Application Informatique
AlgoGen Jeux

Projet de Licence 3 Informatique

Arnaud Peralta, Yohann Goffart, Louis Pariente
Enseignant référent Carla Selmi

https://github.com/arnaudperalta/algogen-battleship

Sommaire

1 Introduction aux algorithmes génétiques 2

1.1 Définitions 2

1.2 Fonctionnement 3

2 Application des algorithmes génétiques sur la bataille navale 5

2.1 Règles de la bataille navale 5

2.2 Pourquoi avons-nous choisi la bataille navale ? 6

2.3 Design de l'algorithme génétique pour la bataille navale 7

2.4 Résultat recherché 8

3 Fonctionnement du programme 9

3.1 Langage 9

3.2 Structure du programme 9

3.3 Interfaces graphiques 10

4 Résultats 10

5 Webographie 11

1 Introduction aux algorithmes génétiques

1.1 Définitions

Algorithme génétique : Algorithme ayant pour but d'obtenir une solution approchée à un problème d'optimisation (ici un jeu) dans une situation où il n'existe pas de méthode pour le résoudre. Ces algorithmes utilisent la notion de sélection naturelle et l'appliquent sur un ensemble d'individus différents (appelé population).
Lorsqu'un individu semble correspondre à la situation dont il est confronté, on peut interpréter ses gènes comme un algorithme optimisé pour résoudre cette situation.

Sélection naturelle : Mécanisme d'évolution d'une espèce qui traduit le succès reproductif et différentiel des gènes entre des individus d'une même espèce.
Les individus les moins adaptés seront voués à disparaître.

Population : Ensemble d'individus de la même espèce possédant chacun, des caractéristiques différentes (appelées gènes) et issu d'une génération.

Individu : Entité issue d'une population possédant un génome (ensemble de gènes) unique.

Gènes : Caractéristique créé par l'aléatoire, la mutation ou l'accouplement (croisement génétique) de deux individus de la génération précédente.

Génération : Degré d'évolution d'une espèce, plus la génération est avancée plus les individus seront adaptés à la situation à laquelle ils sont confrontés.
Pendant celle-ci, les individus peuvent se croiser afin de créer les individus pour la génération suivante. Dans le cadre d'un algorithme génétique, les croisements sont contrôlés, les meilleurs individus sont conservés (déterminés par une fonction d'évaluation) et des nouveaux individus générés aléatoirement sont introduits.
Une fois que tous les nouveaux individus sont présents pour une nouvelle génération, ils subissent une mutation.

Croisement : Aussi appelé cross-over génétique, action produite entre deux individus afin de créer un individu pour la génération suivante dont ses gènes sont hérités de ses parents.

Mutation : Action effectuée sur tous les individus consistant à modifier légèrement leurs gènes de façon aléatoire.

Fonction d'évaluation : C'est la fonction qui va déterminer si un individu est adapté à la situation dont la fonction d'évaluation dépend. Cette fonction va classer tous les individus afin de garder les meilleurs pour la prochaine génération.

1.2 Fonctionnement

Le déroulement d'un algorithme génétique prend en compte plusieurs paramètres :

- Le nombre de générations à établir.
- Le nombre d'individus que constitue la population.
- Une structure de données représentant les gènes d'un individu (Arbre de décision, réseau neuronal, ...).
- Le pourcentage de la population conservé à la fin d'une génération afin de les croiser génétiquement.
- Valeur représentant la force de la mutation.
- Le quantité de gène qu'un individu possède dans son génome.

Tous ces paramètres influes sur la rapidité de l'algorithme ainsi que sur le résultat obtenu, la complexité est de trouver des réglages justes et adaptés à la structure de données choisie pour le génome.

Par exemple, plus le nombre de générations est grand, plus la population comportera de bons individus mais en conséquent le calcul sera plus long.

La clé de la réussite est de trouver pour chaque paramètre la valeur qui permettra d'obtenir une population avec une moyenne de performance d'individu très bonne en un minimum de calcul.

