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

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.

Gène : caractéristique créée 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 de 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 : 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.
- Une quantité de gènes qu'un individu possède dans son génome.

Tous ces paramètres influent 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 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 calculs.

Un algorithme génétique se présente de la façon suivante :

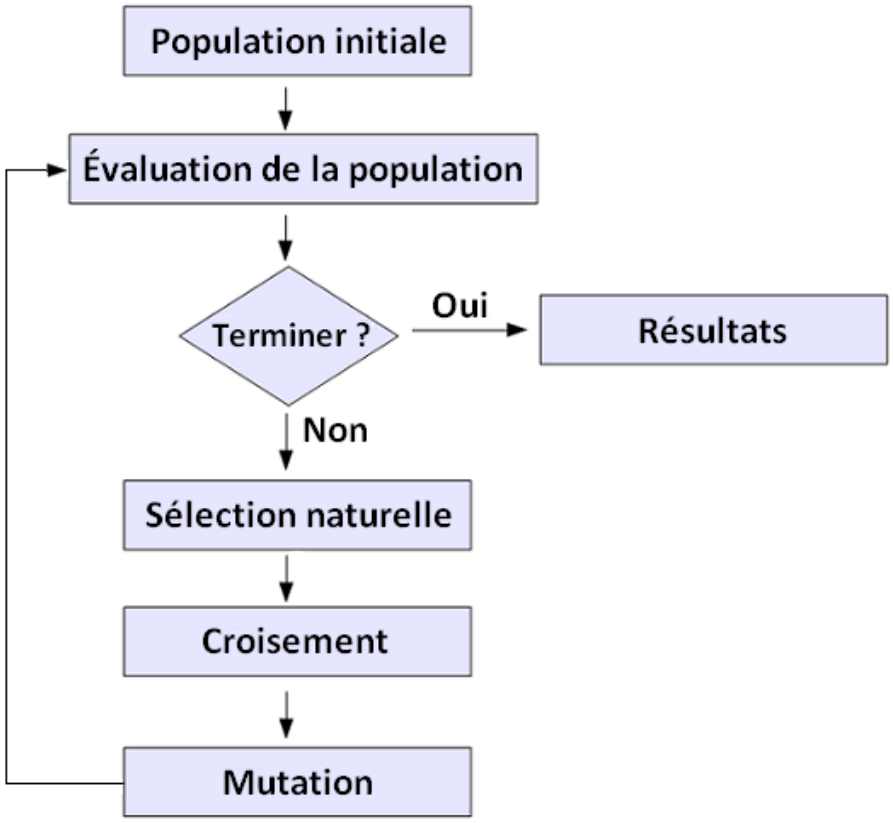

Figure 1: Diagramme décrivant le déroulement général d'un algorithme génétique.

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

- Une première évaluation des individus est faite, selon les paramètres choisis. L'algorithme choisira de boucler 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 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 doivent être placés de façon verticale ou horizontale, 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 du 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 a touché ou raté 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 posé quelques restrictions pour que l'application de l'algorithme sur ce jeu soit simple et sans une quantité de calculs nécessaires 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.
Il devait avoir un nombre de possibilités de décisions faibles pour que 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 portée sur le jeu du puissance 4.
C'est un jeu en "un contre un" et "tour par tour", 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 grand nombre de situations différentes, mais possède 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 :

