Table des matières

Introduction générale

Depuis toujours, l'optimisation occupe une grande place en économie. En effet, le problème d'optimisation se rencontre fréquemment en économie en particulier dans les modèles de croissance, de gestion de stock, d'exploitations de ressources naturelles ou de l'environnement.

Mais tant qu'on parle d'optimisation, le processus décisionnel de Markov y joue un grand rôle dans un très vaste domaine. La plupart des branches d'économie mathématique comme la théorie des jeux, la théorie de commande utilisent l'approche markovienne dans la modélisation.

Toutefois, la résolution du processus décisionnel de Markov n'est pas évident. Il existe plusieurs méthodes de le résoudre. Nous pouvons formuler en programmation linéaire ce processus, de ce fait nous pouvons le résoudre par la méthode de simplexe. Par ailleurs, des algorithmes de la programmation dynamique comme "Value iteration" et "Policy iteration" permettent également de le résoudre. Ces algorithmes sont les plus utilisés par les informaticiens dans l'intelligence artificielle grâce à leur spécificité. Ainsi dans ce mémoire, nous allons étudier explicitement le Processus décisionnel de Markov et ses algorithmes de résolution.

La suite de ce travail est organisée comme suit :

- le premier chapitre sera une présentation générale des pré-requis associés au processus décisionnel de Markov.
- dans le chapitre 2, nous entrerons en détail dans le cadre théorique du processus décisionnel de Markov. Dans cette partie, nous étudierons chaque critère auquel le processus forme.
- quant au chapitre 3, nous nous intéresserons à la formulation sous forme de programmation linéaire du processus décisionnel de Markov et à la méthode du simplexe.

– pour le chapitre 4, nous allons étudier les algorithmes "Value iteration" et "Policy iteration" que nous allons utiliser pour résoudre le processus décisionnel de Markov.

– le chapitre 5 sera l'illustration de tout ce que nous avons vu précédemment par la mise en oeuvre de ces méthodes pour la résolution d'un processus décisionnel de Markov dans le cadre de l'optimisation du revenu provenant de la production d'une machine.

– Le dernier chapitre sera consacré à la conclusion de notre travail ainsi qu'à la présentation des perspectives que nous aimerions développer par la suite.

Chapitre 1

Généralités

Ce chapitre sera consacré à la présentation des différents pré-requis nécessaires avant d'entrer dans le processus décisionnel de Markov : les processus stochastiques, le processus de Markov, la chaine de Markov et la matrice de transition.

Définition 1.1. *Les processus stochastiques [18] - [9]*

Soient Ω un ensemble fixé, non vide. Soit (Ω,$\mathcal{F}$,$\mathcal{P}$), un espace de probabilité, $\mathcal{T}$ un ensemble arbitraire et S un ensemble fini ou dénombrable.

S est appelé espace d'états.

$\mathcal{T}$ est appelé ensemble des indices. $\mathcal{T}$ peut faire référence au temps, à l'espace ou aux deux à la fois. L'indice $t \in \mathcal{T}$ désigne alors un instant $\mathbb{R}_+$, une date $\mathbb{N}$, un point, ou encore un point à un certain instant.

Un processus stochastique (ou aléatoire) est une famille de variables aléatoires (c'est-à-dire, des applications mesurables) définies sur le même espace de probabilité (Ω,$\mathcal{F}$,$\mathcal{P}$) indexée par $\mathcal{T}$ et à valeurs dans S.

Un processus stochastique est noté par $(X_t)_{t\in\mathcal{T}}$. La valeur de la variable aléatoire X_t en un certain $\omega \in \Omega$ est désignée par $X_t(\omega)$.

Nous allons nous intéresser dans toute la suite pour un processus stochastique X_t à valeur dans $\mathbb{N}$ où t désigne le temps.

Définition 1.2. *Processus de Markov [17] - [8]*

Un processus de Markov (portent le nom de leur découvreur, Andreï Markov) est un processus stochastique possédant la propriété de Markov que nous définissons ci-dessous :

Soient S un espace d'état fini ou dénombrable, $\mathcal{T}$ un ensemble arbitraire et $(X_t)_{(t\in\mathcal{T})}$ un processus stochastique.

Un processus stochastique vérifie la propriété de Markov si pour tout t :

$$Prob(X_t = s_i | X_{t-1} = s_j, X_{t-2} = s_{j_{t-2}}, ..., X_0 = s_{j_0}) = Prob(X_t = s_i | X_{t-1} = s_j)$$

avec $s_i \in S$.

Ce qui peut être explicité de la façon suivante : l'état du système à l'instant t dépend uniquement de l'état à l'instant $t-1$.

Définition 1.3. *Chaîne de Markov et stationnarité [16] - [8]*

Une chaîne de Markov est de manière générale un processus de Markov à temps discret ou un processus de Markov à temps discret et à espace d'états discret. C'est-à-dire $\mathcal{T}$ est un ensemble discret et S est aussi un ensemble discret.
Une chaîne de Markov est dite stationnaire si la probabilité de transition entre états est indépendante du temps, plus formellement si pour tout t et k :

$$\begin{aligned} Prob(X_t = s_i | X_{t-1} = s_j) &= Prob(X_{t+k} = s_i | X_{t+k-1} = s_j) \\ &= Prob(X_1 = s_i | X_0 = s_j) \end{aligned}$$

Définition 1.4. *Probabilité et matrice de transition [19] - [8]*

Soient S un espace d'état fini ou dénombrable, $\mathcal{T}$ un ensemble arbitraire et $(X_t)_{t\in\mathcal{T}}$ un processus stochastique.

Le nombre $Prob(X_t = j | X_{t-1} = i)$ est appelé probabilité de transition de l'état i à l'état j en un pas (une unité de temps), ou bien probabilité de transition de l'état i à l'état j s'il n'y a pas d'ambigüité. Notons le par p_{ij} ou $p(j|i)$.

$$p_{ij} = Prob(X_t = j | X_{t-1} = i) = p(j|i)$$

La famille de nombres $P = (p_{i,j})_{i\in S, j\in S}$ est appelée matrice de transition, noyau de transition, ou opérateur de transition de la chaîne de Markov.

La terminologie matrice de transition est la plus utilisée, mais elle n'est appropriée, en toute rigueur, que lorsque, pour un entier $M \geq 1$, $S = \{0, 1, ..., M\}$ c'est-à-dire S fini et discret. Dans ce cas, la matrice de transition est obtenue d'après le tableau suivant :

état	0	1	2	...	M
0	$p_{0,0}$	$p_{0,1}$	$p_{0,2}$	.	$p_{0,M}$
1	$p_{1,0}$	$p_{1,1}$	$p_{1,2}$	.	$p_{1,M}$
2	$p_{2,0}$	$p_{2,1}$	$p_{2,2}$	.	$p_{2,M}$
.	.	.	.	.	.
M	$p_{M,0}$	$p_{M,1}$	$p_{M,2}$	.	$p_{M,M}$

D'où la matrice de transition de t à t+1 est :

$$\mathcal{P} = \begin{pmatrix} p_{0,0} & p_{0,1} & p_{0,2} & . & p_{0,M} \\ p_{1,0} & p_{1,1} & p_{1,2} & . & p_{1,M} \\ p_{2,0} & p_{2,1} & p_{2,2} & . & p_{2,M} \\ . & . & . & . & . \\ p_{M,0} & p_{M,1} & p_{M,2} & . & p_{M,M} \end{pmatrix}$$

Proposition 1.5. *[2]*

La matrice de transition $P = (p_{i,j})_{(i,j)\in\mathbb{N}\times\mathbb{N}}$ est stochastique : la somme des termes de toute ligne de P est égale à 1.

$$\sum_{j\in S} p(j|i,a) = 1$$

où S l'espace d'états et $a \in A$ (espace des actions).

Définition 1.6. *état récurrent-état transitoire [2] - [8]*

Soit $p_{i,i}$ la probabilité, partant de i et de revenir en i. Nous disons que i est un état récurrent si et seulement si $p_{i,i} = 1$, et sinon c'est un état transitoire.

