Université Pierre et Marie Curie

Bataille navale

Rapport de projet

Pablo Donato
Alexandre Doussot
3I005 – Probabilités, Statistiques et Informatique
Paris, le 18 février 2017

Professeur : Eloi Zablocki

Combinatoire du jeu

Question 1

Pour une grille vide $G \in \{0\}^{m \times n}$ et un bateau de longueur l, on peut calculer le nombre de placements possibles du bateau sur G, que l'on appelle n_l, avec la formule :

$$n_l = n \cdot (m - l + 1) + m \cdot (n - l + 1)$$

Dans le fichier `combinatoire.py`, nous avons défini une fonction `nb_placements_bateau` permettant de calculer n_l en énumérant toutes les combinaisons possibles sur une grille donnée à l'aide de la fonction `Grille.peut_placer`.

En appliquant la formule précédente et en appelant la fonction `nb_placements_bateau` sur une grille vide de taille 10×10, on obtient bien des résultats identiques :

l	n_l	`nb_placements_bateau`
5	120	120
4	140	140
3	160	160
2	180	180

Questions 2,4

Soit $L \in \{2, 3, 4, 5\}^n$ la liste des longueurs des bateaux dont l'on veut dénombrer les placements sur la grille vide (de taille 10×10). On appelle n_L l'approximation de ce dénombrement que l'on calcule avec la formule :

$$n_L = \prod_{i=1}^{n} n_{L_i}$$

On a implémenté dans `combinatoire.py` une fonction `nb_placements_liste_bateaux` qui permet d'effectuer le dénombrement à l'aide d'un algorithme récursif. On obtient les résultats suivants :

L	n_L	`nb_placements_liste_bateaux`	Temps de calcul (commande `time`)
[5]	120	120	0.35s
[5,5]	14400	11920	0.97s
[4,3]	22400	20336	1.21s
[2,2]	32400	31252	1.56s
[5,5,5]	1728000	983520	60.51s
[5,4,3]	2688000	1850736	88.99s

On constate qu'à partir de 3 bateaux, on atteint des temps de calcul beaucoup plus élevés que pour 1 ou 2 bateaux. Cela est très probablement dû au coût en complexité de la récursion, qui comme on peut l'observer

expérimentalement implique que le temps de calcul est plus ou moins proportionnel au nombre de cas de base, et donc ici au dénombrement des placements.

Aussi, à part dans le cas où $n = 1$, la mesure approximative donnée par n_L est toujours supérieure au dénombrement réel car elle permet la superposition des bateaux. Cette mesure est d'autant plus approximative que n est grand.

Question 3

Si toutes les grilles sont équiprobables, alors la probabilité de tirer une grille en particulier est de $\frac{1}{N}$, avec N le nombre de grilles calculé à la question 2,4.

On a implémenté dans `combinatoire.py` la fonction `trouve_grille_alea` qui génère des grilles aléatoirement jusqu'à trouver la grille passée en paramètre.

Modélisation probabiliste du jeu

Version aléatoire

Espérance théorique

Dans un premier temps, on cherche à exprimer la loi de probabilité de la variable aléatoire X qui correspond au nombre de coups effectués pour gagner la partie.

Pour ce faire, on peut modéliser notre univers, l'ensemble des grilles de jeu de taille $m \times n$ contenant k tirs réussis (k étant le nombre de cases occupées par les bateaux), comme l'ensemble des séquences de digits binaires $\Omega = \{(c_1, \ldots, c_{m \cdot n}) \in \mathbb{B}^{m \cdot n}\}$.
Un 1 correspond à un tir réussi, un 0 à un tir raté. La séquence est bien toujours de taille $m \times n$ car on élimine les positions déjà jouées (et on complète par des 0 pour les tirs non-effectués car la partie est terminée).

On peut alors exprimer le nombre de grilles de jeu gagnantes en au plus x coups, $|X \leq x|$, comme le nombre de séquences de longueur x contenant k 1, C_x^k. On en déduit la loi de probabilité et l'espérance qui s'en suit :

$$P(X = x) = \frac{|X \leq x| - |X \leq x - 1|}{|\Omega|} = \frac{C_x^k - C_{x-1}^k}{C_{m \cdot n}^k}$$

$$E(X) = \sum_{x=k}^{m \cdot n} x \cdot P(X = x)$$

Dans notre cas où $k = 17$ et $m \cdot n = 10 \cdot 10 = 100$, on trouve $E(X) \approx 95,39$.

Comparaison avec l'espérance expérimentale

On a implémenté dans le fichier `game.py` la fonction `random_play` qui joue aléatoirement au jeu, et renvoie le nombre de coups.

On calcule ensuite la distribution expérimentale en lançant $N = 1000$ itérations de `random_play` à l'aide de la fonction `distribution` dans `modelisation.py`, puis on plot le résultat, avec la ditribution théorique calculée précédemment :

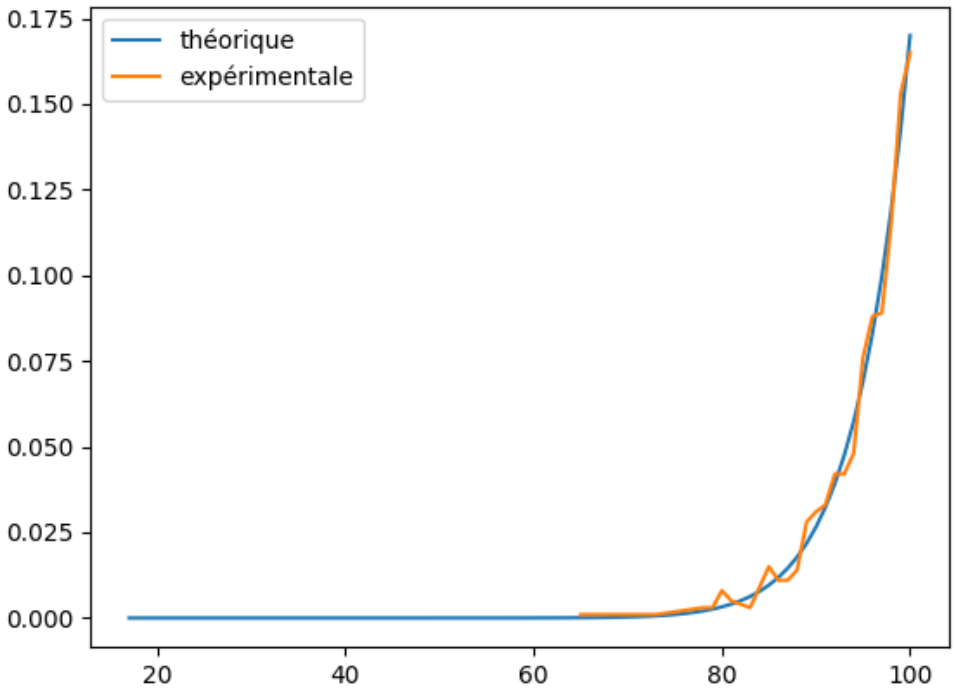

FIGURE 1 – Distribution du nombre de coups : version aléatoire

Les deux distributions semblent donc bien concorder. On observe en effet une espérance expérimentale qui oscille entre 95 et 96 sur plusieurs expériences, et des valeurs qui se rapprochent encore plus de l'espérance théorique lorsqu'on augmente N (nous ne sommes pas allés au-delà de $N = 10000$ pour des raisons de temps de calcul).

Senseur imparfait : à la recherche de l'USS Scorpion

Question 1

$$P(Y_i = 1) = \pi_i$$
$$P(Y_i = 0) = 1 - P(Y_i = 1) = 1 - \pi_i$$

$$P(Z_i = 0|Y_i = 0) = 1$$
$$P(Z_i = 1|Y_i = 0) = 0$$
$$P(Z_i = 0|Y_i = 1) = 1 - p_s$$
$$P(Z_i = 1|Y_i = 1) = p_s$$

Question 2

$$P(Y_k = 1|Z_k = 0) = \frac{P(Y_k = 1 \cap Z_k = 0)}{P(Z_k = 0)} = \frac{P(Z_k = 0|Y_k = 1) \cdot P(Y_k = 1)}{P(Z_k = 0)}$$

Question 3

$$\begin{aligned} P(Y_k = 1|Z_k = 0) &= \frac{P(Z_k = 0|Y_k = 1) \cdot P(Y_k = 1)}{P(Z_k = 0)} \\ &= \frac{(1 - p_s) \cdot \pi_k}{P(Z_k = 0|Y_k = 0) \cdot P(Y_k = 0) + P(Z_k = 0|Y_k = 1) \cdot P(Y_k = 1)} \\ &= \frac{(1 - p_s) \cdot \pi_k}{1 \cdot (1 - \pi_k) + (1 - p_s) \cdot \pi_k} \\ &= \frac{\pi_k - \pi_k p_s}{1 - \pi_k p_s} \end{aligned}$$

Pour mettre à jour π_k, il suffit de lui affecter la nouvelle valeur de $P(Y_k = 1)$, qui en présence de la nouvelle information $Z_k = 0$, est égale à la probabilité $P(Y_k = 1|Z_k = 0)$ que l'on vient de calculer :

$$\pi_k \leftarrow \frac{\pi_k - \pi_k p_s}{1 - \pi_k p_s}$$

Question 4

Pour mettre à jour π_i pour $i \neq k$, on ajoute de manière uniforme l'opposé de la différence $\pi_k^t - \pi_k^{t-1}$, avec π_k^t la nouvelle valeur de π_k, et π_k^{t-1} la valeur de π_k avant mise à jour :

$$\pi_i \leftarrow \pi_i - \frac{1}{N-1}\left(\pi_k^t - \pi_k^{t-1}\right)$$

$\pi_k^t - \pi_k^{t-1}$ sera normalement toujours négatif, puisqu'une réponse négative du senseur doit intuitivement baisser la probabilité π_k. On aura donc constamment une légère augmentation des probabilités π_i de localiser le sous-marin sur les autres cases.