Listing des exemples de MDP

Emmanuel Hyon

3 décembre 2020

Table des matières

1 Introduction **2**
1.1 Portabilité 2

2 Le solveur sous c++ : MarmoteMDP **2**
2.1 Description de MarmoteMDP 2
2.1.1 Description du logiciel 2
2.1.2 Description du dossier 2
2.2 Installation du logiciel 3
2.2.1 Installer les sources 3
2.2.2 Vérification de l'installation 3
2.3 Documentation 4
2.4 Ajouter un code 4

3 Le solveur sous python : pyMarmoteMDP **5**
3.1 Description de pyMarmoteMDP 5
3.1.1 Description du logiciel 5
3.1.2 Description du dossier 5
3.1.3 Précautions d'usage 5
3.2 Installation du logiciel 5
3.2.1 Pas de paquetage python 5
3.2.2 Vérification de l'image fournie 5
3.3 Documentation 5

4 Description des modèles de MDP implémentés **5**
4.1 Liste des modèles implémentés 5
4.2 Manipulation des ensembles 6
4.2.1 MarmoteJouet0 6
4.3 Modèles à horizon infini et critères escomptés 6
4.3.1 MarmoteJouet10 6
4.3.2 MarmoteJouet11 7

4.4 Modèles à critère moyen 8
4.4.1 MarmoteJouet20 8
4.4.2 MarmoteJouet21 9
4.5 Modèles à horizon infini et critère total 9
4.5.1 MarmoteJouet30 9
4.5.2 MarmoteJouet31 10
4.6 Modèles à horizon fini et critère total 11
4.6.1 MarmoteJouet40 11
4.7 Modèles à horizon fini et critère discounté 12
4.7.1 MarmoteJouet50 12

5 Ressources Bibliographiques 13

1 Introduction

Ce document présente les deux portages du marmoton (un marmote add-on) pour résoudre les processus de décision Markovien.

Il existe un portage sous c++ et un portage issu en python (c'est à dire que la manipulation se fait sous python mais que le coeur est programmé en c++).

Le code implémenté en C++ permet de résoudre un MDP dans un programme écrit en C++ en incluant la bibliothèque au code.

Il décrit le logiciel, son installation et son utilisation à travers quelques exemples faciles.

1.1 Portabilité

2 Le solveur sous c++ : MarmoteMDP

2.1 Description de MarmoteMDP

2.1.1 Description du logiciel

Le programme marmotecore est une suite logicielle dédiée aux chaînes de Markov et à leur manipulation numérique. Vous pouvez trouvez des informations sur ce soft là :
http ://marmotecore.gforge.inria.fr/dokuwiki/doku.php ?id=start
et des images précompilées et des exemples ici :
http ://www-sysdef.lip6.fr/ hyon/Marmote/home.php

Le programme marmoteMDP est le paquetage de marmotecore spécifique aux Processus de Décision Markoviens (PDM).

2.1.2 Description du dossier

Le dossier comporte plusieurs dossiers et fichiers.

Les fichiers MDPjouetXX.cpp qui sont des exemples de fichiers codant un modèle de MDP particulier. Il y a un autre fichier source test_install_MDP.cpp pour tester la bonne installation du logiciel. Enfin le fichier Makefile qui décrit comment générer les executables associés aux exemples.

Les dossiers sont

— Documentation qui comporte les manuels
— marmoteBase qui comporte les bibliothèques `marmotecore`
— marmoteMDP qui comporte les bibliothèques pour le MDP
— sources qui comporte les sources des bibliothèques

2.2 Installation du logiciel

Pour pouvoir manipuler sous forme numérique des MDP il faut installer une partie de la suite logicielle marmotecore : marmoteBase et également il faut installer le paquetage spécifique aux Processus de Décision Markoviens marmoteMDP.

2.2.1 Installer les sources

Il faut installer le sous ensemble des fichiers de marmoteBase nécessaires au paquetage marmoteMDP.