Un algorithme génétique se présente de la façon suivante :

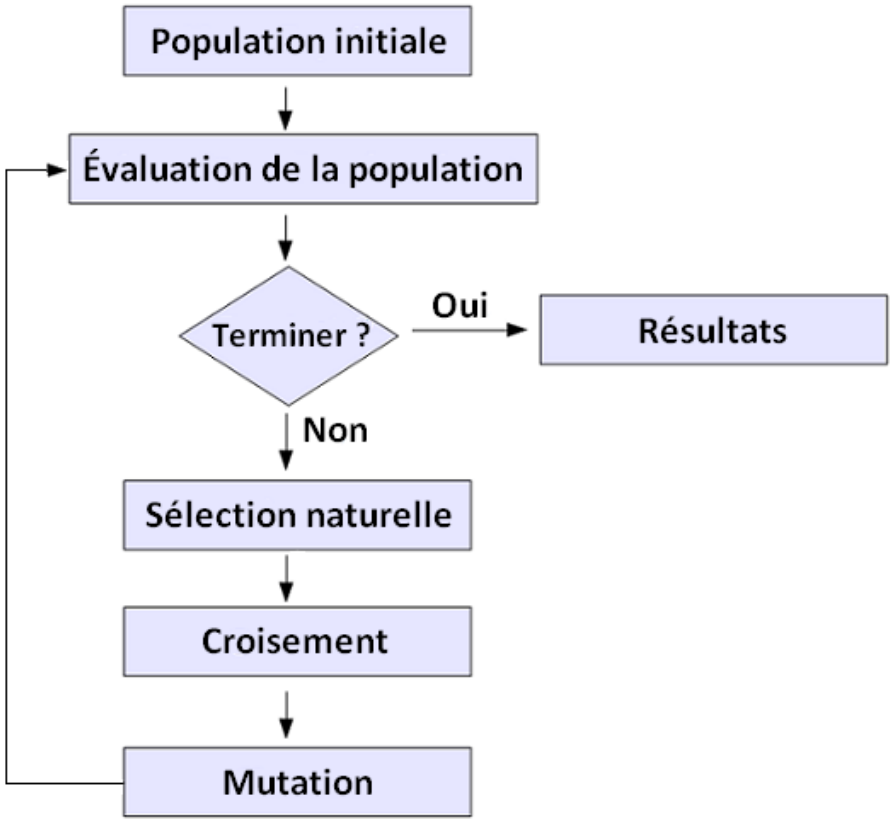

Figure 1: Diagramme décrivant le déroulement général d'un algorithme génétique.

- Les individus constituant la population initiale sont générées de façon totalement aléatoire.

- Une première évaluation des individus est faite, selon les paramètres choisis, l'algorithme choisira de bouclé ou non.

- La boucle consiste à faire une sélection des meilleurs individus selon l'évaluation, puis de croiser les individus retenus pour reconstituer une population de même taille que la précédente. Après cette reconstitution, on fait subir une mutation, puis on boucle à nouveau.

2 Application des algorithmes génétiques sur la bataille navale

Pour réaliser ce projet, nous avons choisi de développer un algorithme génétique sur le jeu de bataille navale, c'est un jeu en un contre un qui nous permettra de jouer contre une intelligence artificielle après avoir calculé plusieurs générations.

2.1 Règles de la bataille navale

La bataille navale est un jeu de société se jouant en un contre un.
Chaque joueur possède deux grilles, une grille occupée par les bateaux du joueur, et une autre grille pour attaquer. Le but du jeu sera de couler les bateaux adverses avant qu'il puisse couler les bateaux nos propres navires.
Le jeu possède deux phases, une phase de placement puis une phase d'attaque :

- La phase de placement consiste à placer un nombre de bateaux sur une grille, les navires peuvent être placés de façon verticale ou horizontale et doivent être placer à l'abri du regard de l'adversaire.
- Une fois les bateaux placés sur les grilles, chacun leur tour, les joueurs devront attaquer grâce à leur grille d'attaque la flotte adverse. Pour chaque attaque le joueur devra transmettre les coordonnées dur tir effectué puis marquer sur sa grille la case où le tir vient d'être effectué.
  Le joueur adverse doit signaler si le tir effectué à toucher ou rater ses bateaux. Si un bateau est entièrement touché, dans ce cas l'adversaire doit signaler que le bateau a coulé.