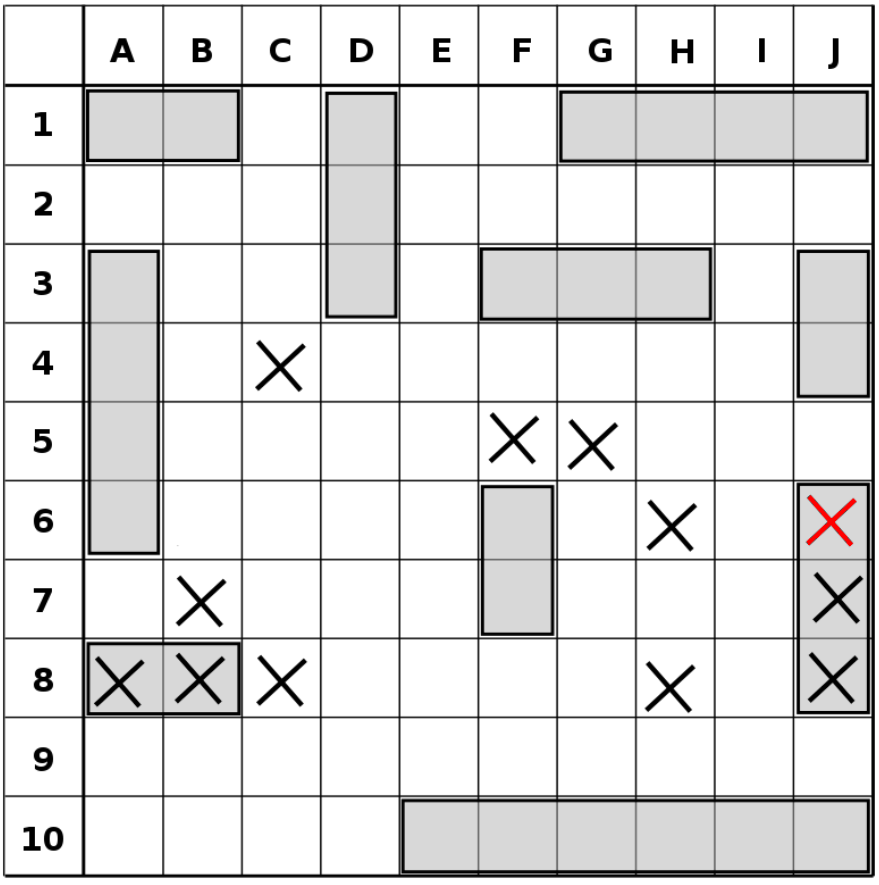

Figure 3: Grille alliée où deux bateaux sont coulés.

Dans la Figure 3 ci-dessus, on observe une situation récurrente. L'adversaire 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 adopter. C'est pourquoi nous avons choisi la bataille navale.

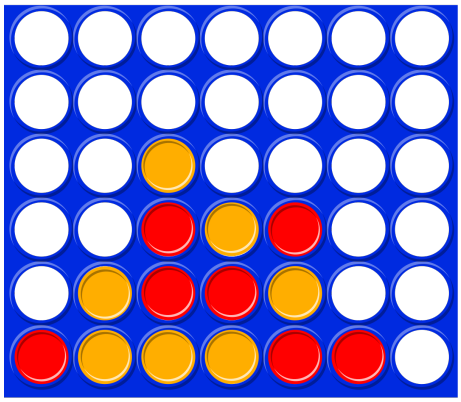

Figure 4: Grille de puissance 4.

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 peut donc déterminer une fonction d'évaluation qui renverrait le nombre de tirs effectués pour couler tous les bateaux de cette grille.
Un individu sera donc une séance de tirs, 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 tirs suffisants pour gagner la partie pour cet individu.

Ensuite, pour réaliser le bon choix de structure de données, nous avons dû 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 à additionner par rapport au premier tir de base. Si le tir est réussi, on parcourt la branche de gauche, sinon on parcourt celle de droite. On effectue un deuxième tir selon le nouveau décalage du second nœud et ainsi de suite.
3. Une fois un bateau coulé, la stratégie de l'individu recommence à l'étape 1.

Dans cette stratégie, l'étape 1 représente l'ensemble de toutes les situations récurrentes 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 tirs effectués sur une même grille au sein d'une population, nous avons déterminé que 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 lorsque nous connaissons une coordonnée d'une pièce de bateau sur la grille.

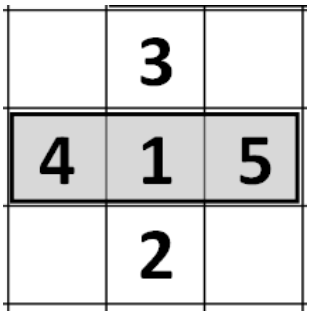

Figure 5: Portion de grille où les tirs sont numérotés.

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 correspondraient donc à un choix d'orientation. Lorsqu'on touche une case du bateau, on emprunterait donc une branche de gauche qui permettrait 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. Cela nous a permis de nous concentrer uniquement sur l'algorithme, qui est le cœur de ce projet. L'efficacité de ce langage est moins bonne que celle d'autres langages, tels que le C ou le Java en terme de calculs, mais nous avons préféré un langage confortable, muni de librairies standards complètes permettant de réaliser un algorithme génétique fonctionnelle.

Sa librairie TKinter nous a permis de réaliser simplement une interface graphique, pour tester complètement les capacités de l'algorithme génétique. Nous avons même pu intégrer un système de graphique dans TKinter, grâce à la libraire Matplotlib, permettant 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 parties 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 pu être retenue, car TKinter ne possède pas les outils pour développer une telle architecture, de façon efficace. D'autres libraires proposent une version MCV de TKinter, mais ne correspondait pas à la philosophie de ce projet. Nous avons donc un module core qui est chargé des calculs, un autre qui s'occupe de l'interface graphique 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 est inscrit les paramètres du programme.