Chapitre 2

Les Processus Décisionnels de Markov

Cette partie consistera à présenter le cadre théorique des processus de décisions markoviens. En partant de la définition du Processus Décisionnel de Markov, nous allons présenter les différentes familles de politiques et ensuite voir en détail chacun des différents critères utilisés en ce processus.

Définition 2.1. *Processus Décisionnel de Markov [19]*

Les processus décisionnels de Markov ou en anglais "Markovian decision processes" (MDP) sont définis comme des processus stochastiques contrôlés satisfaisant la propriété de Markov, assignant des récompenses aux transitions d'états. Nous les définissons par un quintuplet $(S, A, \mathcal{T}, p, r)$ *où :*

- S est l'espace d'états dans lequel évolue le processus. S peut être fini ou dénombrable et peut être fonction de l'instant t. Dans toute la suite nous supposerons que S fini.

- A est l'espace des actions ou décisions qui contrôlent la dynamique de l'état. De même A peut être fini ou dénombrable. Dans le cas général, l'espace A peut être dépendant de l'état courant (A_s *pour* $s \in S$*), peut être fonction de l'instant t. Dans toute la suite, nous nous limiterons au cas où A est fini.*

- $\mathcal{T}$ *est l'espace des temps, il représente le temps.*

$\mathcal{T}$ *est un ensemble discret, assimilé à un sous ensemble de* $\mathbb{N}$*, qui peut être fini ou infini (on parle d'horizon fini ou d'horizon infini).*

- p() sont les probabilités de transition entre états.

Les probabilités de transition caractérisent la dynamique de l'état du système. Pour une action $a \in A$ fixée, $p(j|i,a)$ ou $p_{i,j}(a)$ désigne alors la probabilité de transition quand le système se déplace d'un état $i \in S$ vers un nouveau état $j \in S$ dans la prochaine période du temps observée quand la décision $a \in A$ est pris.

- $r()$ est la fonction de récompense (ou aussi le gain, le coût, ou encore le revenu) sur les transitions entre états. C'est à dire $r_t(s,a)$ désigne la fonction de récompense quand on a choisi l'action a dans l'état s à l'instant t. r_t peut être considérée comme un gain ou sinon un coût selon le contexte étudié.

Les processus décisionnels de Markov intègrent les concepts d'état qui résument la situation de l'agent à chaque instant, et à chaque action (ou décision) qui influence la dynamique de l'état, de revenu (ou récompense) qui est associé à chacune des transitions d'état. Les MDP sont des chaînes de Markov visitant les états, contrôlées par les actions et valuées par les revenus.

Définition 2.2. *Politique [19]*

Une politique ou stratégie, est une séquence de décisions ou la procédure suivie par l'agent pour choisir à chaque instant l'action à exécuter.

Plus précisement, une politique représente le choix d'une action à effectuer dans un état donné. Nous la noterons par π.

2.1 Les différentes familles de politiques

Soient S l'espace d'états et A l'espace des actions, tous les deux ensembles finis et discrets et l'espace des temps $\mathcal{T}$ avec $\mathcal{T} \subseteq \mathbb{N}$. Nous distinguons quatre familles de politiques :

- la politique histoire-dépendante déterministe

Cette politique se base sur l'historique $h_t = (s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}, s_t)$ et détermine précisément l'action à effectuer.

On la définit comme une fonction :

$$\pi^{HD} : H_t \to A$$
$$h_t \longmapsto a_t$$

où $h_t = (s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}, s_t)$ pour tout $t \in \mathcal{T}$ avec

$$H_t = \{h_t = (s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}, s_t) | (s_k, a_k) \in S \times A; 1 \leq k \leq t-1; s_t \in S\}$$

et

$$S \times A = \{(s, a) | s \in S, a \in A\}$$

C'est-à-dire à chaque historique h_t nous associons une action a_t. Nous notons Π^{HD} l'ensemble des politiques histoire-dépendantes déterministes.

-la politique histoire-dépendante aléatoire

De même, cette politique se base sur l'historique h_t du processus, en outre elle définit une distribution de probabilité selon laquelle on sélectionne une action. On la définit comme une fonction :

$$\begin{aligned} \pi^{HA} : & H_t \times A \to [0, 1] \\ & (h_t, a_t) \longmapsto \pi^{HA}(h_t, a_t) \end{aligned}$$

où $h_t = (s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}, s_t)$ pour tout $t \in \mathcal{T}$ avec

$$H_t = \{h_t = (s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}, s_t) | (s_k, a_k) \in S \times A; 1 \leq k \leq t-1; s_t \in S\}$$

et

$$S \times A = \{(s, a) | s \in S, a \in A\}$$

où $\pi^{HA}(h_t, a_t)$ désigne la probabilité d'effectuer l'action a_t en sachant l'historique h_t.

Nous notons Π^{HA} l'ensemble des politiques histoire-dépendantes aléatoires.

-la politique markovienne déterministe

Comme la politique histoire-dépendante déterministe à la différence que nous ne considérons plus l'historique h_t du processus mais simplement l'état courant du processus. C'est-à-dire à chaque état courant s_t nous associons une action a_t.

Donc, nous pouvons la représenter par une fonction :

$$\begin{aligned} \pi^{MD} : & S \to A \\ & s_t \longmapsto a_t \end{aligned}$$

pour tout $t \in \mathcal{T}$.

Nous notons Π^{MD} l'ensemble des politiques markoviennes déterministes.

-la politique markovienne aléatoire

Comme la politique histoire-dépendante aléatoire à la différence que nous ne considérons plus l'historique du processus mais simplement l'état courant du processus. Donc :

$$\pi^{MA} : S \times A \to [0,1]$$
$$(s_t, a_t) \longmapsto \pi^{MA}(s_t, a_t)$$

où $\pi^{MA}(s_t, a_t)$ désigne la probabilité d'effectuer l'action a_t en sachant l'état courant s_t.

Nous notons Π^{MA} l'ensemble des politiques markoviennes aléatoires.

Nous pouvons résumer dans ce tableau les quatre familles de politiques, comme indiqué sur le tableau suivant :

Politique	Déterministe	Aléatoire
Markovienne	$s_t \longmapsto a_t$	$(a_t, s_t) \longmapsto [0,1]$
Histoire-dépendante	$h_t \longmapsto a_t$	$(a_t, h_t) \longmapsto [0,1]$

Remarque 2.3. *Il existe des relations d'inclusion entre ces 4 familles de politiques :*

$\Pi^{MD} \subseteq \Pi^{HD} \subseteq \Pi^{HA}$

$\Pi^{MD} \subseteq \Pi^{MA} \subseteq \Pi^{HA}$

$\Pi^{HD} \subseteq \Pi^{MA} \subseteq \Pi^{MD}$

En effet, si nous considérons une politique $\pi \in \Pi^{MA}$,$\pi : A \times S \to [0,1]$. Remarquons que si la politique demande $\pi(a_t, s_t) = 0$ ou 1 pour tout $a_t \in A$ et $s_t \in S$, c'est une politique déterministe car nous effectuons l'action ou nous ne l'effectuons pas.

Définition 2.4. *Politique stationnaire [19]*

Une politique est stationnaire si $\pi_t = \pi$ à c'est-à-dire ne dépend pas du temps.

Le processus décisionnel de Markov peut ou non dépendre explicitement du temps. Nous allons noter :

-D^A la famille des politiques markoviennes aléatoires stationnaires

$$D^A = \left\{ \pi^{MA} / \pi^{MA} : (s,a) \in S \times A \to (s,a) = Prob(s|a) \right\}$$

-D la famille des politiques markoviennes déterministes stationnaires

$$D = \left\{\pi^{MD}/\pi^{MD} : s \in S \rightarrow \pi^{MD}(s) \in A\right\}$$

Nous supposerons dans toute la suite dans le cas de l'horizon infini, que le MDP considéré est stationnaire.