  Le premier joueur à avoir coulé l'ensemble des navires adverses remporte donc la partie.

Figure 2: Jeu de société de bataille navale.

2.2 Pourquoi avons-nous choisi la bataille navale ?

Le sujet n'imposant pas de jeu pour mettre en place un algorithme génétique, un travail de recherche a été mené afin de trouver le jeu pour ce projet. Pour faciliter la recherche, nous nous sommes poser quelques restrictions pour que l'application de l'algorithme sur ce jeu soit simple et sans une quantité de calculs nécessaire trop grande pour obtenir des résultats convenables.

Le jeu devait être tour par tour et comporter 2 joueurs afin de jouer contre un individu de la dernière génération calculée.
Avoir un nombre de possibilités de décisions faibles pour les structures de données décrivant le comportement de chaque individu ne soit pas volumineuse et que l'algorithme ne soit pas gourmand en temps de calcul.

Notre première réflexion s'est porté sur le jeu du puissance 4.
C'est un jeu en un contre un et tour par tout mais qui a comme défaut d'avoir un nombre de configurations beaucoup trop grand (Quantité de jeu estimé à $1,6 \times 10^{13}$).
La bataille navale possède elle aussi un très grande nombre de situations différentes mais a la grande qualité d'avoir un scénario de jeu récurrent.
Ce que nous entendons par récurrent, c'est sa capacité à faire revenir le joueur à une réflexion de base plusieurs fois :

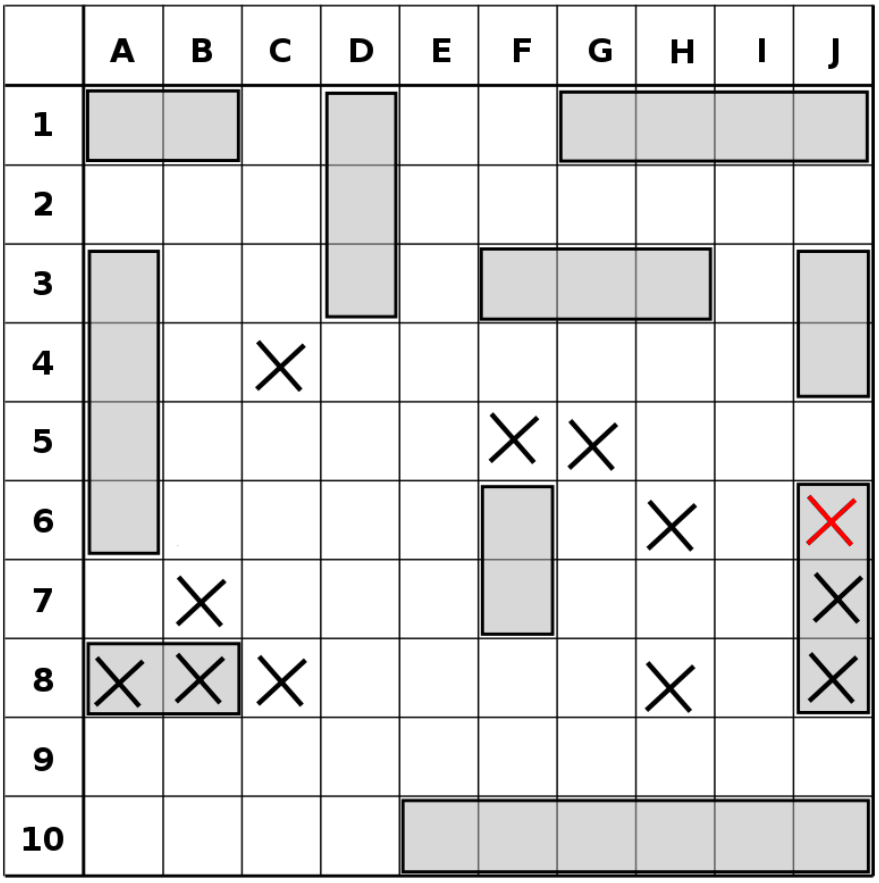

Figure 3: Grille alliée où deux bateaux sont coulés.