3.3 Interfaces graphiques

Le menu se présente comme ci-dessous :

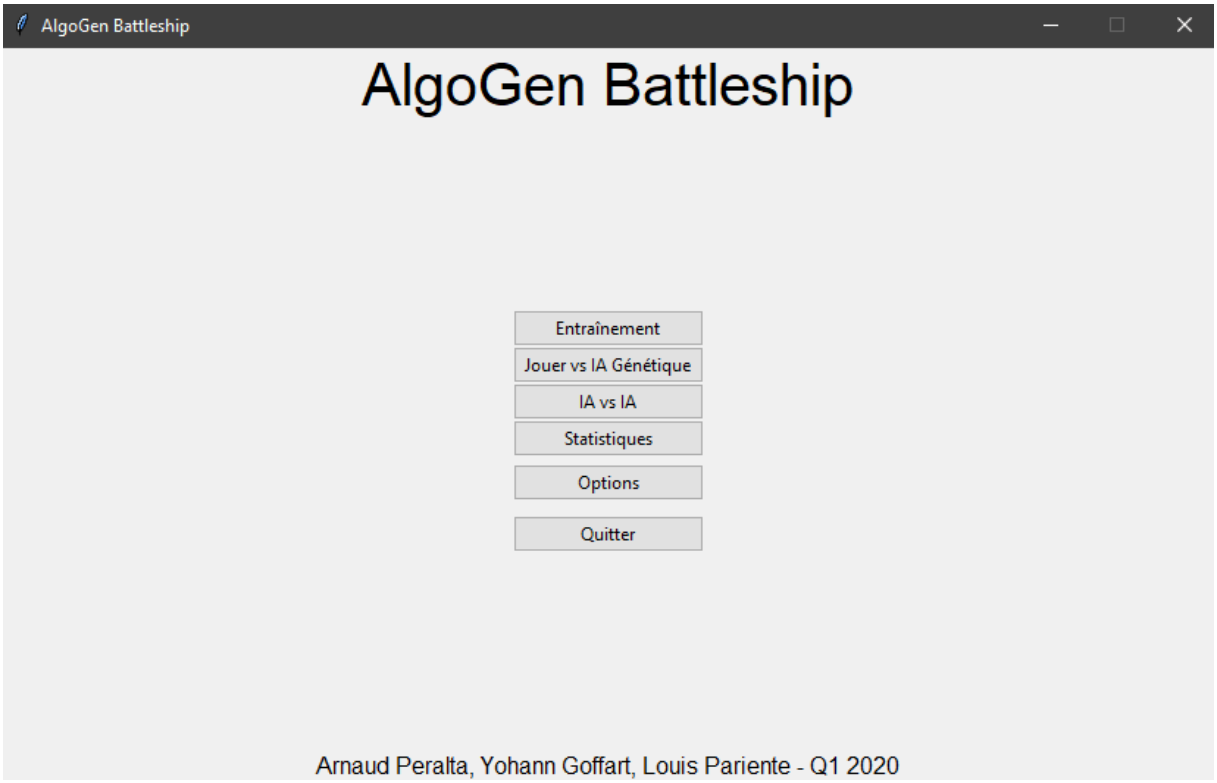

Figure 6: Menu du programme.

On peut accéder aux différents modules du programme, mais il y a 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 n'a été effectué, cette IA jouera purement aléatoirement et présentera un faible niveau de jeu.

Entraînement

La section Entraînement présente trois options :

- un entraînement pas à pas qui permet l'entrainement sur une génération de la population actuelle (On obtiendra donc une population de génération actuelle + 1).
- un entrainement complet qui entraine la population jusqu'à la génération maximum déterminée dans les paramètres.
- une ré-initialisation de la population actuelle, ce qui est essentielle pour entraîner une nouvelle génération basée sur de nouveaux paramètres.

Jouer vs IA

Cette section du programme permet de jouer contre l'ordinateur au jeu de la bataille navale, sur un plateau composé de deux grilles. La grille de gauche représente celle du joueur humain, et celle de droite représente la grille d'attaque sur le plateau de l'IA.
L'IA sera extraite de la population entrainée grâce à l'algorithme génétique développé dans le programme.

