Projet Algorithmique

Proies et prédateurs sont des bandits

Enseignant : L. Ralaivola

Date : 6 janvier 2014

1 Objectifs du projet

Langage C. Ce projet a plusieurs objectifs. En tout premier lieu, il s'agit évidemment pour les étudiants de se familiariser avec le langage C au travers d'une mise en pratique sur un sujet ludique, celle des bandits à K-bras. Les notions prioritaires qui seront abordées seront celles de pointeurs et de programmation modulaire ; une petite partie du projet s'intéressera à l'usage de la librairie graphique Gtk+ pour la réalisation d'une illustration graphique des algorithmes développés.

Bandit à K bras. Un deuxième objectif est de comprendre une problématique scientifique amusante : celle, déjà mentionnée, des bandits à K-bras. Cette problématique mathématique, dont s'est emparée il y a peu de temps la communauté d'apprentissage automatique (qui s'intéresse à faire en sorte que les ordinateurs soient capables d'*apprendre*, comme les êtres humains), trouve son origine dans les machines à sous, autrement appelées bandits manchots [1], que l'on trouve dans les casinos. La question que se pose un joueur qui se rend dans un casino et qui veut jouer sur ces machines, est évidemment de maximiser ses gains. Face à K machines différentes [2], dont chacune rapporte un gain – inconnu du joueur – moyen différent, le joueur est confonté au problème *« exploitation/exploration » :* s'il a essayé M machines différentes, avec $M < K$, et qu'il a identifié en faisant la moyenne de ses gains sur chacune d'elle que la machine m rapporte plus que les autres, il peut décider d'*exploiter* cette connaissance et ne jouer que cette machine à sous. Il se peut néanmoins qu'une autre machine, parmi les $K - M$ non encore essayées, rapporte plus que la machine m : pour le savoir, le joueur est obligé d'essayer ou *explorer* les machines non encore utilisées. Le joueur doit donc définir une stratégie d'exploration/exploitation lui permettant, au bout de T actions de jeu, d'avoir un gain aussi proche que possible qu'un joueur ayant joué T fois sur la machine m^* dont la moyenne des gains associés est la plus élevée. Des stratégies très efficaces existent, dont, en particulier, les stratégies `UCB` (pour Upper Confidence Bound) et ε-`gloutonne`, et il s'agira de les programmer.

Prédateurs et proie. Une illustration de cette problématique de bandits à K bras est la suivante. Des prédateurs sont présents dans un environnement donné, dans lequel se trouve une proie. Ces prédateurs ont la possibilité de choisir une parmi K directions possibles à chaque pas de temps, sachant que certaines de ces directions ne permettent pas de s'approcher de la proie, alors que d'autres donnent la possibilité de s'en approcher très rapidement. Pour

1. Ces machines d'usage très simple fonctionnent comme suit : il suffit d'y introduire une pièce et d'en actionner le bras pour savoir si/combien on a gagné.

2. Notons que l'on appelle le problème celui des bandits à K bras plutôt que celui des K bandits à 1 bras.

découvrir la(les) bonne(s) direction(s) à choisir, les prédateurs implémentent simplement les stratégies de la littérature pour le problème des bandits à K-bras et la vitesse avec laquelle les prédateurs fondent (ou pas) sur la proie permet de caractériser la pertinence et l'efficacité des algorithmes de bandits existants. L'interface graphique qui sera programmée au cours de ce projet aura pour objectif de permettre une visualisation du comportement des prédateurs suivant les stratégies retenues. Un exemple de réalisation est fourni sur la Figure 1.

Créativité. Enfin, ce projet doit offrir la possibilité à chacun de laisser libre cours à son imagination et sa créativité. Une extension assez directe de l'affrontement entre les prédateurs et la proie et celle où la proie n'est plus fixe, mais qu'elle implémente également un algorithme de bandit pour échapper à ses prédateurs. Si les algorithmes de bandits standard sont toujours utilisables, la configuration ne répond alors plus exactement aux hypothèses classiques de leur utilisation : les gains moyens associés à chaque direction (i.e. bras) des prédateurs et de la proie ne sont plus fixes mais *évoluent dans le temps*, puisqu'une direction qui est très prometteuse à un instant donné, peut ne plus l'être quelques instants après. Les étudiants sont invités à créer leurs propres algorithmes de bandits pour gérer cette situation, où la question de simuler adéquatement « l'oubli » est particulièrement centrale.

