Université Paris 6
MASTER ANDROIDE M2 – MADI

2015-2016

MADI – Mini-Projet

Le cavalier à marche aléatoire

Introduction

On s'intéresse aux déplacements d'un cavalier dans une grille donnée dont certaines cases sont interdites (cases colorées en noir). On considère que le cavalier est initialement dans le coin supérieur gauche de la grille et l'on veut atteindre une case cible donnée (typiquement le coin inférieur droit) en un minimum de mouvements. En chaque case, le cavalier choisit une action parmi l'un des huit coups légaux autorisés aux échecs (chaque action correspond à une case cible distincte qui ne doit pas être hors de la grille et ne doit pas être une case interdite). Ces actions sont notées R,T,Y,U,J,H,G,F respectivement et sont représentées dans la figure ci-dessous (le rond jaune représente la position du cavalier et les lettres représentent les différentes cases cibles possibles correspondant aux 8 actions.

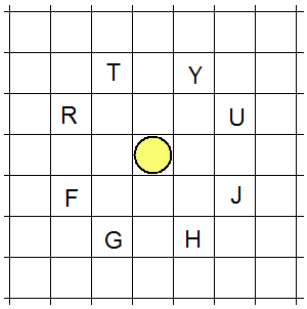

FIGURE 1 – Actions possibles

Malheureusement, les effets de ces différentes actions ne sont pas nécessairement déterministes. Dans le cas non-déterministe, on supposera que la case cible n'est pas atteinte avec certitude et que l'une des 8 cases périfiques peut l'être également pourvu qu'elle ne soit pas interdite. Si q est le nombre de cases périphériques non-interdites autour d'une case cible donnée ($q \leq 8$), la probabilité d'atteindre la case cible est $1 - \frac{q}{16}$ et la probabilité de chaque case périphérique est $\frac{1}{16}$. Ainsi dans l'exemple ci-dessous, si l'on déclenche l'action J, les 8 cases périphériques autour de J sont celles du carré rouge. Ici on observe que 2 d'entre elles sont interdites donc ici $q = 6$. La probabilité de tomber sur J est donc de 0.625 et la probabilité de chaque case périphérique est 0.0625.

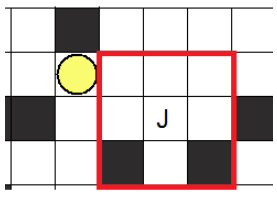

FIGURE 2 – Effet aléatoire d'une action

L'objet de ce miniprojet est de modéliser différents problèmes de planification dans cette grille comme des processus décisionnels Markoviens, puis d'implanter et de tester des algorithmes permetant de déterminer

les politiques optimales de déplacement dans cette grille à l'horizon infini. Certaines cases de la grille sont marquées d'un point de couleur pour signifier qu'elles engendrent une conséquence spécifique lorsqu'on passe dessus. Cette conséquence est codifiée par une échelle de couleur (vert, bleu, rouge, noir). Selon les cas, les couleurs pourront désigner différents niveaux de risque encourus ou de coûts, ou encore différents types de récompenses dans d'autres problèmes. Un exemple de telle grille est donné ci-dessous, le disque jaune représentant la position du cavalier, et les cases noires les cases interdites.

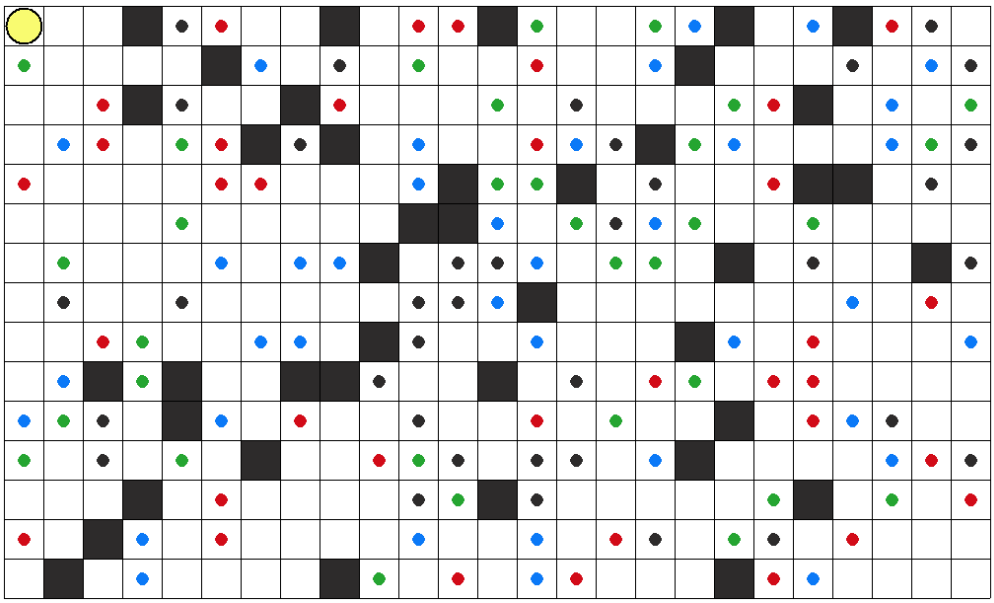

FIGURE 3 – Un exemple de grille tiré aléatoirement

Pour engendrer de telles grilles et simuler les déplacements d'un cavalier suivant une politique donnée on utilisera le programme python mdpmadi16v1.py joint à ce document, programme qu'on pourra éventuellement modifier ou enrichir. Ce programme permet de guider manuellement le cavalier en indiquant les actions à effectuer au clavier. Il permet aussi, en appuyant sur space, d'executer une politique préalablement calculée (par défaut une politique aléatoire s'execute dans la version initiale).