Installer marmoteBase

1. Aller dans le répertoire `sources/marmoteBase`
2. Exécuter le Makefile de marmoteCore en tapant :
   `make distribs`

Installer marmoteMDP Il faut installer marmoteMDP

1. Aller dans le répertoire `sources/marmoteMDP`
2. Exécuter le Makefile de marmoteMDP en tapant :
   `make distribs`

Si tout se passe bien vous avez installé les paquetages.

Les logiciels sont maintenant installés dans les deux répertoires dédiés : `marmoteMDP` et `marmotecore` du répertoire courant et pas ceux des sources.

2.2.2 Vérification de l'installation

Retourner dans le répertoire courant celui qui contient les fichiers exemples.

1. Nettoyer les images préexistantes en tapant `make clean`
2. Compiler les fichiers `.cpp` présents en tapant la commande `make test`
3. Lancer les exécutables de test :
   (a) `./test_install_MDP`. Fichier qui teste les principales fonctionnalités d'un MDP en une dimension.

Détail des tests Les valeurs que l'on doit obtenir après lancement de `./test_install_MDP` sont $v = (12, 28763; 11, 31989)$ pour le premier MDP et $v = (5.204674, 5.380112)$ pour le second.

Il y a la constructions de deux MDP et pour chacun d'eux la politique est calculée et on recalcule le coût de la politique.

Il faut aussi noter que le fichier source /test_install_MDP est commenté et explicite chaque action.

2.3 Documentation

Il existe aussi une documentation qui vient avec le dossier. Il s'agit du manuel (généré par doxygen) du paquetage marmoteBase et celui du paquetage marmoteMDP. Chaque classe des paquetages est décrite avec ses méthodes, ses attributs ainsi que ses constructeurs.

2.4 Ajouter un code

Pour ajouter un code dans le dossier avec les fichier `MDPjouet`, il suffit de créer un nouveau fichier cpp et d'écrire le code associés. Pour compiler ce code, il suffit de rajouter une cible dans le fichier Makefile (similaire dans sa forme aux autres entrées du fichier) et l'ajouter à la liste des *applis*.

Par exemple : si votre fichier s'appelle monMDP.cpp, on rajoute une cible (l'espace sur la seconde ligne est une tabulation) :

```
monMDP: monMDP.cpp
    $g $(CFLAGS) -I$(INCLUDEDIR) -I$(MDPINCLUDE) $^ -o $@ -L$(LIBDIR) -L$(MDPDIR) $(LIBRARIES)
```

Et on rajoute cette cible à la liste des APPLIS.

3 Le solveur sous python : pyMarmoteMDP

3.1 Description de pyMarmoteMDP

3.1.1 Description du logiciel

3.1.2 Description du dossier

3.1.3 Précautions d'usage

3.2 Installation du logiciel

3.2.1 Pas de paquetage python

3.2.2 Vérification de l'image fournie

3.3 Documentation

4 Description des modèles de MDP implémentés

Un certain nombre de modèles de MDP sont déjà implémentés.

Lorsque l'exemple est implémenté en C++ il existe un fichier .cpp, lorsque l'exemple est implémenté en python

4.1 Liste des modèles implémentés

Modèle	Nom fichier	Catégorie	Descriptif
1	MDPjouet0.cpp	Manipulation des ensembles	Exemple de manipulation, espace d'état à 2 dimensions.
2	MDPjouet10.cpp	Horizon infini critère escompté	Exemple d'un MDP à deux états.
3	MDPjouet11.cpp	Horizon infini critère escompté	Exemple d'un MDP qui illustre l'intérêt de matrice creuse.
4	MDPjouet20.cpp	Horizon infini critère moyen	Exemple tiré du livre de Puterman.
5	MDPjouet21.cpp	Horizon infini critère moyen	Exemple d'un problème de fiabilité.
6	MDPjouet30.cpp	Horizon infini critère total	Exemple sans signification
7	MDPjouet31.cpp	Horizon infini critère total	Exemple lié à un plus court chemin stochastique
8	MDPjouet40.cpp	Horizon fini critère total	Exemple tiré du livre de Puterman.
9	MDPjouet50.cpp	Horizon fini critère escompté	Exemple tiré d'un cours.