Concepts rencontrés. Les concepts rencontrés au cours de ce projet sont les suivants :

- Méthodologie : programmation modulaire, fichiers en-tête, `makefile` ;
- Algorithmique/structure de données : liste chaînée, algorithme UCB ;
- Technique : passage de paramètres par adresse/valeur, pointeurs, interface graphique, Gtk+ ;
- Apprentissage automatique/statistique : bandits à K-bras, stratégie UCB, stratégie ε-gloutonne.

2 Modalités

- Langage de programmation : C ;
- Chaque rendu sera composé d'un rapport et du code réalisé.
- Dates de rendus pour les étudiants en présentiel :
  - **21/02/2014.** Rendu 1 : les algorithmes de bandits UCB et ε-glouton sont programmés ; le programme est découpé en plusieurs modules ; la compilation du projet se fait grâce à un `makefile`.
  - **19/03/2014.** Rendu 2 : une interface graphique permet de visualiser les bandits et l'évolution de leurs trajectoires au cours du temps. La proie est fixe.
  - **16/04/2014.** Rendu 3 : la proie peut maintenant être mobile et utiliser un algorithme de bandit pour se diriger. Des algorithmes originaux de bandits sont proposés et mis en oeuvre. Le projet est terminé.
- Date de rendu pour les étudiants en télé-enseignement (les étudiants peuvent également se caler sur le calendrier des rendus prévu pour les étudiants en présentiel et déposer sur le site leurs rendus intermédiaires).