2.2 La fonction de valeur

Définition 2.5. *Fonction de valeur [19]*

La fonction de valeur est une fonction qui quantifie la qualité d'une politique. La fonction de valeur permet donc de définir ce qui est une bonne politique.

Soit une politique π fixée, la fonction de valeur est une fonction de S dans $\mathbb{R}$ qu'à tout état $s \in S$ nous associons $V^{\pi}(s) \in \mathbb{R}$.

$$\begin{aligned} V^{\pi} :& S \rightarrow \mathbb{R} \\ & s \mapsto V^{\pi}(s) \end{aligned}$$

$V^{\pi}(s)$ sera définie explicitement plus tard selon le critère étudié. Nous noterons W l'espace des fonctions de S dans $\mathbb{R}$.

Remarque 2.6. *La résolution d'un MDP consiste à déterminer la meilleure politique possible qui spécifie l'action à entreprendre en chacune des étapes pour toutes les situations futures possibles de l'agent de manière optimale.*
L'objectif d'un problème décisionnel de Markov est alors de caractériser et de rechercher s'il existe des politiques $\pi^* \in \Pi^{HA}$ telles que :

$$V^{\pi}(s) \leq V^{\pi^*}(s)$$

pour tout $s \in S$. soit encore

$$\pi^* \in argmax_{\pi \in \Pi^{HA}}(V^{\pi^*})$$

Nous notons alors

$$V^* = \max_{\pi \in \Pi^{HA}} V^{\pi} = V^{\pi^*}$$

la fonction de valeur optimale pour une politique optimale π^*.

2.3 Caractéristique d'une politique markovienne

Soit π une politique markovienne ($\pi \in \Pi^{MA}$). Le processus décrit par l'état $s_t \in S$ vérifie, pour tout $s_0, s_1, ..., s_t, s_{t+1} \in S$

$$\begin{aligned} P^\pi(s_{t+1}|s_0, s_1, ..., s_t) &= \sum_{a\in A} P^\pi(a_t = a|s_0, s_1, ..., s_t) P^\pi(s_{t+1}|s_0, s_1, ..., s_t, a_t = a) \\ &= \sum_{a\in A} \pi(a, s_t) P^\pi(s_{t+1}|s_t, a_t = a) \\ &= P^\pi(s_{t+1}|s_t) \end{aligned}$$

avec $a \in A$ et $\pi(a, s_t) = Prob(action = a \cap état = s_t)$.

Par conséquent, c'est un processus markovien, qui forme une chaîne de Markov dont la matrice de transition notée P_π est définie par :

$$\forall s, s', \quad P_{\pi,s,s'} = P^\pi(s_{t+1} = s'|s_t = s) = \sum_{a\in A} \pi(a, s) p(s'|s, a)$$

Plus explicitement, pour $\pi \in \Pi^{MA}$, si on prend $S = \{0, 1, ..., M\}$

$$P_\pi = \begin{pmatrix} \sum_{a\in A} \pi(a,0)p(0|0,a) & \sum_{a\in A} \pi(a,0)p(1|0,a) & . & \sum_{a\in A} \pi(a,0)p(M|0,a) \\ \sum_{a\in A} \pi(a,1)p(0|1,a) & \sum_{a\in A} \pi(a,1)p(1|1,a) & . & \sum_{a\in A} \pi(a,1)p(M|1,a) \\ . & . & . & . \\ \sum_{a\in A} \pi(a,M)p(0|M,a) & \sum_{a\in A} \pi(a,M)p(1|M,a) & . & \sum_{a\in A} \pi(a,M)p(M|M,a) \end{pmatrix}$$

Pour π une politique markovienne déterministe ($\pi \in \Pi^{MD}$)

$$\forall s, s' \in S, \quad P_{\pi,s,s'} = p(s'|s, \pi(s))$$

Pour $\pi \in \Pi^{MD}$, la matrice de transition P_π est construite simplement en retenant pour chaque état s la ligne correspondante dans la matrice P_a avec $a = \pi(s)$. Plus explicitement, pour $\pi \in \Pi^{MD}$, avec l'espace d'états $S = \{0, 1, ..., M\}$,

$$P_\pi = \begin{pmatrix} p(0|0,\pi(0)) & p(1|0,\pi(0)) & . & p(M|0,\pi(0)) \\ p(0|1,\pi(1)) & p(1|1,\pi(1)) & . & p(M|1,\pi(1)) \\ . & . & . & . \\ p(0|M,\pi(M)) & p(1|M,\pi(M)) & . & p(M|M,\pi(M)) \end{pmatrix}$$

Nous notons r_π le vecteur de composante $r(s, \pi(s))$ pour $\pi \in \Pi^{MD}$ et $\sum_{a\in A} \pi(a, s) r(s, a)$ pour $\pi \in \Pi^{MA}$.

Ainsi, pour $\pi \in \Pi^{MD}$, si on prend $S = \{0, 1, ..., M\}$

$$r_\pi = \begin{pmatrix} r(0, \pi(0)) \\ r(1, \pi(1)) \\ r(2, \pi(2)) \\ . \\ . \\ r(M, \pi(M)) \end{pmatrix}$$

et pour $\pi \in \Pi^{MA}$, avec l'espace d'états $S = \{0, 1, ..., M\}$

$$r_\pi = \begin{pmatrix} \sum_{a \in A} \pi(a, 0) r(0, a) \\ \sum_{a \in A} \pi(a, 1) r(1, a) \\ \sum_{a \in A} \pi(a, 2) r(2, a) \\ . \\ . \\ \sum_{a \in A} \pi(a, M) r(M, a) \end{pmatrix}$$

2.4 Critère d'optimalité

Définition 2.7. *Critère d'optimalité [19]*

Le critère d'optimalité permet de caractériser les politiques qui permettront de générer des séquences de récompenses les plus importantes possibles.

La politique optimale est calculée en fonction de la fonction de gain ou de coût : il s'agit d'optimiser les récompenses possibles ou de minimiser les coûts possibles. Nous évaluons une politique sur la base d'une mesure du cumul espéré des récompenses instantanées le long d'une trajectoire.

Les critères que nous allons étudiés au sein de ce mémoire sont :

- le critère fini,
- le critère total α-pondéré à horizon infini (discounted infinite horizon reward)
- le critère moyen à horizon infini(average-reward criterion).

Remarque 2.8. *Pour ces différents critères, lorsque l'on connait l'état initial (ou une distribution de probabilité sur l'état initial), toute politique histoire-dépendante aléatoire peut être remplacée par une politique markovienne aléatoire ayant la même fonction de valeur. La section suivante nous confirme cette relation.*

2.5 Relation entre les politiques histoires-dépendantes et les politiques markoviennes

Proposition 2.9. *[19]*

Soit $\pi \in \Pi^{HA}$ une politique aléatoire histoire-dépendante. Pour chaque état initial $x \in S$, il existe alors une politique aléatoire markovienne $\pi' \in \Pi^{MA}$ telle que

1. $V_T^{\pi'}(x) = V_T^{\pi}(x)$ *(fonction de valeur pour le critère fini)*
2. $V_\alpha^{\pi'}(x) = V_\alpha^{\pi}(x)$ *(fonction de valeur pour le critère $\alpha-$pondéré)*
3. $\phi^{\pi'}(x) = \phi^{\pi}(x)$ *(fonction de valeur pour le critère moyen)*

