Project LifProjet :
AMFR5. Algorithme évolutionniste, simulation physique

IA Apprentissage de conduite

Rapport de projet

Sommaire

I) Introduction :

Intéressés par le domaine des jeux vidéos et de l'intelligence artificielle, nous avons choisi le sujet AFMR5, qui consiste à implémenter un algorithme évolutionniste. Comme son nom l'indique, l'algorithme évolutionniste est basé sur le concepts d'évolution. En accord avec notre professeur encadrant, Mr Meyer, nous avons choisi de modifier un peu le sujet. Nous n'avons pas choisi d'apporter de modifications physiques à nos objets au fil des générations comme cela nous a été présenté au début du semestre. Les modifications se passent ici directement dans la partie « intelligente » de notre objet, en maximisant les connexions qui offrent un meilleur résultats.
Nous avons choisi d'implémenter ici la méthode NEAT, que nous allons détailler dans ce rapport.

II) Explication du problème et méthode de résolution :

Pour simuler notre réseau de neurones, nous avons choisi de partir d'un circuit de voitures, comportant un nombre important de voitures au départ (pour favoriser la diversité de la population). Notre objectif était de faire en sorte que les voitures réussissent à faire un tour complet sans rentrer en contact avec les rebords du circuit. Ce système peut sembler très abstrait, mais adapté à un problème réel peut s'avérer très intéressant ; les voitures autonomes sont en développement et commencent à faire leur apparition, une simulation de la sorte pourrait aider à améliorer la sécurité de ces dernières, en étudiant les erreurs produites par les modèles qui les ont précédé, et même à rattraper des erreurs de conduite humaine (en ce basant sur des données, ou des simulations d'accidents, par exemple).

II.1) Organisation du projet

Tout d'abord, nous avons commencé par nous mettre d'accord sur le langage que nous allions utiliser. Nous savions le python très utilisé dans le domaine de l'intelligence artificielle, mais ayant choisi d'implémenter le réseau de neurones nous-même plutôt que d'utiliser une bibliothèque particulière, nous avons opté pour le python, qui offre une maniabilité remarquable sur ses différents objets, et une syntaxe très épurée, qui nous aura permis de gagner beaucoup de temps, et c'était

également l'occasion de nous familiariser plus précisément avec un langage que nous n'avions pas utilisé depuis la terminale, le langage C++ étant très majoritairement enseigné à la faculté, au dépend des autres. Nous nous sommes appuyés sur la bibliothèque graphique Pygame, qui est une adaptation de la SDL pour python (SDL que nous avons utilisé l'année dernière dans le cadre de l'UE LIFAP4, ce qui nous a permis de gagner également un peu de temps, étant donnée que nous connaissions relativement bien son fonctionnement).
Après quelques heures passées à nous familiariser avec Python et Pygame, il a fallut commencer à se plonger plus précisément dans le sujet. Nous avons donc établi nos objectifs pour le projet :
- modifications des variables par l'utilisateur (pour connaître le niveau d'influence de chacune, et étudier les variations de comportement)

- permettre, après suffisamment de mutations génétiques, aux voitures de faire le tour du circuit sans rencontrer un mur

- posséder une physique relativement proche de la réalité au niveau de la voiture et de ses déplacements

Nous avons donc défini l'architecture de notre projet, pour conserver une structure cohérente et clair dans les fichiers tout au long du projet, toujours dans l'optique de gagner du temps, tout ce qui étant bien fait dès le départ ne nécessitant pas d'être modifié par la suite.

Les choses concrètes arrivaient à grands pas, et nous nous sommes donc réparti dans un premier temps les tâches de la sorte :

Aurélien s'occupait de la partie physique de la voiture, de la gestion des capteurs de la voiture (qui alimenteront le réseau de neurones), et de la partie graphique et Michel s'occupait majoritairement de la partie intelligence artificielle. A noter que cette répartition a été purement indicative ; travaillant toujours ensemble, chacun participait à la construction de la partie de l'autre (Michel intervenait fréquemment quant à la physique de la voiture, et Aurélien a également développer des fonctionnalités telles que le calcul du score de chaque voiture, ou le calcul de différence entre deux individus, par exemple).

Nous n'avons pas choisi la méthode préconisée par nos enseignants pour implémenter notre réseau de neurones, mais avons utiliser l'approche NEAT, en nous appuyant sur une courte série de vidéos disponibles sur Youtube, dont les liens sont en annexe. Ces vidéos sont en anglais et réalisées en langage Java, ce qui nous a également permis de

travailler notre anglais et d'apprendre à nous adapter à des solutions proposées dans des langages que nous ne maîtrisons pas forcément.

II.1) … Et, c'est quoi au juste la méthode NEAT ?

La méthode préconisée par nos enseignants était la suivante : à la fin de chaque générations, on sélectionne les individus les plus performants, que l'on mélange, dans le but de produire des nouveaux individus, obligatoirement performants.