4.2 Manipulation des ensembles

Un des avantages de marmoteBase tient dans le fait qu'il manipule des ensemble où l'état (ou l'action) est un ensemble à plusieurs dimensions avec des codage par index et des décodage par index.

4.2.1 MarmoteJouet0

Ici l'espace d'action est un produit de deux ensemble et l'espace d'état est une boîte. C'est à dire un tableau à plusieurs dimensions.

4.3 Modèles à horizon infini et critères escomptés

4.3.1 MarmoteJouet10

Modèle On utilise un modèle à deux états (x_1, x_2) et dans chaque état deux actions (a_1, a_2).

Les probabilités de transitions sont :

$\mathbf{p}(x_1\|x_1, a_1) = 0.6$	$\mathbf{p}(x_2\|x_1, a_1) = 0.4$
$\mathbf{p}(x_1\|x_1, a_2) = 0.2$	$\mathbf{p}(x_2\|x_1, a_2) = 0.8$
$\mathbf{p}(x_1\|x_2, a_1) = 0.5$	$\mathbf{p}(x_2\|x_2, a_1) = 0.5$
$\mathbf{p}(x_1\|x_2, a_2) = 0.7$	$\mathbf{p}(x_2\|x_2, a_2) = 0.3$

Les coûts sont

$r(x_1, a_1) = 4.5$	$r(x_1, a_2) = 2$
$r(x_2, a_1) = -1.5$	$r(x_2, a_2) = 3$

Manipulation informatique On représente chacune des matrices associées à une action. Il s'agit des matrices P_1 associée à la décision 1 et P_2 associée à la décision 2

$$P_1 = \begin{bmatrix} 0.6 & 0.4 \\ 0.5 & 0.5 \end{bmatrix} \quad P_2 = \begin{bmatrix} 0.2 & 0.8 \\ 0.7 & 0.3 \end{bmatrix}$$

La matrice de gains associés au processus est :

$$r = \begin{pmatrix} 4.5 & 2 \\ -1.5 & 3.0 \end{pmatrix}.$$

où $r(x, a)$ est la coordonnée sur la ligne x colonne a de r

Une matrice est un objet `sparseMatrix` qui doit être crée. Ensuite, les coordonnées sont ajoutée une à une avec l'instruction `addToEntry`.

Un objet MDP est créé et il y a 3 résolutions. Une par la méthode de value iteration, une par une méthode de value iteration avec Gauss Seidel, une dernière par un méthode de

policy iteration modified. Pour plus de précision sur ces méthodes voir le livre de Puterman [2].

La méthode `policyCost` permet d'évaluer le coût d'une politique donnée et est utilisée pour calculer le coût de la politique optimale qui vient d'être calculée.

Resultat attendu Les résultats doivent être $V(0) = 8,4285$ et $V(1) = 6.9998$ ainsi que $\pi(0) = 0$ et $\pi(1) = 1$.

4.3.2 MarmoteJouet11

Nous décrivons une chaîne avec 3 etats (s_0, s_1, s_2) et 3 actions (a_0, a_1, a_2).
La fonction de transition est donnée par :

Etat Initial	Action	Etat final	Probabilité
s_0	a_0	s_0	0.7
	a_0	s_1	0.3
s_0	a_1	s_0	1.0
s_0	a_2	s_0	0.8
	a_2	s_1	0.2
s_1	a_0	s_1	1.0
s_1	a_2	s_2	1.0
s_2	a_1	s_0	0.8
	a_1	s_1	0.1
	a_1	s_2	0.1

On a des gains qui sont dépendant de la transition finale. Ils sont donnés par

Etat Initial	Action	Etat final	Gain
s_0	a_0	s_0	$r = 10$
s_1	a_2	s_2	$r = -50$
s_2	a_1	s_0	$r = 40$

L'ensemble d'actions possible n'est pas le même en fonction de l'état. On va supposer que l'espace d'action a toujours la même taille quelque soit l'état. Ceci implique que les actions impossibles dans un état ne doivent pas avoir d'effets d'une part (ce qui se traduit par une transition vers soit même). D'autre part, elles doivent être pénalisées pour éviter qu'elle soient choisies.