Démonstration. Soit $x \in S$, et π une politique aléatoire histoire-dépendante. Soit π' la politique aléatoire markovienne définie à partir de π et x selon :
$\forall t \in \mathcal{T}\,; \forall s \in S, \forall a \in A$

$$\pi'(a_t = a, s_t = s) = P^{\pi}(a_t = a | s_t = s, s_0 = x)$$

Nous avons ainsi $P^{\pi'}(a_t = a | s_t = s) = P^{\pi}(a_t = a | s_t = s, s_0 = x)$.
Montrons par récurrence sur t que

$$P^{\pi}(a_t = a | s_t = s, s_0 = x) = P^{\pi'}(a_t = a | s_t = s, s_0 = x)$$

L'égalité est directe pour $t = 0$.
Pour $t > 0$, en supposant établie la propriété jusqu'à $t-1$, nous avons

$$\begin{aligned} P^{\pi}(s_t = s | s_0 = x) &= \sum_{i \in S} \Big(\sum_{a \in A} P^{\pi}(s_{t-1} = i, a_{t-1} = a | s_0 = x) p(s|i,a) \Big) \\ &= \sum_{i \in S} \Big(\sum_{a \in A} P^{\pi'}(s_{t-1} = i, a_{t-1} = a | s_0 = x) p(s|i,a) \Big) \\ &= P^{\pi'}(s_t = s | s_0 = x) \end{aligned}$$

D'où

$$\begin{aligned} P^{\pi'}(s_t = s, a_t = a | s_0 = x) &= P^{\pi'}(a_t = a | s_t = s) P^{\pi'}(s_t = s | s_0 = x) \\ &= P^{\pi}(a_t = a | s_t = s, s_0 = x) P^{\pi}(s_t = s | s_0 = x) \\ &= P^{\pi}(s_t = s, a_t = a |, s_0 = x) \end{aligned}$$

ce qui établit la récurrence. Nous concluons en remarquant alors que pour tout $x \in S$

$$V_T^\pi(x) = \sum_{t=0}^{T-1} E^\pi[r(s_t, a_t)|s_0 = x]$$

$$V_\alpha^\pi(x) = \sum_{t=0}^{\infty} E^\pi \alpha^t[r(s_t, a_t)|s_0 = x]$$

$$\phi^\pi(x) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} E^\pi\{r(s_t, a_t)|s_0 = x\}$$

(Nous supposons d'abord que ces limites existent, nous allons voir un peu plus tard la démonstration) et

$$E^\pi[r(s_t, a_t)|s_0 = x] = \sum_{s\in S}[\sum_{a\in A} r(s, a)P^\pi(s_t = s, a_t = a|s_0 = x)]$$

□

2.6 Etude des différentes critères

2.6.1 Critère fini

Soit T le nombre d'étapes que l'agent doit effectuer pour contrôler le système avec T fini.

La fonction de valeur pour le critère fini est la fonction qui associe à tout état s l'espérance mathématique de la somme des T prochaines récompenses en suivant la politique π à partir de s.

Si $t \in \{1, ..., T\}$, pour tout $s \in S$

$$V_T^\pi(S) = E^\pi[\sum_{t=1}^{T} r_t|s_1 = s] \tag{2.1}$$

Avec s_1 l'état initial du processus et E^π est l'espérance mathématique sur l'ensemble des réalisations du MDP en suivant la politique π associée à la distribution de probabilité P^π sur l'ensemble de ces réalisations.

Equations d'optimalité pour le critère fini

Théorème 2.10. *[12]*

Soient $x_{T+1}(s) = 0$, avec $s \in S$. Soit pour $t \in \{T, T-1, ..., 1\}$ consécutivement, respectivement une politique déterministe π_t et un vecteur x_t défini par :

$$r_{\pi_t}(s) + P_{\pi_t} x_{t+1}(s) = \max_{a \in A} \{r_t(s,a) + \sum_j p_t(s'|s,a) V_{t+1}(s')\} \text{ pour tout } s \in S$$

et

$$x_t = r_{\pi_t} + P_{\pi_t} x_{t+1}$$

Alors, $\pi^* = (\pi_1, \pi_2, ..., \pi_T)$ est une politique optimale et x_1 est la fonction de valeur optimale V_T^*.

Démonstration. Nous utilisons la récurrence sur T. Soit $\pi = (\pi_1, \pi_2, ..., \pi_T)$ une politique arbitraire.
Pour $T = 1$:

$$\begin{aligned} V_1^{\pi}(s) &= \sum_{a \in A} r_1(s,a)\pi_1(s,a) \\ &\leq \max_{a \in A} r_1(s,a) = x_1(s) \\ &= V_1^{\pi^*}(s), \text{ pour tout } s \in S \end{aligned}$$

Nous supposons que le résultat est vraie pour $T \in \{1, 2, ..., t\}$. Prenons un état arbitraire s. Du proposition 2.9 , il suit qu'il existe une politique Markovienne $\bar{\pi}$ tel que

$$V_{t+1}^{\bar{\pi}}(s) = V_{t+1}^{\pi}(s)$$

Soit $\bar{\pi} = (\bar{\pi}_1, \bar{\pi}_2, ..., \bar{\pi}_(t+1))$.

Définissons la politique Markovienne $\pi' = \pi'_1, \pi'_2, ..., \pi'_t$ par

$$\pi'_k(s,a) = \bar{\pi}_{k+1}(s,a) \text{ pour } k = 1, 2, ..., t.$$

Par l'hypothèse de récurrence, il suit que $V_t^{\pi}(s') \leq x_2(s)$, $s \in S$ car pour un planification de horizon $t+1$ périodes x_2 est la même que x_1 car pour un horizon

de planification de t périodes. Par conséquent,

$$\begin{aligned} V_{t+1}^{\pi}(s') &= V_{t+1}^{\bar{\pi}}(s') \\ &= \sum_{a\in A} \bar{\pi}_1(s,a)\{r_1(s,a) + \sum_{s'\in S} p_1(s'|s,a)V_t^{\pi'}(s')\} \\ &\leq \sum_{a\in A} \bar{\pi}_1(s,a)\{r_1(s,a) + \sum_{s'\in S} p_1(s'|s,a)x_2(s')\} \\ &\leq \max_{a\in A}\{r_1(s,a) + \sum_{s'\in S} p_1(s'|s,a)x_2(s')\} \\ &= x_1(s) \end{aligned}$$

D' autre part

$$\begin{aligned} x_1 &= r_{\pi_1} + P_{\pi_1}x_2 \\ &= r_{\pi_1} + P_{\pi_1}\{r_{\pi_2} + P_{\pi_2}x_3\} \\ &= ... \\ &= \sum_{k=1}^{t+1} P_{\pi_1}P_{\pi_2}...P_{\pi_{k-1}}r_{\pi_k} \\ &= V_{t+1}^{\pi^*} \end{aligned}$$

c'est à dire $V_{t+1}^{\pi^*} = x_1 \geq V_{t+1}^{\pi}$, c'est à dire π^* est une politique optimale et x_1 la fonction de valeur optimale. □

2.6.2 Critère $\alpha-$pondéré à horizon infini

Soit $\pi \in \Pi^{MA}$. La fonction de valeur du critère α-pondéré est celle qui associe à tout état s la limite lorsque N tend vers l'infini de l'espérance en suivant la politique π à partir de s de la somme des N futures prochaines récompenses, pondérées par un facteur d'actualisation α avec $0 < \alpha < 1$.