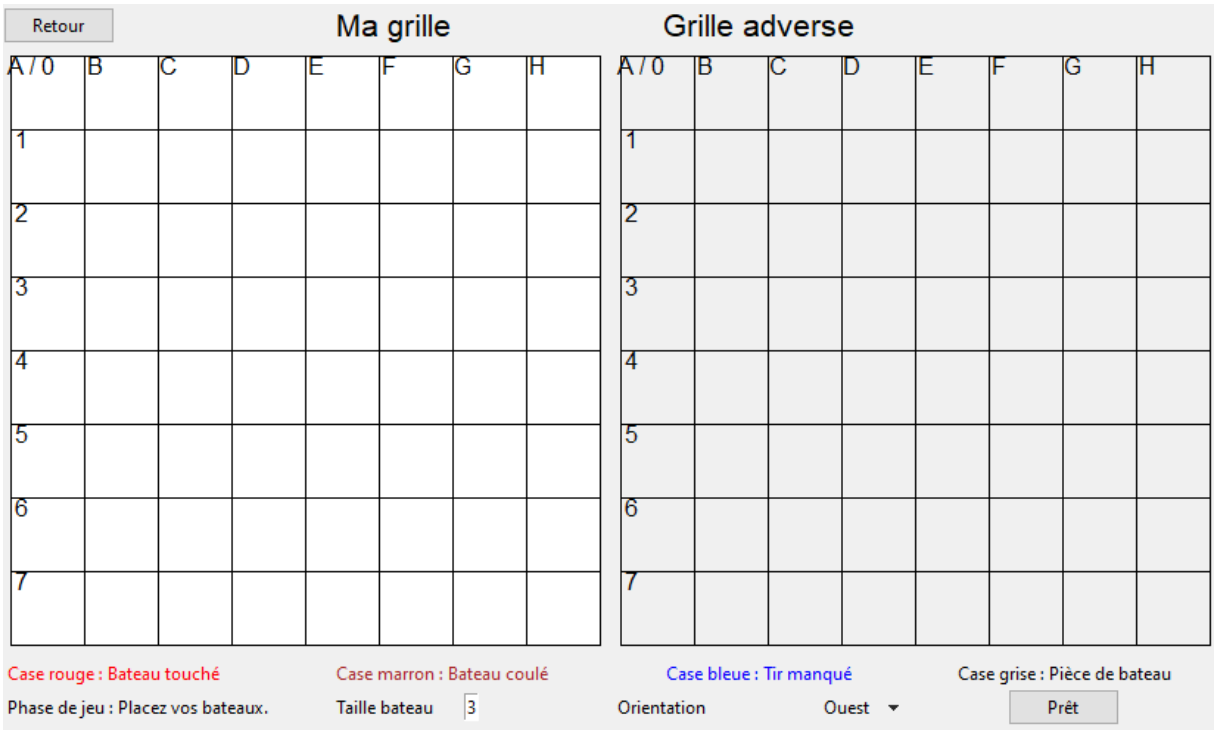

Figure 7: Plateau de jeu du programme.

La partie commence par une phase de placement des bateaux, ceux de l'ordinateur sont déjà placés mais invisible pour le joueur. Il faut choisir la taille et l'orientation du bateau à placer avant de cliquer sur la grille pour placer le bateau. Une fois le bon nombre de bateaux placé, on peut cliquer sur le bouton "Prêt" afin de commencer la partie.
Un code couleur pour les cases a été établi comme suivant :

- Rouge pour un bateau touché
- Marron pour un bateau coulé
- Bleu pour un tir raté
- Gris pour une pièce de bateau, invisible dans le cas d'une grille ennemie.

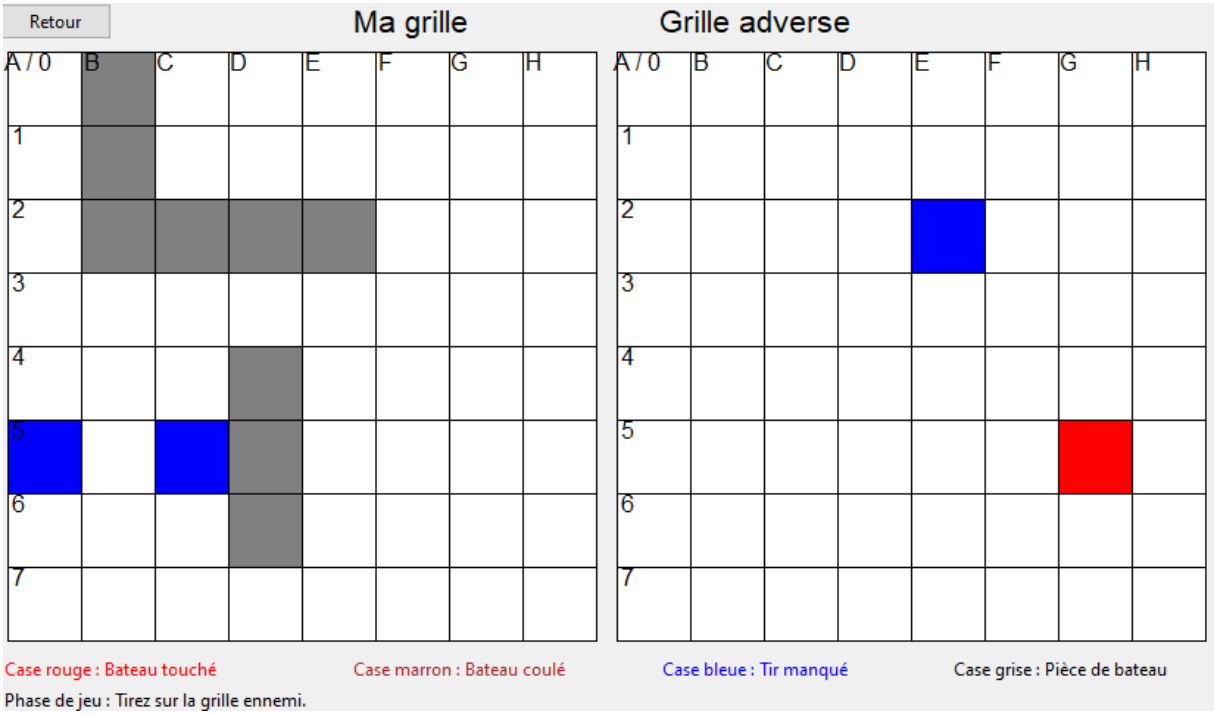

Figure 8: Plateau de jeu du programme.

