Projet 1 : Bataille navale
LU3IN005

1 Modélisation et fonctions simples

Pour la implémentation des fonctions de base du jeu de bataille navale, nous avons choisi d'utiliser les NumPy Arrays au lieu des listes Python car l'utilisation de NumPy Arrays entraîne une consommation de mémoire plus faible et une durée d'exécution plus courte. Une autre raison d'utiliser les NumPy Arrays était leur compatibilité avec les bibliothèques d'analyse statistique et de visualisation que nous avons utilisées pour analyser nos données.

2 Combinatoire du jeu

2.1

Une borne supérieure simple du nombre de configurations possibles pour la liste complète de bateaux sur une grille de taille 10×10 est le produit des nombres de façons de placer chaque bateau donné sur la même grille.

$$180 \times 160 \times 160 \times 140 \times 120 = 77414400000 \tag{1}$$

2.2

Les résultats de notre fonction qui permet de calculer le nombre de façons de placer un bateau donné sur une grille vide correspond aux résultats théoriques. Soit N nombre de façons de placer un bateau donné sur une grille vide de taille $n \times n$ et t la taille de bateau:

$$N = (n - t + 1) \times n \times 2 \tag{2}$$

2.3

L'approche *Brute-Force* de cet algorithme signifie que nous aurions besoin de générer un maximum de 77414400000 grilles, ce qui prendrait énormément de temps.

2.4

En considérant toutes les grilles équiprobables et E l'événement de tirer une grille donnée (nous examinerons plus tard la valeur approximative donnée ici):

$$P(E) = \frac{1}{N} \approx \frac{1}{30095060976} \tag{3}$$

2.5

Nous avons implémenté un algorithme qui génère aléatoirement une grille à partir d'une liste de bateaux et qui compte le nombre de grilles différentes qui ont été générées. Cet algorithme, dans le pire des cas, devrait générer jusqu'à environ 30095060976 grilles, ce qui représente une complexité temporelle trop importante.

2.6

Nous avons implémenté un algorithme qui utilise un processus aléatoire pour approcher le nombre total de grilles possibles. Pour comprendre l'algorithme, essayons d'abord de visualiser le problème. Imaginons un arbre avec une racine $b0$ qui a autant de branches qu'il y a de façons de placer le premier bateau. Aux extrémités de ces branches, nous avons les nœuds $b1_0, b1_1, b1_2, ..., b1_n$ et pour chaque nœud $b1_i$, nous dessinons autant de branches qu'il y a de placements possibles du bateau $b2$ en sachant que $b1_i$ est déjà placé dans une position unique dans la grille. Ensuite, pour chaque nœud de $b1_0, b1_1, b1_2, ..., b1_n$, nous dessinons d'autres branches qui correspondent aux possibilités de placement du deuxième bateau $b2$, sachant que $b0$ et $b1_i$ sont déjà placés, etc. Nous obtenons un arbre avec un nombre de feuilles N qui correspond à un nombre total de combinaisons possibles de la grille. Comme nous avons déjà compris dans les questions précédentes que nous ne pouvons pas revenir en arrière dans l'arbre et calculer le nombre exact de feuilles, nous utilisons l'algorithme suivant pour obtenir une valeur N approximative :

1. Initialiser N à 1 qui représente le nombre de nœuds au niveau 0 de l'arbre (uniquement la racine).
2. Finir si atteint les feuilles de l'arbre.
3. Sinon, choisir au hasard n_i un des nœuds du niveau actuel de l'arbre, soit m_i le nombre de branches de n_i.
4. Supposer que toutes les valeurs à ce niveau de l'arbre aient m_i branches et actualiser $N = N \times m_i$ le nombre de nœuds du niveau suivant de l'arbre.

Cet algorithme se termine parce qu'il finit par atteindre les feuilles d'un arbre. Il est équiprobable de générer n'importe quel combinaison de nœuds qui signifie que la variable aléatoire X (le nombre de placements possibles dans une grille générée de façon aléatoire) est équiprobable. Selon la loi forte des grands nombres, la moyenne empirique est un estimateur fortement convergent de l'espérance : $\lim_{n\to\infty} \frac{1}{n}(X_1 + X_2 + ... + X_n) = E(X)$. Alors l'algorithme est valide et converge vers le nombre total de grilles possibles.