Pour $0 < \alpha < 1$, pour tout $s \in S$.

$$V_{\alpha}^{\pi}(s) = \lim_{N\to\infty} E^{\pi}[\sum_{t=0}^{N}(\alpha^t r_t|s_0 = s)]$$

Cette limite existe, en effet :

$$\begin{aligned}
V_\alpha^\pi(s) &= \lim_{N\to\infty} E^\pi[\sum_{t=0}^{N}(\alpha^t r_t|s_0=s)] \\
&= \lim_{N\to\infty} \sum_{t=0}^{N} \alpha^t E^\pi[r_t|s_0=s] \\
&= \lim_{N\to\infty} \sum_{k=0}^{N} \alpha^t \sum_{s'\in S}\sum_{a\in A} P^\pi(s_t=s', a_t=a|s_0=s) r(s',a) \\
&= \lim_{N\to\infty} \sum_{k=0}^{N} \alpha^t \sum_{s'\in S}\sum_{a\in A} [\pi(a,s') P^\pi(s_t=s'|s_0=s) r(s',a)] \\
&= \lim_{N\to\infty} \sum_{k=0}^{N} \alpha^t \sum_{s'\in S} P^\pi(s_t=s'|s_0=s) \sum_{a\in A} [\pi(a,s') r(s',a)] \\
&= \lim_{N\to\infty} \sum_{k=0}^{N} \alpha^t \sum_{s'\in S} P^\pi(s_t=s'|s_0=s) r_\pi(s') \\
&= \lim_{N\to\infty} \sum_{k=0}^{N} \alpha^t \sum_{s'\in S} P^t_{\pi,s,s'} r_\pi(s') \\
&= \lim_{N\to\infty} \sum_{k=0}^{N} \alpha^t P^t_\pi r_\pi(s)
\end{aligned}$$

Or pour t suffisamment grand,

$$\max_{s\in S} |\alpha^t P^t_\pi r_\pi(s)| \leq \alpha^t \max_{s\in S} |r_\pi(s)|$$

et comme la série de terme générale α^t converge pour $|\alpha| < 1$,

$$\sum_{t=0}^{\infty} \alpha^t = \frac{1}{1-\alpha} \quad \textit{pour} \ \ |\alpha| < 1$$

Nous obtenons

$$V_\alpha^\pi(s) \leq \max_{s\in S} |r_\pi(s)| \sum_{t=0}^{\infty} \alpha^t = \frac{1}{1-\alpha} \max_{s\in S} |r_\pi(s)|$$

pour $|\alpha| < 1$, d'où

$$V_\alpha^\pi(s) = \lim_{N\to\infty} E^\pi[\sum_{t=0}^{N}(\alpha^t r_t|s_0=s)]$$

existe et

$$V_\alpha^\pi(s) = \sum_{t=0}^{\infty} \alpha^t P^t_\pi r_\pi(s) \tag{2.2}$$

Remarque 2.11. *Le facteur d'actualisation α représente la valeur à la date t d'une unité de récompense reçue à la date $t+1$. Il permet de diminuer l'importance des récompenses lointaines. Ce facteur α a pour principal intérêt d'assurer la convergence de la série en horizon infini.*

Définition 2.12. *opérateur L_π [19]*

Soit π une politique markovienne stationnaire $\pi \in D^A$.

Soit L_π un opérateur de W dans W, espace vectoriel muni de la norme max :

$$\forall V \in W, \|V\| = \max_{s \in S} |V(s)|$$

Pour $\pi \in D^A$, $V \in W$,on définit L_π par :

$$L_\pi V = r_\pi + \alpha P_\pi V$$

avec $0 < \alpha < 1$

Rappelons que pour $\pi \in \Pi^{M^A}$ la matrice de transition notée P_π est définie par

$$\forall s, s' \in S, \quad P_{\pi,s,s'} = P^\pi(s_{t+1} = s' | s_t = s) = \sum_{a \in A} \pi(a, s) p(s' | s, a)$$

Théorème 2.13. *Caractérisation de V_α^π [19]*

Soient $0 < \alpha < 1$, et $\pi \in D^A$ une politique stationnaire markovienne.

Alors V_α^π est solution unique de l'équation $V = L_\pi V$:

$$\forall s \in S, V(s) = r_\pi(s) + \alpha \sum_{s' \in S} P_{\pi,s,s'} V(s') \tag{2.3}$$

et

$$V_\alpha^\pi = (I - \alpha P_\pi)^{-1} r_\pi$$

Démonstration. Soit $0 < \alpha < 1$ et soit V solution de $V = L_\pi V$. Nous avons donc $(I - \alpha P_\pi)V = r_\pi$.

Or la matrice P_π étant stochastique, toutes les valeurs propres de la matrice αP_π sont de modules inférieurs ou égaux à $\alpha < 1$ et donc la matrice $I - \alpha P_\pi$ est inversible avec

$$(I - \alpha P_\pi)^{-1} = \sum_{k=0}^{\infty} \alpha^k P_\pi^k$$

d'où

$$V = (I - \alpha P_\pi)^{-1} r_\pi = \sum_{k=0}^{\infty} \alpha^k P_\pi^k r_\pi$$

or d'après (2.2), pour tout $s \in S$

$$V_\alpha^\pi(s) = \sum_{t=0}^{\infty} \alpha^t P_\pi^t r_\pi(s)$$

Nous avons donc $V = V_\alpha^\pi$.

Pour l'unicité, faisons la démonstration par l'absurde. Supposons alors qu'il existe 2 solutions différentes de l'équation $V = LV$. Soient U_π^α et R_π^α ces solutions avec $U_\pi^\alpha \neq R_\pi^\alpha$, c'est à dire il existe $s \in S$ tel que $U_\pi^\alpha(s) \neq R_\pi^\alpha(s)$. Prenons $U_\pi^\alpha(s) < R_\pi^\alpha(s)$.

Comme U et R sont solutions de l'équation $V = LV$, nous avons $U_\alpha^\pi = (I - \alpha P_\pi)^{-1} r_\pi$ et $R_\alpha^\pi = (I - \alpha P_\pi)^{-1} r_\pi$ donc $U_\alpha^\pi = R_\alpha^\pi$ c'est à dire pour tout $s \in S$, $U_\pi^\alpha(s) = R_\pi^\alpha(s)$. Ce qui contredit l'hypothèse. D'où la solution de l'équation $V = LV$ est unique. □

Equations d'optimalité pour le critère α-pondéré

Définition 2.14. *Opérateur L [3]*

Soit l'opérateur L de l'ensemble des fonctions de valeur W dans lui-même, nommé opérateur de programmation dynamique. Définissons L par $\forall V \in W$, $\forall s \in S$

$$LV(s) = \max_{a \in A} \{r(s,a) + \alpha \sum_{s' \in S} p(s'|s,a) V(s')\}$$

Soit en notation vectorielle

$$\forall V \in W, LV = \max_{\pi \in D} \{r_\pi + \alpha P_\pi V\}$$

Le théorème suivant est le principal théorème concernant l'optimalité des fonctions de valeur pour le critère α-pondéré.

Théorème 2.15. *Equation de Bellman-optimalité des fonctions de valeur pour le critère α-pondéré [3]*

Soit $0 < \alpha < 1$. *Alors* V_α^* *est l'unique solution de l'équation* $V = LV$:

$$\forall s \in S, V(s) = \max_{a \in A} \{r(s,a) + \alpha \sum_{s' \in S} p(s'|s,a) V(s')\} \tag{2.4}$$

Démonstration. Montrons que $\forall V \in W$, et pour $0 < \alpha < 1$,

$$LV = \max_{\pi \in D} \{r_\pi + \alpha P_\pi V\} = \max_{\pi \in D^A} \{r_\pi + \alpha P_\pi V\}$$

Pour cela, considérons une fonction de valeur V et $\delta \in D^A$ une politique stationnaire markovienne. Pour tout s, du fait du caractère positif des $\delta(a,s)$, nous avons :

$$\sum_a \delta(a,s) \{r(s,a) + \sum_{s' \in S} p(s'|s,a) V(s')\}$$

$$\leq \sum_a \delta(a,s) \max_{a'} \{r(s,a') + \sum_{s' \in S} p(s'|s,a')V(s')\}$$

$$\leq \sum_a \delta(a,s) LV(s)$$

$$\leq LV(s)$$

Ainsi pour tout $\delta \in D^A$

$$r_\delta + \pi P_\delta V \leq \max_{\pi \in D}[r_\pi + \alpha P_\pi V]$$

soit

$$\max_{\delta \in D^A}[r_\delta + \pi P_\delta V] \leq \max_{\pi \in D}[r_\pi + \alpha P_\pi V]$$

L'inégalité inverse est immédiate car $D \subseteq D^A$.

Montrons alors que $\forall V, V \geq LV$. Nous avons donc

$$V \geq \max_{\pi \in D}[r_\pi + \alpha P_\pi V]$$

Soit $\pi = (\pi_0, \pi_1, ...) \in \Pi^{MA}$. Pour tout $t \in \mathcal{T}, \pi_t \in D^A$, d'où

$$V \geq r_{\pi_0} + \alpha P_{\pi_0} V \geq r_{\pi_0} + \alpha P_{\pi_0}(r_{\pi_1} + \alpha P_{\pi_1} V)$$

$$\geq r_{\pi_0} + \alpha P_{\pi_0} r_{\pi_1} + \alpha^2 P_{\pi_0} P_{\pi_1} r_{\pi_2} + ... + \alpha^{n-1} P_{\pi_0} ... P_{\pi_{n-2}} r_{\pi_{n-1}} + \alpha^n P_{\pi_n} V$$

Nous avons donc

$$V - V_\alpha^\pi \geq \alpha^n P_\pi^n V - \sum_{k=n}^{\infty} \alpha^k P_\pi^k r_{\pi_k}$$

car

$$V_\alpha^\pi = \sum_{k=0}^{\infty} \alpha^k P_\pi^k r_{\pi_k}$$

avec

$$P_\pi^k = P_{\pi_0} P_{\pi_1} ... P_{\pi_{k-1}}$$

Les deux termes de droite peuvent être rendus aussi petits que l'on veut, pour n suffisamment grand, car

$$\|\alpha^n P_\pi^n V\| \leq \alpha^n \|V\|$$

et

$$\| \sum_{k=n}^{\infty} \alpha^k P_\pi^k r_{\pi_k} \| \leq \sum_{k=n}^{\infty} \alpha^k R \leq \frac{\alpha^n}{1-\alpha} R$$

avec

$$R = \max_{s,a} r(s,a)$$

Nous en déduisons

$$V - V_\alpha^\pi \geq 0$$

Cela tant vrai pour toute politique $\pi \in \Pi^{MA}$, nous avons donc

$$V \geq \max_{\pi \in \Pi^{MA}} V_\alpha^\pi = \max_{\pi \in \Pi^{HA}} V_\alpha^\pi = V_\alpha^*$$

D'où

$$V \geq V_\alpha^*$$

Inversement soit V telle que $V \leq LV$. Nous avons donc

$$V \leq \max_{\pi \in D} [r_\pi + \alpha P_\pi V]$$

Supposons ce max atteint en π^*. Nous avons donc

$$V \leq r_{\pi^*} + \alpha P_{\pi^*} V$$

$$\leq r_{\pi^*} + \alpha P_{\pi^*}(r_{\pi^*} + \alpha P_{\pi^*} V)$$

$$\leq r_{\pi^*} + \alpha P_{\pi^*} r_{\pi^*} + ... + \alpha^{n-1} P_{\pi^*}^{n-1} r_{\pi^*} + \alpha^n P_{\pi^*}^n V$$

$$V - V_\alpha^{\pi^*} \leq -\sum_{k=n}^{\infty} \alpha^k P_{\pi^*}^k V$$

Les termes de droite pouvant être rendus aussi proches de 0 que désiré, nous avons donc

$$V - V_\alpha^{\pi^*} \leq 0$$

Soit

$$V \leq V_\alpha^{\pi^*} \leq V_\alpha^*$$

Nous avons ainsi montré que

$$V \geq LV \Rightarrow V \geq V_\alpha^*$$

$$V \leq LV \Rightarrow V \leq V_\alpha^*$$

ce qui implique que $V = LV \Rightarrow V = V_\alpha^*$: toute solution de l'équation $LV = V$ est nécessairement égale à la fonction de valeur optimale V_α^*. □

Montrons maintenant qu'une telle solution existe.

Rappelons pour cela le théorème du point fixe de Banach :

Théorème 2.16. *Théorème du point fixe de Banach [1]*

Soient U un espace de Banach (c'est à dire : espace vectoriel normé complet) et K une contraction sur U, (c'est-à-dire $\forall u, v \in U \|Ku - Kv\| \leq \alpha \|u - v\|$ pour tout $0 \leq \alpha < 1$. Alors :

1) Il existe un unique $u^* \in U$ tel que $Ku^* = u^*$;

2) Pour tout $u_n \in U$, la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_{n+1} = Ku_n = K^{n+1}u_0$ converge vers u^*.

L'espace W muni de la norme max est un espace vectoriel normé de dimension fini donc complet. Il suffit donc de montrer que l'opérateur L est une contraction pour cette norme.

Proposition 2.17. *[19]*

Soit $0 < \alpha < 1$. L'opérateur de programmation dynamique L défini par

$$LV = \max_{\pi \in D} [r_\pi + \alpha P_\pi V]$$

est une contraction sur V.

Démonstration. Soient U et V dans W et $s \in S$. Supposons $LV(s) \geq LU(s)$.
Soit

$$a_s^* \in \arg\max_{a\in A} \{r(s,a) + \alpha \sum_{s'\in S} p(s'|s,a)V(s')\}$$

$$0 \leq |LV(s) - LU(s)| = LV(s) - LU(s)$$

$$\leq r(s,a_s^*) + \alpha \sum_{s'\in S} p(s'|s,a_s^*)V(s') - [r(s,a_s^*) + \alpha \sum_{s'\in S} p(s'|s,a_s^*)U(s')]$$

$$\leq \alpha \sum_{s'\in S} p(s'|s,a_s^*)[V(s') - U(s')]$$

$$\leq \alpha \sum_{s'\in S} p(s'|s,a_s^*)\|V - U\|$$

$$\leq \alpha \|V - U\|$$

D'où

$$\|LV - LU\| = \max_s |LV(s) - LU(s)| \leq \alpha \|V - U\|$$

□

Cette propriété de contraction assure donc l'existence pour l'opérateur L d'un point fixe unique qui est donc égal à V_α^*.
Nous allons terminer par le théorème suivant la caractérisation des politiques optimale pour le critère α-pondéré.

Théorème 2.18. *Caractérisation des politiques optimales [19]*

Soit $0 < \alpha < 1$. *Alors*

1) $\pi^* \in \Pi^{HA}$ *est optimale* $\Longleftrightarrow V_\alpha^{\pi^*}$ *est solution de* $LV = V$ *et* $V_\alpha^{\pi^*} = V_\alpha^*$ *;*

2) toute politique stationnaire π^* *définie par*

$$\pi^* \in \arg\max_{\pi \in D}[r_\pi + \alpha P_\pi V_\alpha^*]$$

est une politique optimale.

Démonstration. La première équivalence est évidente du fait du théorème précédent. Soit alors $\pi^* \in \arg\max_{\pi \in D}[r_\pi + \alpha P_\pi V_\alpha^*]$, nous avons alors

$$L_{\pi^*} V_\alpha^* = r_{\pi^*} + \alpha P_{\pi^*} V_\alpha^*$$

$$\max_{\pi \in D}\{r_\pi + \alpha P_\pi V_\alpha^*\} = LV_\alpha^* = V_\alpha^*$$

L'unicité de la solution de $V = L_{\pi^*} V$ démontrée par le théorème 2.13 permet de déduire que $V_\alpha^* = V_\alpha^{\pi^*}$ et donc que π^* est optimale. □

2.6.3 Critère moyen : average-reward criterion

Lorsque la fréquence des décisions est importante, avec un facteur d'actualisation proche de 1, ou lorsqu'il n'est pas possible de donner une valeur économique aux récompenses, il est préférable de considérer un critère qui représente la moyenne des récompenses espérées et non plus leur somme pondérée.