Intéresser par le fonctionnement d'une autre méthode, nous avons donc choisi la méthode NEAT, pour *NeuroEvolution of Augmenting Topologies,* que nous trouvons moins sélective. Cette méthode est plus axée sur l'optimisation des poids entre les neurones, notamment sur les structures les plus simples, pour rajouter des éléments (et donc des nouveaux neurones) de manière efficace.
Au lieu de mélanger automatiquement les individus les plus performants, nous classons à la fin de chaque générations les individus en différentes familles. Cette classification permet l'optimisation d'une caractéristique relativement spécifique à la famille, grâce aux légères modifications apportées à chaque génération.
Contrairement à la méthode présentée en cours, la méthode NEAT permet de réutiliser les génomes moins performants, au lieu de les supprimer directement. En effet, un génome qui n'a pas obtenu un bon score n'a pas forcément tout de mauvais, peut-être que certaines de ses caractéristiques permettront aux individus les plus performants de progresser dans les générations futures.

II.2) Comment se déroule une génération ?

Le concept global est assez simple : on lance le programme avec des génomes identiques, que l'on fait travaillé durant la génération. La génération se termine au moment où toutes les voitures (et donc tous les génomes) sont morts (s'ils ont rencontré un mur ou qu'elles n'ont pas bougé), ou si le temps prédéfini pour la génération est dépassée. A ce moment là, nous calculons le score de chaque voiture en fonction de la distance qu'elle a parcouru. Ensuite, on classe les génomes par famille en fonction des similarités qu'elles présentent (la moyenne des poids de connexions, les connexions en excès et le nombre de connexions disjointes). On mélange ensuite les génomes d'une même famille pour obtenir des génomes fils qui hériterons des caractéristiques de leurs parents, avant de subir de légères mutations, et donc obtenir des génomes potentiellement plus performants. La génération suivante est donc prête et se lance, et le processus se répète.

III) Quels problèmes avons nous rencontré ?

Découvrant le domaine de l'intelligence artificielle, et particulièrement cette méthode, nous avons un peu sous-estimé la complexité quant aux objectifs que nous nous étions fixés initialement. Le problème majeur quant à l'implémentation de ce système, est que nous ne pouvions pas tester le programme dans son ensemble avant « l'assemblage » de l'ensemble des modules principaux. Nous avons donc dû corriger beaucoup de petites erreurs dissimulées dans le programme que les tests unitaires ne nous avaient pas permis de dévoiler.

N'étant que deux dans le groupe, il a été compliqué de remplir tous les objectifs que nous nous étions fixés au départ (par exemple tester si les voitures les plus performantes gardaient le même comportement sur un autre circuit, éviter les collisions entre les voitures, utilisations d'un grand nombre de capteurs, ect...).

Il a également fallu que nous nous adaptions au langage Python, avec lequel nous n'étions pas familiers. Nous lui avons découvert une souplesse extraordinaire, qui nous a permis de gagner beaucoup de temps.

Nous rencontrons également quelques problèmes de performances, nous empêchant, par sécurité, de lancer le programme trop longtemps avec un nombre trop important de voitures, nous limitant à 30 unités, pour garder toutefois un affichage un minimum fluide, et un temps de calcul pour chaque génération le plus faible possible.
puisque nous sommes contrains de nous limiter quant au nombre d'individus, nous avons une plus faible diversité dans nos génomes.

L'optimisation des algorithmes de ce type dépend beaucoup des valeurs des constantes que l'on peut lui assigné, et il a été compliqué de trouver des valeurs optimales pour ces dernière.

IV) Conclusion :

Ce projet nous a permis d'apprendre énormément de choses en programmation : de nouvelles techniques, un nouveau langage et d'acquérir des connaissances dans un domaine que nous ne connaissions pas : l'intelligence artificielle. Un domaine qui est tout aussi intéressant que complexe, mais très enrichissant pour notre futur. Nous avons donc appris à nous adapter à un problème qui nous était totalement étranger, nous incitant à apprendre par nous-même, avec une aide extérieure relativement moindre. Après toutes ces heures passées à travailler sur notre projet, nous n'avons qu'un seul regret, ne pas avoir réussi à implémenter la totalité des fonctionnalités que nous nous étions fixés au départ. N'étant que deux dans le groupe, nous avons dû faire quelques sacrifice, pour pouvoir livrer un projet fonctionnel. Nous remercions évidemment notre encadrant, Mr Meyer, pour sa présence et l'aide qu'il a pu nous apporter, et pour son soutien régulier.
Bien que très enrichissant, nous avons trouvé ce projet très complexe, car il se trouve être relativement éloigné de ce que nous pouvons traiter à notre niveau en licence. L'intelligence artificielle n'est abordée, à la faculté, qu'en Master. La complexité de ce projet nous a permis de progresser, mais nous regrettons une brève introduction à l'intelligence artificielle plus tôt dans le cursus.

V) ANNEXE :

Liens vers les vidéos que nous avons utilisé pour mettre en œuvre le système neuronal :

première vidéo : https://www.youtube.com/watch?v=1I1eG-WLLrY

deuxième vidéo : https://www.youtube.com/watch?v=mep_1VZOepU

troisième vidéo : https://www.youtube.com/watch?v=PXxRjKNX1uw

un article qui nous a également expliquer pas mal de choses concernant la méthode NEAT : http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf