TP NOTE TDJ

1)

Les résultats suivants ont été trouvés à partir de la fonction Exercice1() du fichier main.py.

Pour la configuration (23,20,-1), la distribution de probabilités est la suivante, dans l’ordre décroissant du nombre de pierres lancées :

0, 0, 5.27824322e-02, 7.23108523e-03, 2.93001357e-18, 2.63912161e-02, 3.78066607e-02, 4.24373265e-02, 3.68007861e-02, 4.40924484e-02, 5.10092695e-02, 5.74797379e-02, 6.23638118e-02, 6.68181335e-02, 7.04429121e-02, 7.03334190e-02, 6.64184543e-02, 8.47955377e-02, 1.02769734e-01, 1.20027035e-01, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00]

Pour le gain : -8.79972965e-01.

Pour la configuration (20,x,-1), le jeu est favorable au joueur 2 à partir de x = 11.

2)

La stratégie prudente est une stratégie mixte. Elle a, pour principal intérêt, de maximiser les gains du joueur qui la joue. Cependant, nous pouvons mesurer l’efficacité de la stratégie prudente seulement sur un grand nombre d’itérations. En tant que stratégie optimale, la stratégie prudente est supposée être efficace face à une stratégie qui intègre de l’IA. Théoriquement, les meilleures stratégies issues de l’IA tendront vers la stratégie prudente. Il s’agit du principe du théorème MinMax : le jeu du troll étant un jeu à somme nulle (les gains du joueur 1 correspondent aux pertes du joueur 2), la stratégie prudente est optimale.

3)

Les résultats suivants ont été trouvés à partir de la fonction Exercice3() du fichier main.py :

Pour trouver le gain du joueur 1 sachant que le joueur 2 joue aléatoirement, nous avons fait comme ceci :

* Pi la probabilite de lancer n1-i pierres pour le joueur1
* Pj la probabilite de lancer n2-j pierres pour le joueur2
* Gij le gain quand il reste i pierres au J1 et j pierres au J2
* G = somme des i (Xi\*(somme des j (Xj\*Gij)))

On trouve un gain de 0.94.

Le joueur 1 peut faire mieux dans ce cas, en jetant n/3+1 pierres au joueur 2. Son gain serait alors de 1.

4)

En ayant combiné les 3 modifications, le gain est négatif à partir de x = 56.

Modifications du code :

Dans JeuDuTroll.py : adaptation des règles dans la fonction PartieModifiee :

* Le joueur 2 rejoue sa stratégie si il a choisi un nombre pair (limite arbitraire de 500 pour prévenir des boucles infinies)
* Ajout d’une clause dans le cas ou le coup du joueur 1 est supérieur au coup du joueur 2 (le cas ou le troll avance de deux cases)
* Avant de prendre le choix du joueur 2 en compte pour l’avancée du troll, vérifications par une variable aléatoire que le joueur 2 a réussi son coup.

Dans Solveur.py : adaptations de la matrice de gains dans la fonction MatriceGainsModifiee :

* Ajout du cas où le troll avance de deux cases
* Suppression des colonnes ou le joueur 2 ferait un coup interdit (restera valeur infinie) pour le calcul de la valeur d’une case, mais aussi pour le calcul des simplex.
* On part du principe que tous les coups du joueur 2 font mouche, c’est le pire des cas, donc pas de modification des gains des valeurs triviales.

Dans Strategies.py, adaptations de la stratégie prudente dans la fonction StrategiePrudenteModifiee :

* Appel à la matrice de gains modifiée
* Filtrage des coups interdits pour le joueur 2 (les valeurs infinies dans la matrice)

G(20,x,0) <0 pour x = 56 : fonction Exercice4.