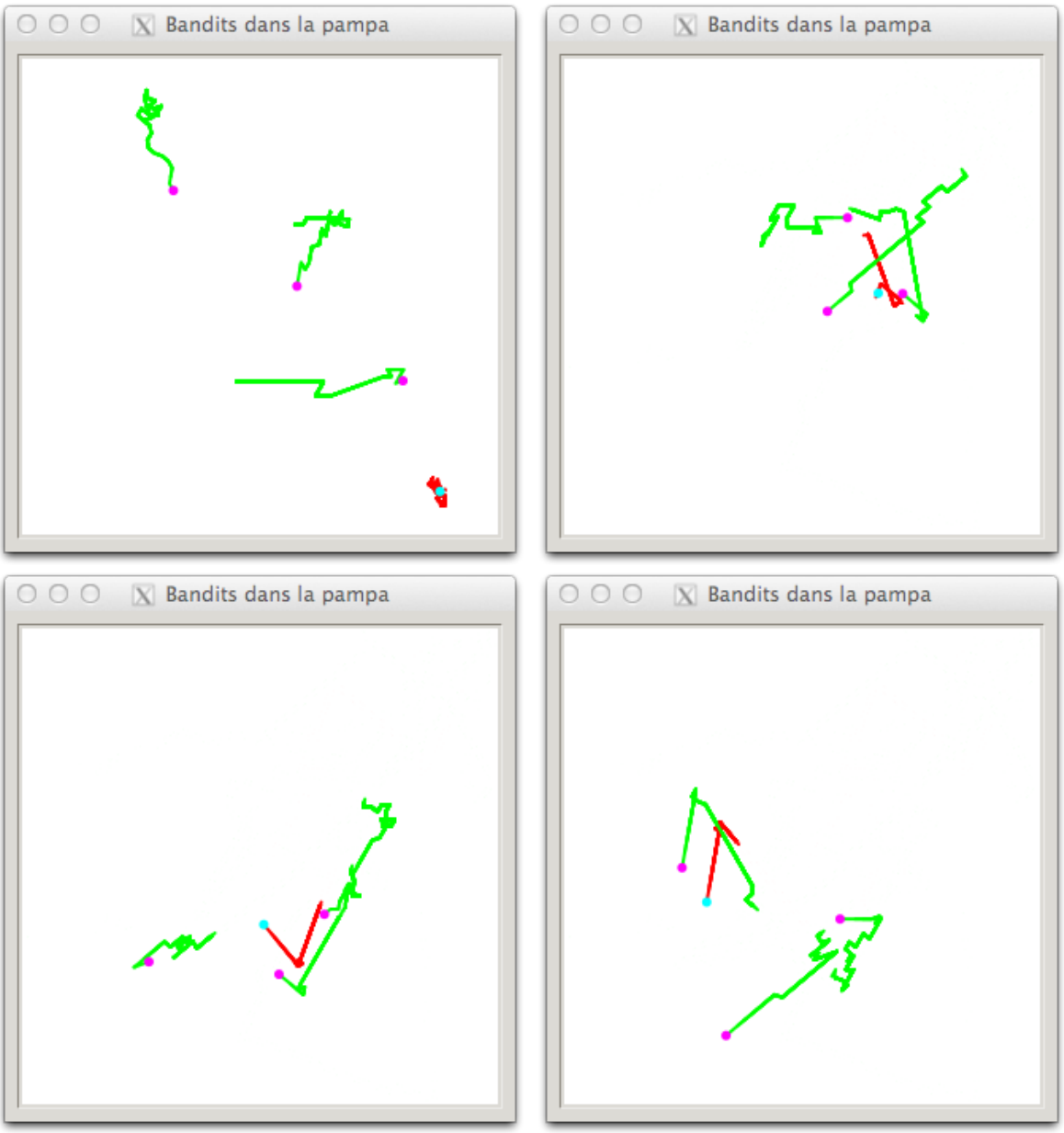

FIGURE 1 – Interface graphique représentant des prédateurs, dont la trace est verte et une proie (mobile) dont la trace est rouge. Les figures sont rangées par ordre chronologique, de gauche à droite et de haut en bas : en haut à gauche, on voit une phase d'exploration, notamment de la part de la proie, qui essaie toutes les directions possibles ; lorsqu'on se dirige vers la droite (et vers le bas), on constate que le processus de poursuite s'engage et que les prédateurs essaient d'exploiter des directions prometteuses qui leur permettent de s'approcher de la proie le plus rapidement possible (cf. figure en bas à droite, le prédateur qui a une trajectoire parallèle à la proie et celui qui amorce une trajectoire visant directement la proie).

3 Tableau de marche indicatif

La charge de travail peut se répartir sur 12 semaines environ.

Semaine 1 : Compréhension du sujet, écriture d'une structure **struct bandit** pour représenter un bandit à K bras ;

Semaine 2 : Ecriture d'un fichier en-tête `bandit.h` comportant les signatures des fonctions associées à un bandit, ainsi que le fichier `bandit.c` qui définit les fonctions. Ecriture d'une stratégie aléatoire qui choisit les actions/les bras au hasard. Mise en place du `makefile`.

Semaine 3 : Implémentation de l'algorithme `UCB` et ε-`glouton`. Premiers tests, avec un seul bandit.

Semaine 4 : Tests intensifs du programme réalisé avec variation des paramètres d'exploration/exploitation. **Rendu 1.**

Semaine 5 : Prise en main de l'interface graphique écrite en Gtk+, fournie : tracé de formes simples telles que carrés, cercles, lignes. Affichage d'une trajectoire d'un point qui se déplace aléatoirement sur l'écran.

Semaine 6 : Augmentation de la **struct bandit** pour qu'elle contienne des informations de localisation et qu'un bandit puisse être représenté sur l'interface graphique. Choix d'une représentation graphique pour les proies et les prédateurs.

Semaine 7 : Intégration des bandits dans l'interface Gtk+. Un seul prédateur et une proie fixe sont représentés.

Semaine 8 : La possibilité d'ajouter plusieurs prédateurs est offerte. La proie est toujours fixe. Tests intensifs. **Rendu 2.**

Semaine 9 : La proie va maintenant pouvoir bouger, suivant une stratégie de bandit. La définition des actions pour la proie devra être inventée.

Semaine 10 : On peut faire évoluer plusieurs bandits en même temps.

Semaine 11 : Des stratégies nouvelles sont imaginées pour tenir compte de la non-stationarité des récompenses.

Semaine 12 : Tests intensifs et fin du projet. Rendu 3.