Dans le critère de la récompense moyenne, on considère le comportement de la limite de $\frac{1}{T}\sum_{t=1}^{T} r(s_t, a_t)$ quand $T \to \infty$. Puisque cette limite peut ne pas exister et nous ne pouvons pas échanger la limite et l'espérance en général, nous considérons alors quatre autres mesures différentes d'évaluation :

1. La limite inférieure de la récompense moyenne prévue :

$$\phi^\pi(s) = \liminf_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} E^\pi\{r(s_t, a_t) | s_0 = s\}$$

avec la fonction de valeur optimale

$$\phi = \max_\pi \phi^\pi$$

2. La limite supérieure de la récompense moyenne prévue :

$$\bar{\phi}^{\pi}(s) = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} E^{\pi}\{r(s_t, a_t)|s_0 = s\}$$

avec la fonction de valeur optimale

$$\bar{\phi} = \max_{\pi} \bar{\phi}^{\pi}$$

3. L'espérance de la limite inférieure de la récompense moyenne :

$$\psi^{\pi}(s) = E^{\pi}[\liminf_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \{r(s_t, a_t)|s_0 = s\}]$$

avec la fonction de valeur optimale

$$\psi = \max_{\pi} \psi^{\pi}$$

4. L'espérance de la limite supérieure de la récompense moyenne :

$$\bar{\psi}^{\pi}(s) = E^{\pi}[\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \{r(s_t, a_t)|s_0 = s\}]$$

avec la fonction de valeur optimale

$$\bar{\psi} = \max_{\pi} \bar{\psi}^{\pi}$$

Lemme 2.19. *[12]*

$$\psi^{\pi} \leq \phi^{\pi} \leq \bar{\phi}^{\pi} \leq \bar{\psi}^{\pi}$$

Démonstration. La deuxième inégalité est évidente. La première et dernière inégalité suivent du lemme de Fatou :

$$\begin{aligned}\psi^{\pi}(s) &= E^{\pi}[\liminf_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \{r(s_t, a_t)|s_0 = s\}] \\ &\leq \liminf_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} E^{\pi}\{r(s_t, a_t)|s_0 = s\} = \phi^{\pi}(s)\end{aligned}$$

et

$$\begin{aligned}\bar{\phi}^{\pi}(s) &= \limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} E^{\pi}\{r(s_t, a_t)|s_0 = s\} \\ &\leq E^{\pi}[\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \{r(s_t, a_t)|s_0 = s\}] = \bar{\psi}^{\pi}(s)\end{aligned}$$

□

Ce lemme nous montre que les quatre mesures d'évaluation sont équivalentes dans le sens que la politique déterministe optimale pour un critère est également optimale pour les autres critères. Nous emploierons le critère 1, la limite inférieure de la récompense moyenne prévue dans toute la suite.

Dans cette section nous commencerons par voir respectivement la matrice stationnaire, la matrice fondamentale et la matrice de déviation d'une chaîne de Markov. Ces matrices jouent un rôle important dans le critère moyen à horizon infini. L'expansion de la série de Laurent nous sert à relier la récompense moyenne à la récompense pondérée. Nous terminons par l'équation d'optimalité dans le cas général de MDP pour le critère moyen.

La matrice stationnaire

Considérons une politique $\pi \in \Pi^{MD}$. Dans le MDP de la récompense moyenne le comportement de la limite de P_π^n quand n tend vers l'infini joue un rôle très important. En général, $\lim_{n\to\infty} P_\pi^n$ n'existe pas. Par conséquent, nous allons considérer d'autres types de convergence.

Soit une suite $\{b_n\}_{n=0}^{\infty}$. Cette suite converge au sens de Cesaro vers si

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} b_k$$

existe et égale à b.

Nous notons cette convergence par : $\lim_{n\to\infty} b_n =_c b$ ou $b_n \to_c b$

La suite converge au sens d'Abel vers b si

$$\lim_{\alpha\uparrow 1}(1-\alpha) \sum_{n=0}^{\infty} \alpha^n b_n$$

existe et égale à b.

Nous notons cette convergence par : $\lim_{n\to\infty} b_n =_a b$ ou $b_n \to_a b$

Théorème 2.20. *[12]*

Si la suite $\{b_n\}_{n=0}^{\infty}$ converge au sens de Cesaro vers b, alors $\{b_n\}_{n=0}^{\infty}$ converge au sens d'Abel vers b.

Théorème 2.21. *[12]*

Soit P une matrice stochastique, c'est à dire une matrice d'une chaîne de Markov. Alors,

1. $P^* = \lim_{n\to\infty} \frac{1}{n}\sum_{k=0}^{n-1} P^k$ *existe c'est à dire* $P^k \to_c P^*$
2. $P^*P = PP^* = P^*P^* = P$

Démonstration. Soit $B^{(n)} = \frac{1}{n}\sum_{k=0}^{n-1} P^k$.

Puisque P^k est stochastique pour tout k , $B^{(n)}$ est aussi une matrice stochastique. D'où, la série $\{B^{(n)}\}_{n=1}^{\infty}$ est bornée. Par conséquent, chaque sous-suite infini de $\{B^{(n)}\}_{n=1}^{\infty}$ a un point d'accumulation. En outre, nous avons

$$\begin{aligned}
B^{(n)} + \frac{1}{n}(P^n - I) &= \frac{1}{n}\sum_{k=0}^{n-1} P^k + \frac{1}{n}(P^n - I) \\
&= \frac{1}{n}\sum_{k=0}^{n-1} P^k + \frac{1}{n}P^n - \frac{1}{n}I = \frac{1}{n}(\sum_{k=0}^{n-1} P^k + P^n) - \frac{1}{n}I \\
&= \frac{1}{n}(\sum_{k=0}^{n} P^k) - \frac{1}{n}I = \frac{1}{n}P^0 + \frac{1}{n}(\sum_{k=1}^{n} P^k) - \frac{1}{n}I \\
&= \frac{1}{n}I + \frac{1}{n}(\sum_{k=1}^{n} P^k) - \frac{1}{n}I = \frac{1}{n}\sum_{k=1}^{n} P^k \\
&= \frac{1}{n}\sum_{k=0}^{n-1} P^{k+1} = \frac{1}{n}(\sum_{k=0}^{n-1} P^k)P = B^{(n)}P \\
&= \frac{1}{n}\sum_{k=0}^{n-1} P^{k+1} = P(\frac{1}{n}\sum_{k=0}^{n-1} P^k) = PB^{(n)}
\end{aligned}$$

donc

$$B^{(n)} + \frac{1}{n}(P^n - I) = B^{(n)}P = PB^{(n)}, n \in \mathbb{N} \tag{2.5}$$

Soit $J = \lim_{k\to\infty} B^{(n_k)}$, où $\{B^{(n_k)}\}_{k=1}^{\infty}$ est une sous-suite convergente de $\{B^{(n)}\}_{n=1}^{\infty}$. De (2.5) nous obtenons

$$J = JP = PJ \tag{2.6}$$

Soit $\{B^{(m_k)}\}_{k=1}^{\infty}$ une sous-suite convergente également de $\{B^{(n)}\}_{n=1}^{\infty}$ avec la matrice de limite K. De (2.5) il suit également que :

$$K = KP = PK \tag{2.7}$$

D'où $J = P^nJ = JP^n$ et $K = P^nK = KP^n$ pour chaque n.

Par conséquent,$J = B^{(n)}J = JB^{(n)}$ et $K = B^{(n)}K = KB^{(n)}$ pour chaque n,

implique que : $J = KJ = JK$ et $K = JK = KJ$ ie $J = K$.
La suite $\{B^{(n)}\}_{n=1}^{\infty}$ a exactement un point d'accumulation, c'est à dire :

$$P^* = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} P^k$$

existe et est la limite de Cesaro de la suite $\{P^n\}_{n=1}^{\infty}$ En outre, nous avons montré cela

$$P^*P = PP^* = P^*P^* = P^*$$

□

Définition 2.22. *Matrice stationaire [12]*

La matrice P^* est appelée la matrice stationaire de la matrice stochastique P.