On va donc avoir les matrices

$$P_0 = \begin{bmatrix} 0.7 & 0.3 & 0 \\ 0 & 1.0 & 0 \\ 0 & 0 & 1.0 \end{bmatrix} \quad P_1 = \begin{bmatrix} 1.0 & 0 & 0 \\ 0 & 1.0 & 0 \\ 0.8 & 0.1 & 0.1 \end{bmatrix} \quad P_2 = \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0 & 0 & 1.0 \\ 0 & 0 & 1.0 \end{bmatrix}$$

et la matrice de reward (avec pénalisation) :

$$R = \begin{bmatrix} 7 & 0.0 & 0.0 \\ 0 & -\infty & -50 \\ -\infty & 32 & -\infty \end{bmatrix}$$

Manipulation informatique On peut coder les matrices stochastiques soit en codant toutes les entrées (même celles dont la valeur de la probabilité est nulle) soit en ne codant que les entrées positives. Nous choisissons cette deuxième solution qui permet de bénéficier de l'effet matrice creuse.

Il y a deux MDP (des MDP pour des critères escomptés soit des objets `discountedMDP` qui sont créés. Le premier avec la matrice de reward sous la forme d'une matrice $r(s, a)$ tandis que le second prend en entrée un vecteur de matrice de coûts d'un état vers un autre pour chacune des actions. La création du mdp va automatiser la construction vers la matrice de coût $r(s, a)$.

On aura les matrices

$$R0 = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\infty \end{bmatrix} \quad R1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\infty & 0 \\ 40 & 0 & 0 \end{bmatrix} \quad R2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -50 \\ 0 & 0 & -\infty \end{bmatrix}$$

L'affichage du second MDP doit donner la même matrice de reward.

Resultat attendu On doit avoir :
$V(0) = 10.769231$, $V(1) = 0.0000$, et $V(2) = 38.218624$.

Aussi :
$\pi(0) = 0,\ \pi(1) = 0,\ \pi(2) = 1.$

4.4 Modèles à critère moyen

4.4.1 MarmoteJouet20

Modèle Ce modèle correspond à l'exemple 8.5.3 du livre de Puterman [2].

On a un espace d'état de taille 3 (s_0, s_1, s_2) et 4 actions possibles : deux dans l'état s_0 : a_0, a_1. Une dans l'état s_1 : a_2 et un dans l'état s_2 : a_3.

Les probabilités de transitions sont :

$$\mathbf{p}(s_2|s_0, a_0) = 1\,, \mathbf{p}(s_1|s_0, a_1) = 1\,, \mathbf{p}(s_0|s_1, a_2) = 1$$

et

$$\mathbf{p}(s_0|s_2, a_3) = \mathbf{p}(s_1|s_2, a_3) = \mathbf{p}(s_2|s_2, a_3) = 1/3\ .$$

Les coûts sont $r(s_0, a_0) = 2$, $r(s_0, a_1) = 1$, $r(s_1, a_2) = 2$ et $r(s_2, a_3) = 3$.

Manipulation informatique Les transitions ont été représentées par 4 matrices creuses avec une seule entrée et 3 entrées pour la dernière. L'objet créé est un objet `averageMDP`.

Resultat attendu On doit obtenir 2.5 comme coût moyen. Les itérations de la méthode d'itération de valeur doivent donner un span de 0.147 apres 5 iterations et un span de 0.005 apres 12 iterations. Ces résultats sont issus du livre de Puterman.

4.4.2 MarmoteJouet21

Modèle Ce modèle est issu du cours de J.Y Potvin.

On suposse une machine qui a 4 états :

— 0 = neuve
— 1 = utilisable avec détérioration mineure
— 2 = utilisable avec détérioration majeure
— 3 = inutilisable

et qui a 3 actions

— 1 = Ne rien faire : reste dans l'état
— 2 = Mise au point : retour à l'état 1
— 3 = Réparation totale : retour à l'état 0

Manipulation informatique Les matrices de transitions sont déclarées dans le code. La politique optimale est calculée suivant 3 méthodes dont on compare les temps d'exécution.