Dans notre expérience avec cet algorithme, nous avons généré $10, 20, ...1000000$ grilles contenant la liste complète des bateaux cinq fois et nous l'avons reportée sur un graphique en log-lin (figure 1). Nous avons trouvé que la moyenne de 5 millions de grilles générées au hasard était de $\mu = 30095060976$ *(ce qui explique l'approximation que nous avons utilisée dans la question 2.4 et 2.5)* et $\sigma = 4321383125.000705$ où $\frac{\sigma}{\mu} \approx 0.14$). La figure 2 est un graphique d'erreur log-log qui montre l'échelle $\frac{1}{\sqrt{N}}$ d'erreur avec le nombre de grilles N générées aléatoirement. C'est une représentation visuelle de l'écart-type des moyennes pour $10, 20, 50, ..., 1000000$ des grilles générées avec un ajustement de la courbe $f(x) = a \times x^b$ où $a = 4.37522903 \times 10^9$ et $b = -4.97741272 \times 10^{-1}$ qui correspond à la loi d'échelle $\frac{1}{\sqrt{N}}$.

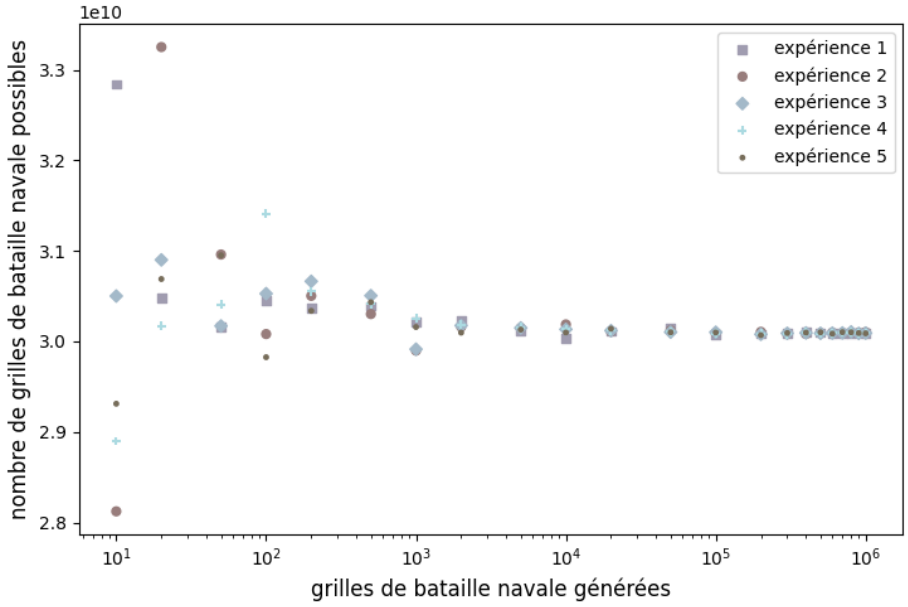


Figure 1: log-lin

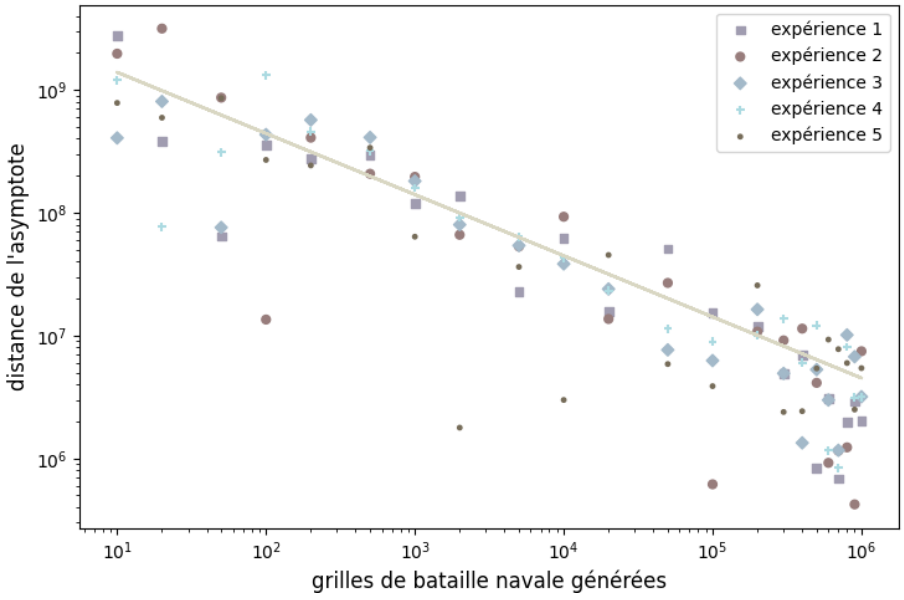


Figure 2: log-log

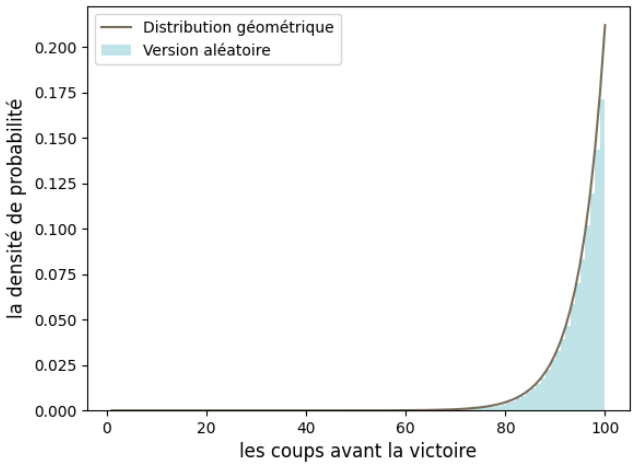
3 Modélisation probabiliste du jeu

3.1 Version aléatoire, naïve et heuristique

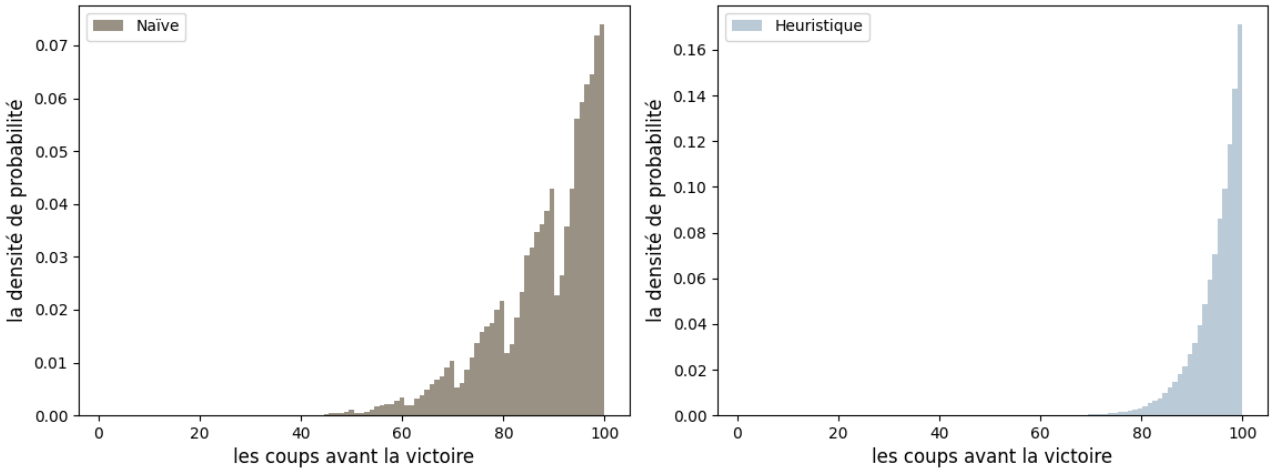
Pour décrire formellement la distribution de probabilité de la variable aléatoire correspondant au nombre de coups pour terminer une partie, on cherche une loi de probabilité discrète du nombre X d'événements indépendantes (hypothèse) de probabilité de premier succès $p \in]0, 1[$. X est la variable aléatoire donnant le nombre de coups $k \in 1, 2, ..., 100$ avant la première victoire de jeux. En fixant

$$P(X = k) = p \times (1 - p)^{(100-k)-1}, p = \frac{17}{100} = 0.17. \tag{4}$$

et en traçant la distribution géométrique à côté de l'histogramme obtenu avec 100000 jeux, on obtient:



Tout en mettant en place le joueur aléatoire, nous avons également implémenté une version naïve qui vérifie les cases de la grille ligne par ligne. En plus d'être plus efficace que le joueur aléatoire, elle a une distribution de probabilité intéressante avec de faibles valeurs vers n coups où $n \mod 10 < 5$. C'est parce qu'environ la moitié des bateaux sont placés horizontalement et que la plupart ont une taille de 3 à 5 cases, ce qui fait que notre joueur naïf finit rarement la partie au début de la ligne. Le joueur heuristique vérifie la ligne et la colonne d'une case où un bateau a été détecté et a une distribution de probabilité similaire à la version aléatoire. La performance de ces deux joueurs est représentée dans les histogrammes suivants (100 000 jeux).



3.2 Version probabiliste simplifiée

Soit B_i l'événement qu'une case contient le bateau b_i. Pour calculer la nouvelle distribution jointe sur tous les bateaux on devrait calculer $P(B_i \cup B_{i+1}) = P(B_i) + P(B_{i+1}) - P(B_i \cap B_{i+1})$ et $P(B_i \cap B_{i+1}) = P(B_i) \times P(B_{i+1}|B_i)$ où pour calculer $P(B_{i+1}|B_i)$ on a besoin d'information sur $P(B_i \cap B_{i+1})$ que nous n'avons pas. Alors on ne peut pas calculer la distribution jointe et on fait une simplification en considèrent chaque bateau de manière indépendante ce qui signifie que $P(B_i \cap B_{i+1}) = P(B_i) \times P(B_{i+1})$. À chaque tour, pour chaque case de la grille on calcule:

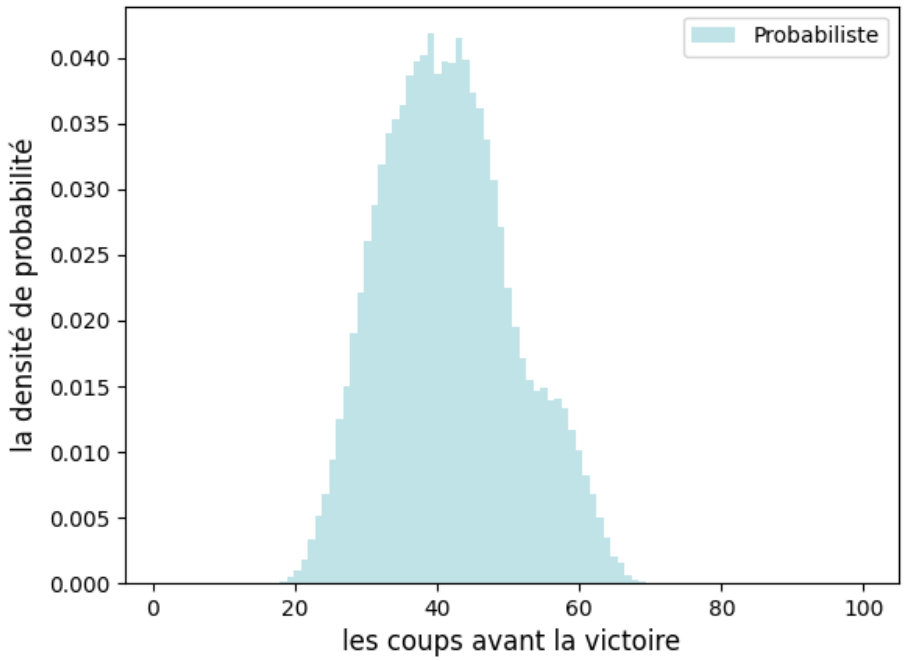
$P(\text{case contient un bateau}) \iff P(b_0 \text{ l'occupe} \cup b_1 \text{ l'occupe} \cup ... \cup b_n \text{ l'occupe})$