1) Recherche d'une trajectoire prudente

Dans un premier temps, on considère que les couleurs représentent le niveau de risque encouru en traversant chaque case, avec un code de couleurs emprunté à celui des pistes de ski (vert : risque très faible, bleu : risque faible, rouge : risque moyen, noir : risque élevé). Les cases sans point de couleur correspondent à un risque nul. Pour les autres on suppose que ces niveaux de risques sont codés par une fonction coût (additive) donnée par le tableau suivant :

x	vert	bleu	rouge	noir
$c(x)$	10	20	30	40

On suppose ici que chaque déplacement de cavalier coûte 1 plus une éventuelle pénalité si l'on se situe sur une case colorée. On cherche alors à déterminer la politique de moindre coût pour atteindre la case cible.

a) Modéliser le problème comme un processus décisionnel Markovien en donnant une récompense significative lorsqu'on atteint la case but (par exemple un bonus de 1000 points mais on pourra essayer d'autres valeurs). Ecrire les équations de Bellman pour ce problème en utilisant un facteur d'actualisation γ.

b) On se place ici dans le cas déterministe. Déterminer par itération de la valeur un chemin optimal de la case initiale vers la case but et tester sur une grille de taille 10×10, puis 10×15, 15×20, 20×30 (on

utilisera $\gamma = 0.9$). On donnera les grilles en visualisant les trajectoires optimales et on donnera les temps de calcul.

c) On se place maintenant dans le cas de transitions aléatoires définies comme expliqué ci-dessus. On souhaite déterminer la politique optimale par itération de la valeur. Effectuer des essais numériques de résolution sur différentes tailles de grille (par exemple 10×10, puis 10×15, 15×20, 20×30). On pourra, ici aussi, jouer à essayer une résolution à la main avant de calculer la solution optimale. Pour chaque taille on tirera 10 instances aléatoirement et on donnera dans un tableau les temps de résolution et le nombre d'itérations. On pourra également étudier l'impact de γ en faisant varier $\gamma = 0.9, \gamma = 0.7, \gamma = 0.5$.

2) Recherche d'une politique équilibrée

Dans cette partie on considère que chaque saut de cavalier coûte 2 mais que chaque case de couleur rapporte une récompense de 1. De plus, on considère ici des grilles contenant environ 15% de cases interdites, ainsi que 20% de cases rouges et 20% de cases bleues, les couleurs vertes et noires n'étant plus présentes.

a) On cherche une politique optimale pour atteindre l'état but à partir de l'état initial tout en collectant de manière équilibrée les récompenses liées aux cases rouges et aux cases bleues. Pour une politique π, on utilise un premier critère $c_b(\pi)$ qui représente le coût espéré de la politique π en considérant que seules les cases bleues engendrent une récompense de 1 (les rouges ne sont pas prises en compte) ; on considère alors un second critère $c_r(\pi)$ qui représente le coût espéré de la politique π en considérant que seules les cases rouges engendrent une récompense de 1 (les bleues ne sont pas pris en compte). On souhaite trouver une politique qui optimise le critère suivant :

$$\min_{\pi} \max\{c_b(\pi), c_r(\pi)\}$$

Expliquer pourquoi on ne peut utiliser l'itération de la valeur pour la résolution de ce problème. Proposer alors une approche de résolution reposant sur la formulation d'un MDP multiobjectif (MOMDP) et sa résolution par programmation mathématique.

b) Tester la résolution pratique du problème de recherche d'une trajectoire équilibrée entre le bleu et le rouge sur des grilles de différentes tailles avec l'approche proposée à la question précédente. On donnera ici encore les temps moyens de résolution. Simuler les politiques optimales obtenues en les éxécutant plusieurs fois sur la même instance. A chaque réalisation, dessiner le point (c_b, c_r) qui représente le coût empiriquement observé de la politique optimale selon les deux critères considérés.

c) Une alternative à l'approche précédente consite à optimiser le critère suivant :

$$\min_{\pi} \frac{c_b(\pi) + c_r(\pi)}{2}$$

Résoudre ce problème par itération de la valeur sur les mêmes grilles que celles utilisées à la question b. Simuler les politiques optimales obtenues en les éxécutant plusieurs fois sur la même instance. A chaque réalisation, dessiner le point (c_b, c_r) et comparer les résultats obtenus à ceux de la question b.

Accès au solveur Gurobi et à une bibliothèque python pour les grilles

Le mini-projet sera développé de préférence en python, langage qui permet de développer facilement les interfaces utiles pour visualiser la grille (voir exemple donné) mais aussi pour dialoguer facilement avec gurobi solver quand il s'agit de résoudre des programmes linéaires. Pour résoudre les programmes linéaires, le solveur gurobi (http ://www.gurobi.com/) est installé dans les salles en libre-service permanent mais peut aussi être installé sur des machines personnelles en téléchargeant depuis une adresse de l'UPMC.

Modalités de travail et délivrables

Le travail est à effectuer en binôme (de préférence). La date de rendu des projets sera précisée prochainement sur cette page. Les projets seront envoyés par mail à *patrice.perny@lip6.fr*. Votre livraison sera constituée d'une archive zip qui comportera les sources du programme, un fichier README détaillant comment compiler et exécuter le programme, et un rapport (un fichier au format pdf) avec les réponses aux différentes questions. Le plan du rapport suivra le plan du sujet.