La politique optimale théorique est Rb soit $(1, 1, 2, 3)$ en théorie et soit (0012) dans le modèle informatique. Le cout moyen optimal est 1666.67.

D'autres politiques sont aussi considérées :

— $Ra = (1, 1, 1, 3)$ de coût moyen 1923.
— $Rc = (1, 1, 3, 3)$ de coût moyen 1727.
— $Rd = (1, 3, 3, 3)$ de coût moyen 3.

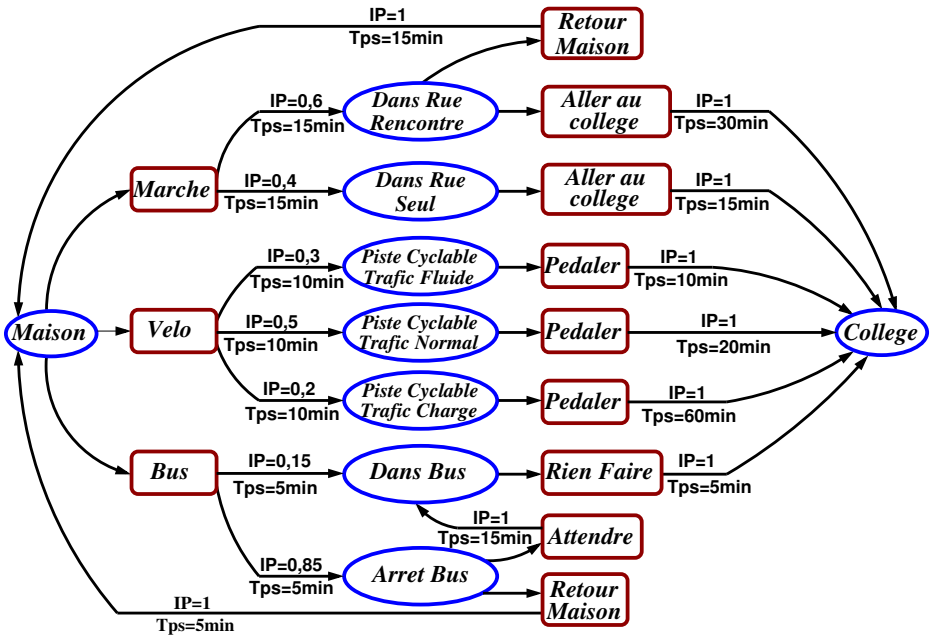
4.5 Modèles à horizon infini et critère total

4.5.1 MarmoteJouet30

Implémentation d'un modèle avec un critère total.

Modèle Il n'y a pas de modèle particulier relié à cet exemple. On va supposer que les matrices sont

$$P_0 = \begin{bmatrix} 0 & 0.875 & 0.0625 & 0.0625 \\ 0 & 0.75 & 0 & 0.25 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}, P_1 = \begin{bmatrix} 0.875 & 0 & 0.125 & 0.0 \\ 0 & 0.75 & 0.125 & 0.125 \\ 0.8 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}, P_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Manipulation informatique La manipulation informatique va utiliser un objet de type `totalRewardMDP`. On utilise le troisième constructeur dans lequel on ajoute les matrice de transition une fois l'objet MDP créé sans créer de vecteur initialement.

On souligne le traitement différents des matrices creuses. Dans une manière on code entièrement la matrice, dans l'autre non et on compare les vitesses.

On fait un test aussi de la propriété de croissance de la fonction de valeur, ce à l'aide d'un objet `structuralValueVF`.

4.5.2 MarmoteJouet31

Modèle Il s'agit d'un modèle de plus court chemin stochastique donné dans le magazine tangente consacré à la RO. Il se décompose comme suit

L'état *Maison* va être représenté par 0 ; l'état *Rue Rencontre* va être représenté par 1 ; l'état *Rue Seul* va être représenté par 2 ; l'état *Piste Fluide* va être représenté par 3 ; l'état *Piste Normal* va être représenté par 4 ; l'état *Piste Chargé* va être représenté par 5 ; l'état *Bus* va être représenté par 6 ; l'état *Arrêt* va être représenté par 7 ; l'état *Collège* va être représenté par 8.

