TP1 Simulation de Système

Terenui Rouby et Karim Mouaddel

February 12, 2019

Contents

1 Partie 1

1.1 Dealer

1.2 Mélange du deck

Afin d'effectuer le mélange du jeu de carte on utilise la méthode *shuffle()* qui est inclut dans la classe dealer. On s'est inspiré pour implémenter cette méthode d'une méthode de mélange de jeu de cartes trouvé sur internet[1]. Pour effectuer ce mélange nous avons utilisé la méthode Random provenant de la bibliothèque java.util qui est un générateur de nombre pseudo aléatoire en java(générateur à congruence linéaire).

1.3 Résultat

Numéro du jeu	Pourcentage de victoire
1	7,7484
2	1,9175
3	47,1514
4	58,6167
5	22,0844

1.4 Pertinence des résultat

Tous les résultats semblent se rapprocher de l'idée que l'on peut se faire des probabilités. Le générateur de nombre pseudo-aléatoire semble assez efficace dans notre contexte: il ne fausse pas les résultats. Cela peut être vérifier avec des jeux dont la probabilité peut être facilement calculer (exemple: le jeu 1 qui a bien une probabilité approchant 1/13 après simulation).

2 Partie 2

2.1 Résultat de la simulation

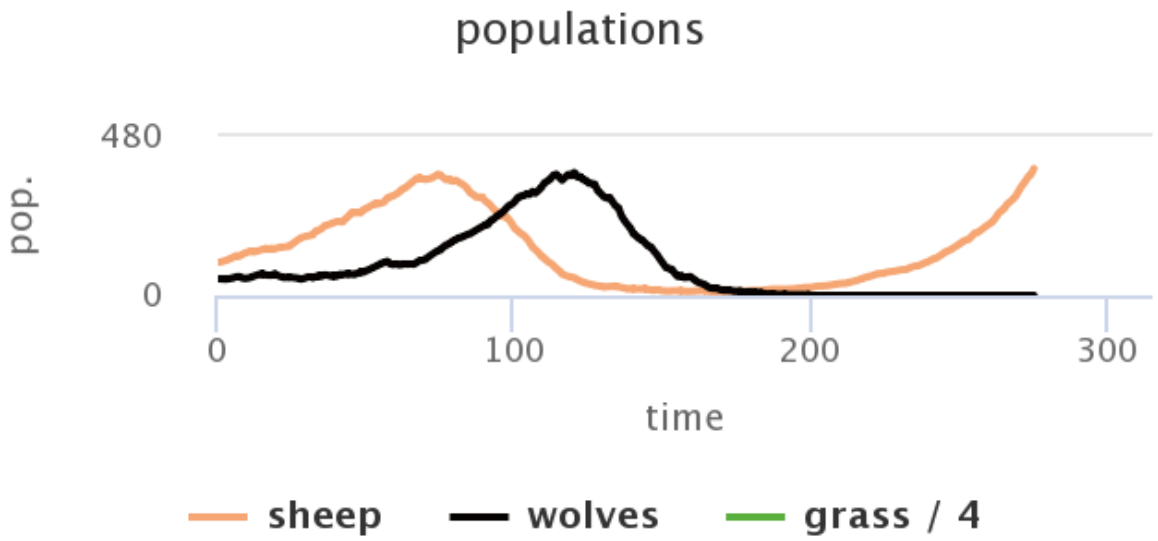

Figure 1: Graphique d'une simulation Prédateur Proie normale

2.2 Analyse des résultats

Quand le paramètre grass regrowth time est mis à 10 les moutons se déplacent très peu. De ce fait lorsque la population de mouton diminue les loups dépensent plus d'énergie pour les atteindre. Arrivé à un certain point la population de loup disparait par manque de nourriture(plus d'energie) car ils ne peuvent pas atteindre les moutons isolés(si il en reste).

References

[1] *https://sebastien-estienne.developpez.com/tutoriels/java/java-chap7/?page=exo7.*