Dans la Figure 3 ci-dessus, on observe une situation récurrente, l'adverse vient de jouer sur notre grille en J-6. Il vient donc de couler notre bateau et se retrouve donc dans une phase de recherche où il devra tirer aléatoirement sur notre grille jusqu'à ce qu'il touche un nouveau bateau.
C'est cette phase de recherche qui va permettre de rassembler quasiment toutes les combinaisons de jeu possibles en une seule et nous permettra de nous concentrer sur la phase de destruction d'un bateau touché. Le puissance 4 ne possédait pas cette récurrence de jeu car une partie ne possède pas de situation récurrente puisque chaque coup peut totalement changer la stratégie à adopte. c'est donc pourquoi nous avons choisi la bataille navale.

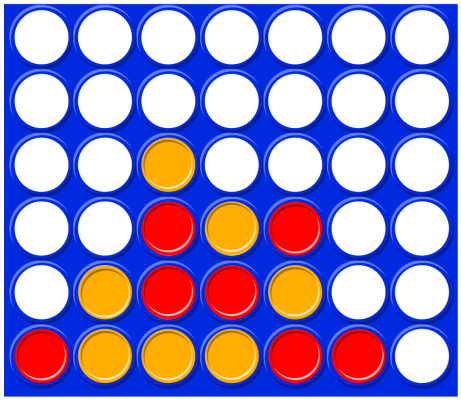

Figure 4: Grille de puissance 4.

2.3 Design de l'algorithme génétique pour la bataille navale

Afin de développer notre algorithme génétique pour un jeu de bataille navale, il a fallu dans un premier temps trouver la bonne structure de données pour nos individus puis une fonction d'évaluation pour réaliser la fitness.
Nous avons fait le choix de représenter une population par un ensemble de séances de tirs sur la même grille, on peux donc déterminer une fonction d'évaluation qui renverrai le nombre de tirs effectués pour couler tous les bateaux de cette grille.
Un individu sera donc une séance de tir avec un nombre boulets à tirer égale à S^2 avec S égale à la taille de la grille, lorsque tous les bateaux placés sur la grille qu'on lui soumet sont coulés, on retiendra le nombre de tir suffisant pour gagner la partie pour cet individu.

Ensuite, pour réaliser le bon choix de structure de données, nous avons du réfléchir à la stratégie que pourrait adopter nos individus durant leurs évolutions. Voici la stratégie type qu'un individu adoptera :

1. Tir aléatoirement sur la grille puis passe à l'étape 2 lorsqu'on touche un bateau.

2. Parcours un arbre de décision sur lequel est inscrit sur chaque nœud de l'arbre un décalage a additionner par rapport au premier tir de base. Si le tir est réussi on parcours la branche de gauche, sinon on parcours celle de droite, on effectue un deuxième tir selon le nouveau décalage du second nœud et ainsi de suite.

3. Une fois le bateau qu'un bateau coule, la stratégie de l'individu recommence à l'étape 1.

Dans cette stratégie l'étape 1 représente l'ensemble de toute les situations récurrente que possède la bataille navale.

2.4 Résultat recherché

En utilisant un arbre de décision et une fonction d'évaluation basée sur le nombre de tir effectué sur une même grille au sein d'une population, nous avons déterminé que la le comportement que pouvait apprendre un individu serait de tirer plusieurs boulets en forme de croix lorsqu'il passe à l'étape 2 de sa stratégie.
C'est en effet la réflexion normal qu'un humain possède lorsqu'il joue à la bataille navale puisque c'est la stratégie la plus efficace lorsqu'un connait une coordonnée d'une pièce de bateau sur la grille.