Similairement,
L'action *Marche* va être représentée par 0 ; l'action *Velo* va être représentée par 1 ; l'action *Bus* va être représentée par 2 ; l'action *Retour* va être représentée par 3 ; l'action *Aller au collège* va être représentée par 4 ; l'action *Pédaler* va être représentée par 5 ; l'action *Rien Faire* va être représentée par 6 ; l'action *Attendre* va être représentée par 7.

Pour bien prendre en compte le caractère absorbant de l'état *Collège* (une fois arrivé on y reste). On va ajouter que l'action *Rien Faire* est possible dans cet état et conduit avec une

probabilité de 1 vers ce même état.

Les probabilités de transitions sont :

$$\begin{array}{lll} \mathbf{p}(1|0,0) = 0.6\,, & \mathbf{p}(2|0,0) = 0.4\,; & \\ \mathbf{p}(3|0,1) = 0.3\,, & \mathbf{p}(4|0,1) = 0.5\,, & \mathbf{p}(5|0,1) = 0.2\,; \\ \mathbf{p}(6|0,2) = 0.15 & \mathbf{p}(7|0,2) = 0.85\,. & \end{array}$$

et

$$\begin{array}{lll} \mathbf{p}(0|1,3) = 1\,, & \mathbf{p}(0|7,3) = 1\,; & \\ \mathbf{p}(8|1,4) = 1\,, & \mathbf{p}(8|2,4) = 1\,; & \\ \mathbf{p}(8|3,5) = 1\,, & \mathbf{p}(8|4,5) = 1\,, & \mathbf{p}(8|5,5) = 1 \\ \mathbf{p}(8|6,6) = 1\,, & \mathbf{p}(8|8,6) = 1\,; & \end{array}$$

et finalement

$$\mathbf{p}(6|7,7) = 1\ .$$

Les coûts sont :

$r(0,0)=15$	$r(0,1)=10$	$r(0,2)=5$	$r(0,3)=K$	$r(0,4)=K$	$r(0,5)=K$	$r(0,6)=K$	$r(0,7)=K$
$r(1,0)=K$	$r(1,1)=K$	$r(1,2)=K$	$r(1,3)=10$	$r(1,4)=30$	$r(1,5)=K$	$r(1,6)=K$	$r(1,7)=K$
$r(2,0)=K$	$r(2,1)=K$	$r(2,2)=K$	$r(2,3)=K$	$r(2,4)=15$	$r(2,5)=K$	$r(2,6)=K$	$r(2,7)=K$
$r(3,0)=K$	$r(3,1)=K$	$r(3,2)=K$	$r(3,3)=K$	$r(3,4)=K$	$r(3,5)=10$	$r(3,6)=K$	$r(3,7)=K$
$r(4,0)=K$	$r(4,1)=K$	$r(4,2)=K$	$r(4,3)=K$	$r(4,4)=K$	$r(4,5)=20$	$r(4,6)=K$	$r(4,7)=K$
$r(5,0)=K$	$r(5,1)=K$	$r(5,2)=K$	$r(5,3)=K$	$r(5,4)=K$	$r(5,5)=60$	$r(5,6)=K$	$r(5,7)=K$
$r(6,0)=K$	$r(6,1)=K$	$r(6,2)=K$	$r(6,3)=K$	$r(6,4)=K$	$r(6,5)=K$	$r(6,6)=5$	$r(6,7)=K$
$r(7,0)=K$	$r(7,1)=K$	$r(7,2)=K$	$r(7,3)=5$	$r(7,4)=K$	$r(7,5)=K$	$r(7,6)=K$	$r(7,7)=15$
$r(8,0)=K$	$r(8,1)=K$	$r(8,2)=K$	$r(8,3)=K$	$r(8,4)=K$	$r(8,5)=60$	$r(8,6)=0$	$r(8,7)=K$

with $K = +\infty$.