$\iff 1 - P(b_0 \text{ ne l'occupe pas} \cap b_1 \text{ ne l'occupe pas} \cap ... \cap b_n \text{ ne l'occupe pas})$

$\iff 1 - P(b_0 \text{ ne l'occupe pas}) \times P(b_1 \text{ ne l'occupe pas}) \times \ ... \ \times P(b_n \text{ ne l'occupe pas})$

$\iff 1 - (1 - P(b_0 \text{ l'occupe})) \times (1 - P(b_1 \text{ l'occupe})) \times \ ... \ \times (1 - P(b_n \text{ l'occupe}))$

La performance de la version probabiliste est donnée dans l'histogramme ci-dessous (100 000 jeux).

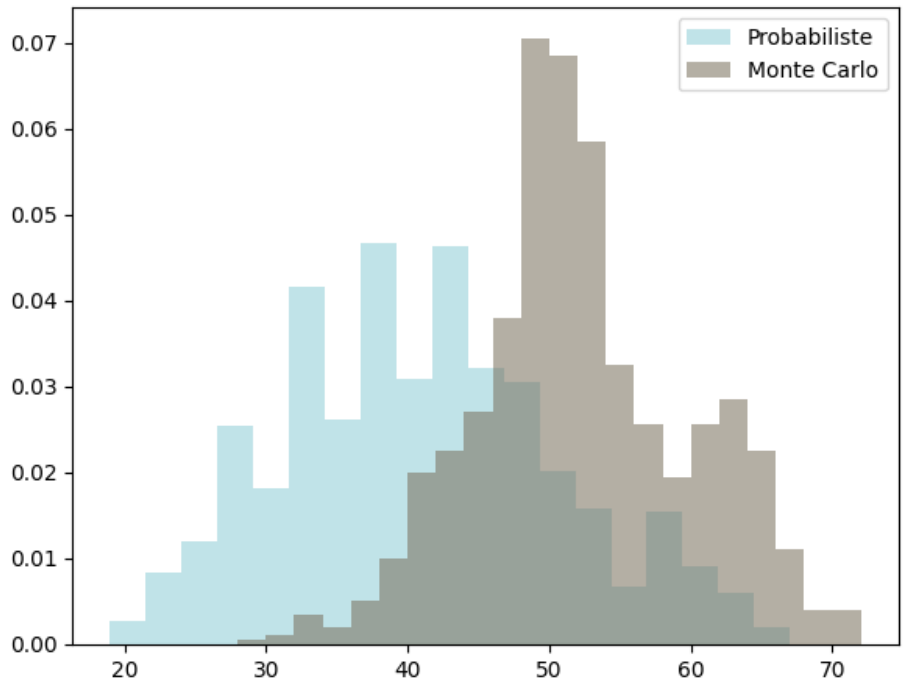


3.3 Version Monte-Carlo

Cet algorithme repose sur un échantillonnage aléatoire répété pour obtenir une matrice de probabilité et tirer la case la plus probable. Les étapes de l'algorithme sont les suivantes:

1. Prend la grille courante en entrée.
2. Crée une copie de la grille.
3. Simule un nombre i d'échantillons (ici $i = 1000$), dont chacun est un grille avec un placement aléatoire des bateaux restantes.
4. Regroupe toutes les simulations et somme le total de bateaux dans chaque case en donnant la priorité aux bateaux qui chevauchent les coups existants.
5. Fait la moyenne de chaque case, donnant une matrice de fréquences.
6. Prend la plus grande valeur correspondant à un mouvement dans la matrice.
7. Répète jusqu'à la victoire.

Nous avons tracé les performances de la méthode de Monte Carlo et du joueur probabiliste sur 100 jeux avec l'échantillonnage Monte Carlo de 1000 grilles.