Sur la grille ci-dessus, un premier coup a été joué en E2 puis l'IA a tiré en A5. Ces deux coups sont ratés puisque les cases sont devenues bleues. Le deuxième tir en G5 est un tir sur un bateau ennemi, puisque la case est devenue rouge.

IA vs IA

Le même plateau que la section précédente est utilisé dans cette partie du programme, excepté que l'utilisateur n'intervient pas dans la partie. Le bouton "Prêt" existe tout de même afin de lancer la partie entre les deux IA. Ces IA sont extraites de la dernière génération calculée par l'algorithme génétique. Le joueur de gauche correspond à l'individu ayant obtenu la meilleur évaluation lors de la dernière génération, le joueur de droite étant le second.

Si on démarre une partie avec une population très peu entrainée, le résultat pourrait ressembler à la Figure 9.

On y observe deux grilles avec beaucoup de tirs effectués, et donc deux mauvaises performances de la part des IA. Cela est causé par le fait que ces deux IA ne sont pas entraînées, et que leurs décisions sont purement aléatoires.

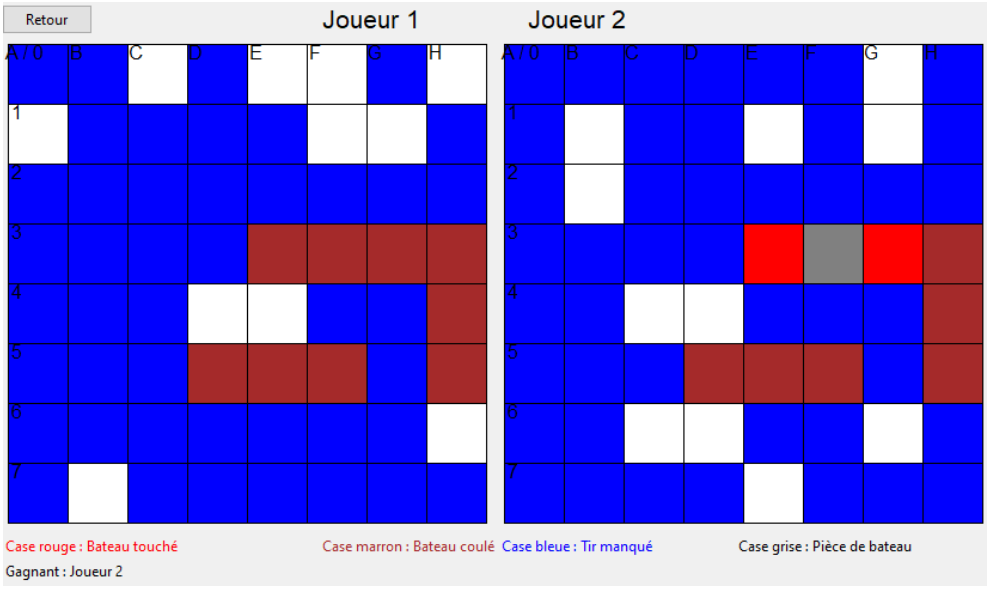

Figure 9: Plateau de jeu du programme avec deux IA non entraînés.

Dans le cas où un entrainement conséquent est réalisé dans le programme, on pourrait observer une partie où très peu de tirs sont effectués, car les IA possèderont une stratégie performante comme le montre la Figure 10 :

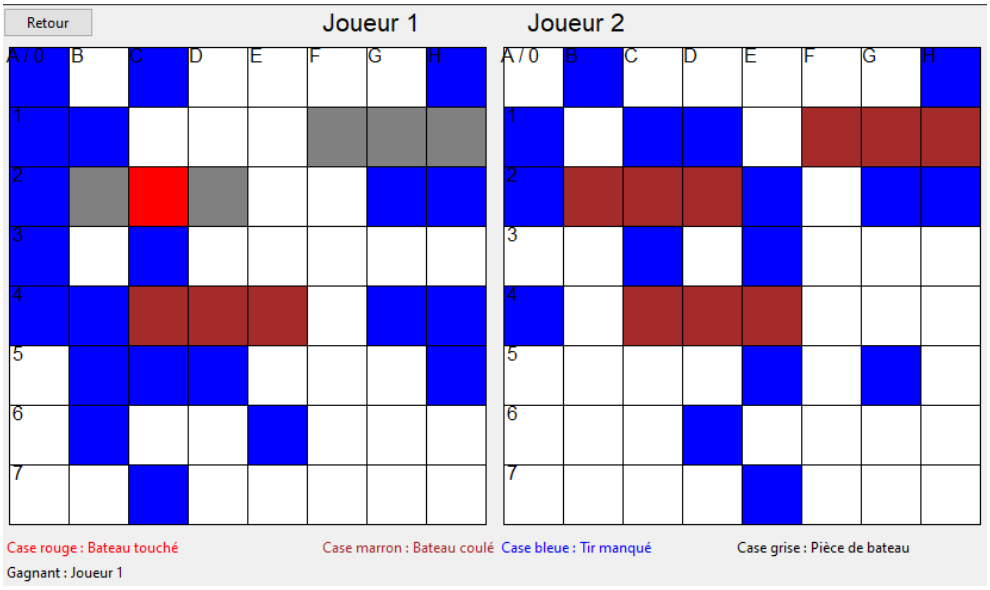

Figure 10: Plateau de jeu du programme avec deux IA entraînées.