Manipulation informatique La manipulation informatique va utiliser un objet de type `totalRewardMDP`. Le nombre d'itération va être relié au nombre d'étapes du chemin.

4.6 Modèles à horizon fini et critère total

4.6.1 MarmoteJouet40

Modèle Il s'agit du modèle dit *stochastic inventory model* étudié dans le livre de Puterman. Il est décrit section 3.2 formellement en section 3.2.1 puis un exemple numérique est repris et détaillé section 3.2.2. Enfin section 4.6.1 les différentes étapes de l'algorithme de résolution sont illustrées.

C'est un problème de gestion de stock sur un horizon de temps fini $N = 4$. On a un espace d'état de taille $M = 3$, l'espace d'action est positif et aussi de taille 3.

La matrice de revenu est :

$$r(x,a) = \begin{bmatrix} 0 & -1 & -2 & -5 \\ 5 & 0 & -3 & -\infty \\ 6 & -1 & -\infty & -\infty \\ 5 & -\infty & -\infty & -\infty \end{bmatrix}.$$

Les matrices de transitions (l'indice est la valeur de l'action) sont :

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.75 & 0.25 & 0 & 0 \\ 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \end{bmatrix}, \qquad P_1 = \begin{bmatrix} 0.75 & 0.25 & 0 & 0 \\ 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0.25 & 0.5 & 0.25 \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0.25 & 0.5 & 0.25 \end{bmatrix}, \qquad P_3 = \begin{bmatrix} 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0.25 & 0.5 & 0.25 \\ 0 & 0.25 & 0.5 & 0.25 \end{bmatrix}.$$

Manipulation informatique On manipule ici des MDP à horizon fini et critère total, `finiteHorizonMDP`. La seule méthode qui existe pour ce genre de modèle est `valueIteration`. Les autres sont là par souci de compatibilité et renvoient des politiques non pertinentes.

On peut noter qu'il y a qu'une seule matrice utilisée et que donc le nettoyage doit faire appel aux références qui sont dans le vecteur.

4.7 Modèles à horizon fini et critère discounté

On s'intéresse maintenant à un modèle à horizon fini mais muni d'un facteur de discount.

4.7.1 MarmoteJouet50

Modèle Il s'agit du modèle du cours de MDP à TSP. L'espace d'état est de taille 3 et l'espace d'action de taille 4.

Les matrices de transitions sont

$$P_0 = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 0.25 & 0.25 \\ 0.4 & 0.2 & 0.4 \end{bmatrix}, \qquad P_1 = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.4 & 0.3 & 0.3 \\ 0.5 & 0.4 & 0.1 \end{bmatrix}, P_2 = \begin{bmatrix} 0.25 & 0.55 & 0.2 \\ 0.1 & 0.2 & 0.7 \\ 0.2 & 0.4 & 0.4 \end{bmatrix},$$

$$P_3 = \begin{bmatrix} 0.0 & 0.1 & 0.9 \\ 0.3 & 0.4 & 0.3 \\ 0 & 0 & 1 \end{bmatrix}.$$

La matrice de coût est :

$$R = \begin{bmatrix} 3.025 & 0.225 & 2.07 & 0.77 \\ 4.24 & 1.22 & 1.44 & -0.31 \\ 4.33 & 0.41 & 2.43 & -0.4 \end{bmatrix}.$$

Manipulation informatique On crée tout d'abord les quatre matrices puis la matrice de coût. On crée le MDP `finiteHorizonDiscountedMDP` puis on effectue le calcul de la solution optimale pour un horizon de 4.

Ensuite on veut evaluer le cout d'une politique où dans tous les états l'action est ne rien faire. ce calcul est fait pour un horizon de 6. On contruit une politique dont on remplit les actions à 0 (et rien pour les valeurs ne son pas remplies) et on fait une evaluation de cette politique.

5 Ressources Bibliographiques

Références

[1] W. B. Powell. *Approximate Dynamic Programming : Solving the Curses of Dimensionality.* Wiley, 2007.

[2] M. Puterman. *Markov Decision Processes Discrete Stochastic Dynamic Programming.* Wiley, 2005.