition est remplie, la partie des parents à sauvegarder pour les enfants de l'autre génération est définie de manière aléatoire et, à partir de là, les autres parties sont remplies selon l'ordre dans lequel les informations apparaissent dans les parents. Ici, il est possible d'interpréter une intensification en combinant les informations de chaque parent, mais en raison des nombres aléatoires présents, il est également possible que les parties qui sont transmises aux générations suivantes n'aident pas l'algorithme à trouver une solution.
Plus globalement, le croisement consiste à échanger une section du profil de commande entre l'un et l'autre, résultant à un échange de trajectoire, mais auxquelles s'ajoute l'inertie différente.

Mutation

En ce qui concerne la mutation, un taux de mutation est fixé pour déterminer quels individus vont muter dans l'algorithme. Il s'agit d'une méthode totalement aléatoire, dans laquelle la fonction aléatoire est utilisée pour définir les limites des informations à modifier et, par conséquent, les nouvelles valeurs que prendra l'individu. Pour que l'algorithme soit plus conforme à la physique du jeu, la variation de la rotation doit être comprise entre -15 et 15 et celle de la puissance entre -1 et 1. Il est important de préciser que la variation ne peut pas dépasser les limites de la rotation (-90 et 90) ou de la puissance (0 et 4), si c'est le cas, la valeur choisie est la limite, qu'elle soit inférieure ou supérieure.

Après les croisement et mutations, une fonction passe sur chaque enfant pour le « lisser » afin d'être sûr qu'il respecte la règle de commandes progressive de ne variant que de +/-1 et +/-15°.

Remplacement

Le remplacement choisi a pour objectif de conserver une partie d'intensification et d'élitisme, sans empêcher de l'exploration et de l'aléatoire. Ainsi, les meilleurs de chacune des populations : ancêtres et enfants, seront choisis pour la prochaine population, et les derniers sont choisis totalement aléatoirement.

Utilisation du code

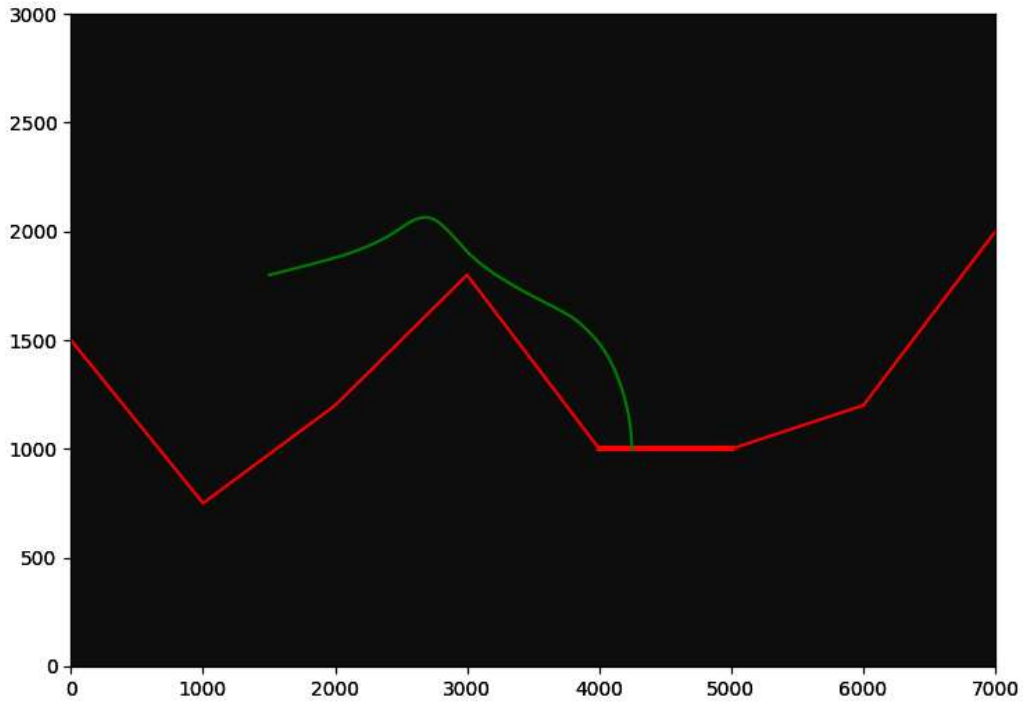
Le code du projet a été conçu en Python et utilise la conception objet. Les fichiers sources du projet représentent 3 classes : Game, Lander, Genetic ; et le dernier fichier « play.py » est celui à lancer afin de réaliser l'apprentissage et de visualiser les résultats.
(Aucun export de l'apprentissage n'a été fait, mais ce dernier est très largement possible via export de la solution optimale en format JSON)
Voici la commande pour lancer le programme : **python3.9 .\src\play.py**

Certaines variables peuvent être modifiées ans le fichier « play.py » pour modifier les paramètres de l'apprentissage :

VARIABLE	FONCTIONNALITÉ
nombrePopulation	nombre d'individu dans une génération
map	choisir la map (map1,map2 ou map3)
nbActions	nombre d'actions prises par un individu
tempsSimualtion	Nombre de tics pour la simulation
echantillonnage	Nb de tics du moteur de jeu par secondes de simulation
initX	Position initiale X
initY	Position initiale Y
tauxCross	Taux de croisement
tauxMut	Taux de mutation

Exemples de résultats

Map 1 :



Map 2 :