Statistiques

Cette section du programme permet d'observer les résultats de la fonction d'évaluation (fitness) sur chaque génération. Si des options correctes et un nombre de générations suffisant sont entrées dans le programme, une courbe décroissante devrait être obtenue, ce qui indiquerait une amélioration de la population au fur et à mesure des générations.

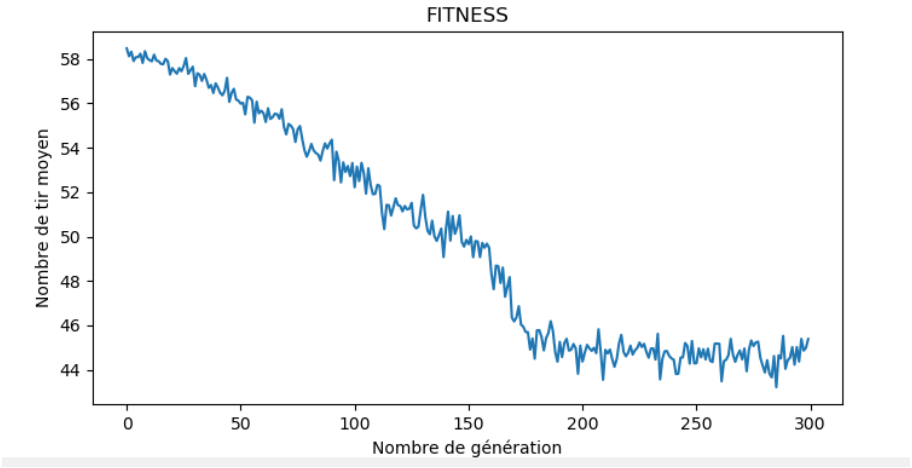

Figure 11: Graphique résumant l'entraînement sur 300 générations selon le nombre de tir moyen par individu sur une grille aléatoire.

Options

Le menu d'options présente les différents paramètres que comporte un algorithme génétique, comme décrit dans la section 1.2 du document, ainsi que les paramètres du jeu de bataille navale.

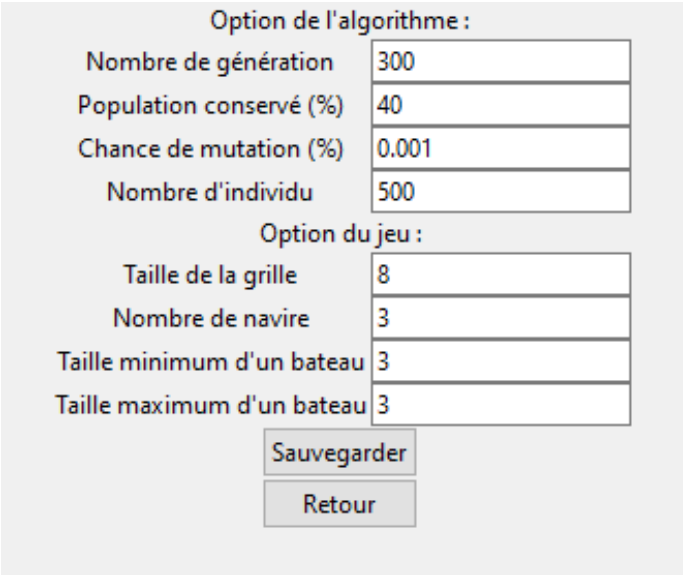

Figure 12: Menu d'options.

Les paramètres sauvegardés via ce menu, sont stockés dans .JSON dans le dossier cfg/ du projet. Ils seront chargés lors de la prochaine ouverture du programme.

4 Développement de la collaboration

Ce projet a nécessité une phase de recherche scientifique et de développement informatique, qui a représenté une charge de travail conséquente. Le résultat obtenu n'aurait jamais pu être atteint sans une bonne manipulation d'outils collaboratifs et d'une communication régulière des membres du groupe de projet.
Nous avons utilisé deux outils collaboratifs pour ce projet :

4.1 Notion.so

Pour s'assurer une productivité optimale et éviter l'empiètement mutuel qu'un projet de développement informatique pourrait avoir, nous nous sommes réparti, en plusieurs taches, les phases de recherche et développement du projet.
Notion.so est une plateforme se présentant comme un bloc note en ligne synchronisé, pour tous les membre d'un groupe. Des templates sont mis à disposition afin de réaliser un workspace complet.