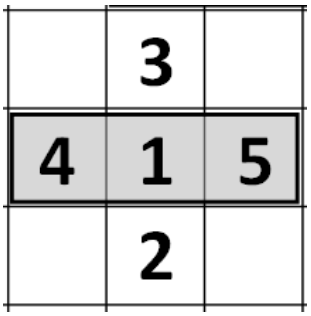

Figure 5: Portion de grille où les tirs sont numérotés.

Sur la figure 5, on observe la stratégie qu'un individu devra développer durant son évolution, lorsqu'il touchera un bateau (ici dessiné en gris) il formera une croix avec ses tirs afin de trouver l'alignement du bateau et de l'éliminer en tirant sur la longueur de celui-ci.
Les branches de droite de l'arbre correspondrait donc à un choix d'orientation, lorsqu'on touche une case du bateau, on emprunterai donc une branche de gauche qui permettra de tirer sur toute la longueur. Ce comportement sera donc créer de façon automatique grâce à l'algorithme génétique.

3 Fonctionnement du programme

3.1 Langage

Pour réaliser ce projet nous avons choisi d'utiliser Python. Ce langage est multi-paradigme et interprété ce qui fait de lui un langage facile d'utilisation, ce qui nous a permis de programmer l'esprit libre afin de nous concentrer uniquement sur l'algorithme qui est le cœur de ce projet.
L'efficacité de ce langage est moins bonne que d'autre langages telles que le C ou le Java en terme de calculs mais nous avons privilégié un langage confortable et muni de librairies standards complètes afin de réaliser un algorithme génétique fonctionnelle.

Sa libraire TKinter nous a permis de réaliser une interface graphique simplement afin de tester complètement les capacités de l'algorithme génétique. Nous avons même pu intégrer un système de graphique dans TKinter grâce à la libraire Matplotlib afin d'afficher les résultats de l'algorithme génétique dans l'interface graphique.

3.2 Structure du programme

Le code source du programme a été divisé en plusieurs partie dans le dossier /src

- Un dossier chart réservé à l'affichage du graphique dans une fenêtre Tkinter à partir des résultats générés par le modèle Core.
- Un dossier core pour le code de l'algortithme génétique et du modèle du jeu de bataille navale. Il présente une classe Game décrivant le comportement d'une partie de bataille navale en utilisant la classe Boat.
- Un dossier graphic où toute la partie interface graphique est stockée, on y trouve une classe singleton App qui lancera une nouvelle fenêtre au début du programme et construira un menu.

Une approche MVC n'a pas peu être établit car TKinter ne possède pas les outils pour développer une architecture telle de façon efficace. D'autres libraires proposent une version MCV de TKinter mais ce n'était pas la philosophie de ce projet.
Nous avons donc un module core qui s'occupe des calculs, un autre qui s'occupe de l'interface graphique et et effectue des appels de fonctions sur ce module core. Le module chart quant à lui est seulement utilisé lorsqu'on souhaite afficher les résultats des calculs.
Hormis les dossiers organisant les fonctionnalités du programme, un fichier source main est placé à la racine du dossier /src pour instancier la classe Core ainsi que la création de la fenêtre et du menu.

Le dossier /cfg est utilisé pour stocker le fichier de configuration où sont inscrits les paramètres du programme.

3.3 Interfaces graphiques

Le menu se présente comme ci-dessous : On peut accéder aux différents modules du programme, il y a partout un sens logique à respecter pour bien utiliser le programme.

En ce qui concerne les statistiques il est préférable de lancer un entrainement complet avant de visualiser la courbe de progression de l'algorithme, cette courbe détermine le nombre de tirs moyen en fonction du numéro de la génération.
En revanche, il est possible de jouer contre l'IA (un individu extrait de la dernière génération) même si aucun entrainement a été effectué, cet IA jouera purement aléatoirement et présentera un faible niveau de jeu.

4 Résultats

Après plusieurs essais nous avons déterminé cette liste de paramètre comme étant la meilleur pour notre programme :

5 Webographie