4 Senseur imparfait : à la recherche de l'USS Scorpion c.

4.1

$$Y_i(y) = \begin{cases} \pi_i, y = 1 \\ 1 - \pi_i, y = 0 \end{cases}$$

$$P\left(Z_i | Y_i\right) = \frac{P\left(Z_i \cap Y_i\right)}{P\left(Z_i\right)}$$

On donne la loi marginale d'un vecteur aléatoire:

	$P(Y_i = 0)$	$P(Y_i = 1)$
$P(Z_i = 0)$	$1 - \pi_i$	$(1-p_s) \times \pi_i$
$P(Z_i = 1)$	0	$p_s \times \pi_i$

4.2

D'après la loi marginale de la question 4.1, nous obtenons que $P(Y_k = 1 \cap Z_k = 0) = (1 - p_s) \times \pi_i$ et $P(Z_k = 0) = P(Z_k = 0 \cap Y_k = 0) + P(Z_k = 0 \cap Y_k = 1) = 1 - \pi_i \times p_s$, selon la formule de Bayes:

$$P(Y_k = 1 | Z_k = 0) = \frac{P(Y_k = 1 \cap Z_k = 0)}{P(Z_k = 0)} = \frac{(1 - p_s) \times \pi_i}{1 - \pi_i \times p_s} \tag{5}$$

4.3

D'après la loi marginale de la question 4.1 et la formule de Bayes:

$$\pi_k = P(Y_k = 1 | Z_k = 0) = \frac{(1 - p_s) \times \pi_i}{1 - \pi_i \times p_s} \tag{6}$$

4.4

Si on a sondé la case k et qu'il n'y a pas eu de détection, on mets a jour la probabilité π pour $i \neq k$ d'une matrice de taille $n \times n$:

$$\pi = \pi_i + \frac{\pi_i - \pi_k}{n^2 - 1} \tag{7}$$

On a implémenté l'algorithme et étudié son comportement avec une distribution $\pi = 0.8$ de la probabilité a priori sur une grille de recherche 10×10 (figure 5).

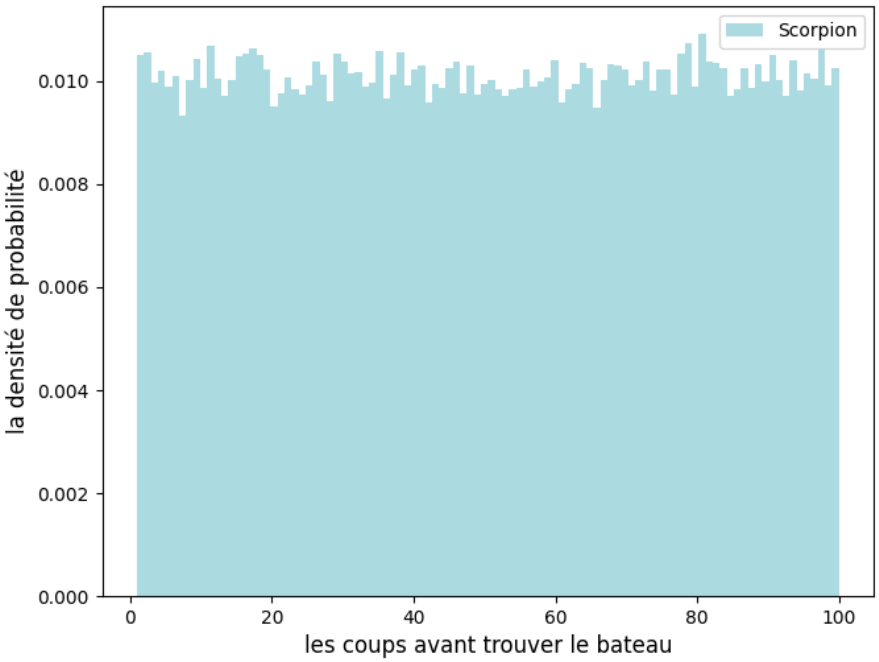


Figure 5: cherche Scorpion avec senseur de $\pi_i = 0.8$