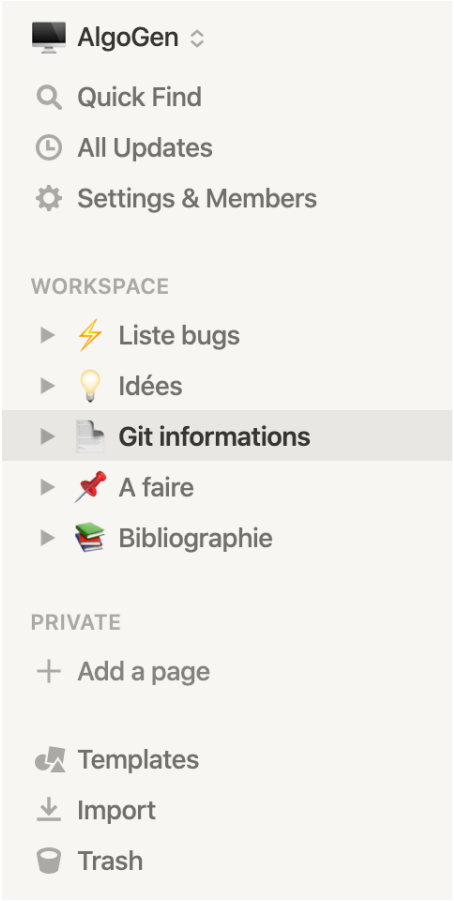

Figure 13: Les sections où des informations relatives au projet sont listées.

La section "A faire" fut la plus utile, car c'est ici que les taches ont été attribuées aux membres du groupe. La répartition des tâches peut être résumée de cette façon :

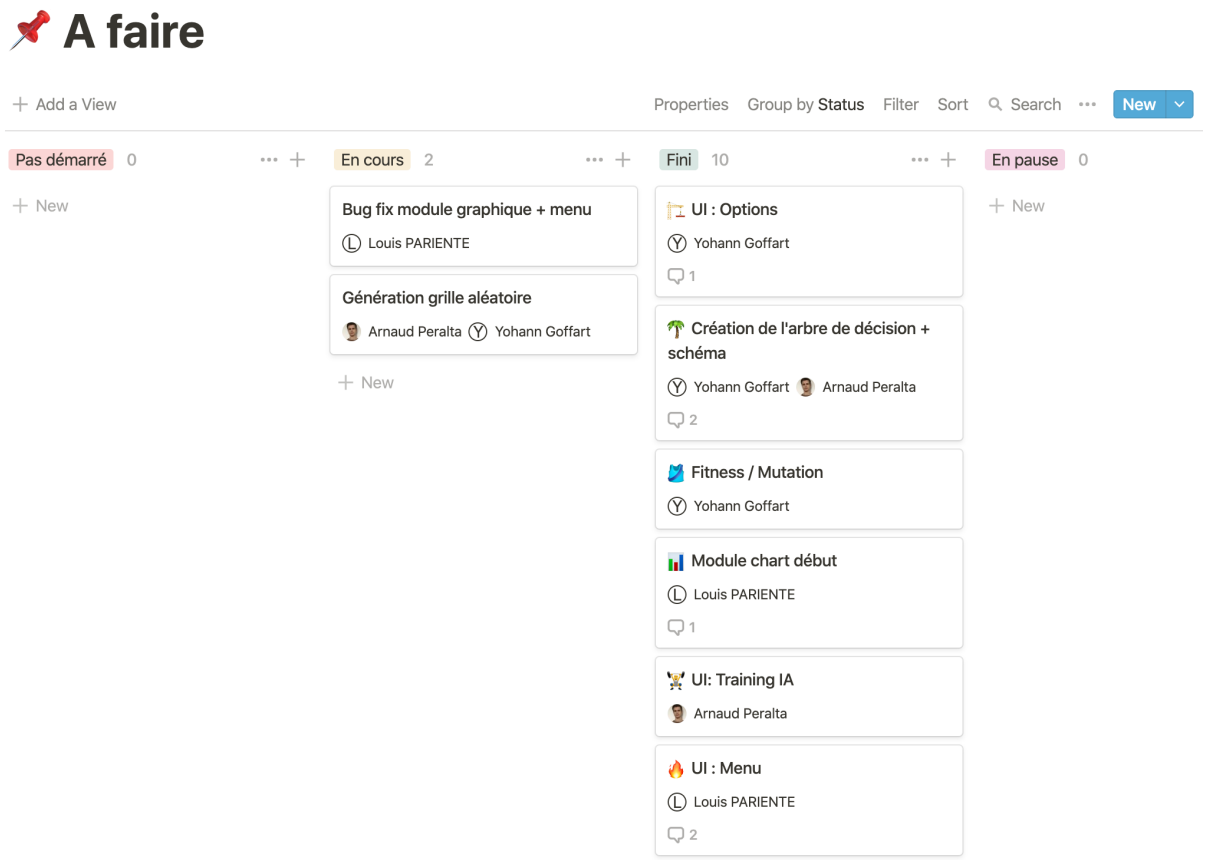

Figure 14: Répartition des taches à faire entre les membres du groupe.

- Yohann Goffart : Menu Options, classe Population, classe Individu, fonction d'évaluation.
- Louis Pariente : Menu Statistiques, menu principal, IA vs IA.
- Arnaud Peralta : Plateau de jeu, joueur vs IA, raccordement des modules, classe Core, classe Game

La recherche scientifique préalable au développement de ce projet a été réalisée par l'ensemble des membres du groupe. Les résultats ont été stockés dans une section du workspace Notion, pour que chaque membre puisse y mettre la main quand il le souhaitait.

4.2 Git

Une tache une fois attribuée, doit être réalisée par les personnes associées à celle-ci. Plusieurs taches simultanées sont attribuées au sein du projet, c'est pourquoi Git nous a été utile. Nous avons pu travailler chacun de notre coté, sans pour autant attendre qu'un partenaire puisse finir sa modification. Afin d'avoir un versionnage possédant une interface graphique pour revoir une ancienne version de notre code, ou bien observer les dernières modifications effectuées par exemple, nous avons utilisé la plateforme GitHub pour héberger notre dépôt Git.

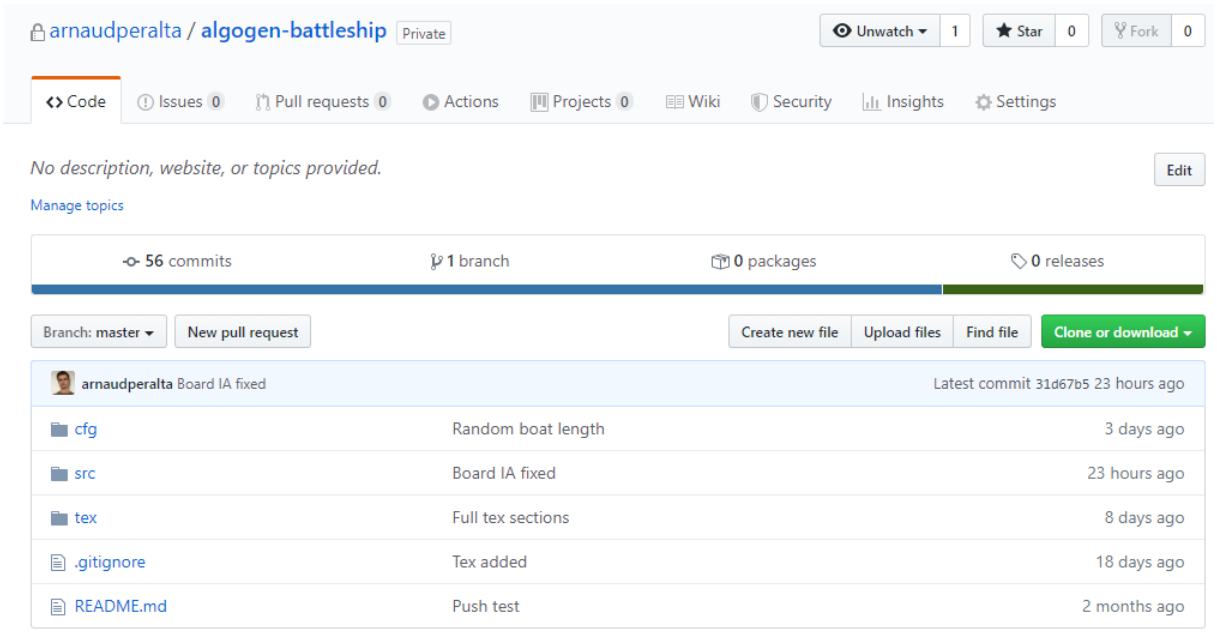

Figure 15: Dépôt GitHub avec l'arborescence du projet.

Ce dépôt est disponible à l'adresse suivante :

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

5 Webographie

Ce mémoire a été le premier quartile de 2020, cette webographie a été réalisée le 17 mars 2020.

Compréhension générale de la problématique :
http://www-igm.univ-mlv.fr/ dr/XPOSE2013/tleroux_genetic_algorithm/hello-world.html
http://helios.mi.parisdescartes.fr/ bouzy/Doc/AA1/AlgoGenetiques.pdf

Définitions :
https://fr.wikipedia.org/wiki/Algorithme_génétique

Règles officielles de la bataille navale :
https://fr.wikipedia.org/wiki/Bataille_navale_(jeu)

Quantité de jeu du puissance 4 :
http://villemin.gerard.free.fr/aJeux1/Societe/Puiss4.htm

Photo du jeu de société de bataille navale :
https://images-na.ssl-images-amazon.com/images/I/41MRNQU0cvL._AC_SX425_.jpg