Corollaire 2.23. *[12]*

$$\lim_{n\uparrow 1} \sum_{n=0}^{\infty} \alpha^n (P^n - P^*) = 0$$

Démonstration. Puisque P^n converge au sens de Cesaro vers P^*, $P^n - P^*$ converge au sens de Cesaro vers 0, et par conséquent converge au sens d'Abel vers 0, c'est à dire :

$$\lim_{n\uparrow 1} \sum_{n=0}^{\infty} \alpha^n (P^n - P^*) = 0$$

□

La matrice fondamentale et la matrice de la déviation

Théorème 2.24. *[12]*

Soit P une matrice stochastique arbitraire. Alors, $I - P + P^*$ est non singulier et $Z = (I - P + P^*)^{-1}$ satisfait

$$Z = \lim_{n\to\infty} \sum_{i=1}^{n} \sum_{k=0}^{i-1} (P - P^*)^k$$

Démonstration. Puisque $P^*P = PP^* = P^*P^* = P^*$, il suit, par la récurrence sur n, que $(P - P^*)^n = P^n - P^*$, pour $n \in \mathbb{N}$.
En effet, pour n=1, nous avons :
$(P - P^*)^1 = P - P^* = P^1 - P^*$.
En supposons que l'hypothèse est vrai pour $n = k$ c'est à dire $(P - P^*)^k = P^k - P^*$,

montrons le pour $n = k + 1$ c'est à dire $(P - P^*)^{k+1} = P^{k+1} - P^*$.

$$\begin{aligned}(P - P^*)^{k+1} &= (P - P^*)^k (P - P^*) \\ &= (P^k - P^*)(P - P^*) \\ &= P^k P - P^k P^* - P^* P + P^* P^* \\ &= P^{k+1} - P^* - P^* + P^* \\ &= P^{k+1} - P^*\end{aligned}$$

Soit $B = P - P^*$. Puisque

$$I - B^i = (I - B)(I + B + ... + B^{i-1}) \tag{2.8}$$

nous avons , en faisant la moyenne de (2.8),

$$\frac{1}{n}\sum_{i=1}^{n}(I - B^i) = I - \frac{1}{n}\sum_{i=1}^{n} B^i = (I - B)\frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{i-1} B^k \tag{2.9}$$

Puisque

$$\frac{1}{n}\sum_{i=1}^{n} B^i = \frac{1}{n}\sum_{i=1}^{n}(P^i - P^*) = \frac{1}{n}\sum_{i=1}^{n} P^i - P^*$$

nous obtenons

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n} B^i = \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n} P^i - P^* = P^* - P^* = 0$$

c'est à dire $I - B = I - P + P^*$ est non singulier et

$$Z = \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{i-1}(P - P^*)^k$$

□

Définition 2.25. *Matrice fondamentale [12]*

La matrice $Z = (I - P + P^*)^{-1}$ est appelée la matrice fondamentale de P.

Définition 2.26. *Matrice de déviation [12]*

La matrice de déviation D est définie par :

$$D = Z - P^* = \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{i-1}(P - P^*)^k - P^*$$

Théorème 2.27. *[12]*

La matrice de déviation D satisfait

1.

$$D = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{i-1} (P^k - P^*)$$

2.

$$P^*D = DP^* = (I - P)D + P^* - I = D(I - P) + P^* - I = 0$$

.

Démonstration. (1) Puisque $(P-P^*)^k = (P^k-P^*)$ pour $k = 1, 2, ...$, nous obtenons

$$\sum_{i=1}^{n} \sum_{k=0}^{i-1} (P - P^*)^k = n.I + \sum_{i=2}^{n} \sum_{k=1}^{i-1} (P - P^*)^k = n.I + \sum_{i=2}^{n} \sum_{k=1}^{i-1} (P^k - P^*)$$

et

$$\sum_{i=1}^{n} \sum_{k=0}^{i-1} (P^k - P^*) = n.(I - P^*) + \sum_{i=2}^{n} \sum_{k=1}^{i-1} (P^k - P^*)$$

Par conséquent,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{i-1} (P^k - P^*) = \lim_{n\to\infty} \frac{1}{n} [n.(I - P^*) + \sum_{i=2}^{n} \sum_{k=1}^{i-1} (P - P^*)^k]$$

$$= Z - P^*$$

(2)

$$P^*D = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{i-1} P^*(P^k - P^*) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{i-1} P^*(P^* - P^*) = 0$$

$$\begin{aligned}
(I - P)D &= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{i-1} (I - P)(P^k - P^*) \\
&= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \sum_{k=0}^{i-1} (P^k - P^{k+1}) \\
&= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (I - P^k) \\
&= I - P^*.
\end{aligned}$$

□

Dans les 2 théorèmes précédents, la matrice fondamentale Z et la matrice de déviation D sont exprimées comme limites de Cesaro. Ces matrices peuvent également être exprimées en forme abélienne, le prochain théorème nous le montre.

Théorème 2.28. *[12]*

1. $Z = \lim_{\alpha\uparrow 1} \sum_{n=0}^{\infty} \alpha^n (P - P^*)^n$.

2. $D = \lim_{\alpha\uparrow 1} \sum_{n=0}^{\infty} \alpha^n (P^n - P^*)$.

Démonstration. (1) Comme $(I - Q)^{-1} = \sum_{n=0}^{\infty} Q^n$ pour $\|Q\| < 1$, nous avons

$$H(\alpha) = \sum_{n=0}^{\infty} [\alpha(P - P^*)]^n = [I - \alpha(P - P^*)]^{-1}$$

Par conséquent

$$I = H(\alpha)[I - \alpha(P - P^*)] = H(\alpha)(I - P + P^*) + (1 - \alpha)H(\alpha)(P - P^*)$$

Puisque $P^n - P^*$ converge au sens de Cesaro vers 0,$P^n - P^*$ converge aussi au sens de Abel vers 0, c'est à dire

$$\lim_{n\to 1} (1 - \alpha)H(\alpha) = 0.$$

Par conséquent,

$$Z = (I - P + P^*)^{-1} = \lim_{\alpha\uparrow 1} \sum_{n=0}^{\infty} \alpha^n (P - P^*)^n.$$

(2) Puisque

$$\begin{aligned} \sum_{n=0}^{\infty} \alpha^n (P^n - P^*) &= I - P^* + \sum_{n=1}^{\infty} \alpha^n (P^n - P^*) \\ &= I - P^* + \sum_{n=1}^{\infty} \alpha^n (P - P^*)^n \\ &= \sum_{n=0}^{\infty} \alpha^n (P - P^*)^n - P^* \end{aligned}$$

nous avons alors

$$\begin{aligned} \lim_{\alpha\uparrow 1} \sum_{n=0}^{\infty} \alpha^n (P^n - P^*) &= \lim_{\alpha\uparrow 1} \sum_{n=0}^{\infty} \alpha^n (P - P^*)^n - P^* \\ &= Z - P^* \\ &= D \end{aligned}$$

□

Relation entre les trois critères

Le théorème suivant donne la relation entre les récompenses moyennes (au-dessus d'un horizon infini), les récompenses pondérées (au-dessus d'un horizon infini) et les récompenses totales au-dessus d'un horizon fini.

Théorème 2.29. *[12]*

Soit π une politique déterministe. Alors,

1. $\phi^\pi = P_\pi^* r_\pi$
2. $\phi^\pi = \lim_{\alpha\uparrow 1}(1-\alpha)V_\alpha^\pi$
3. $V_T^\pi = T\phi^\pi + D_\pi r_\pi - P_\pi^T D_\pi r_\pi$

Démonstration. (1)

$$\phi^\pi = \liminf_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} P_\pi^t r_\pi = P_\pi^* r_\pi$$

(2) Puisque P^* la limite de Cesaro de P^t, elle est aussi la limite d'Abel , c'est à dire :

$$\phi^\pi = P_\pi^* r_\pi = \lim_{\alpha\uparrow 1}(1-\alpha)\sum_{t=0}^{\infty}[\alpha P_\pi]^t r_\pi = V_\alpha^\pi$$

(3) Nous appliquons la récurrence sur T.

Pour T=1 :

$\phi^\pi + D_\pi r_\pi - P_\pi D_\pi r_\pi = [P_\pi^* + [I - P_\pi]D_\pi]r_\pi = r_\pi = V_1^\pi$ car $P_\pi^* + [I - P_\pi]D_\pi = I$ voir la partie(2)du théorème 2.27

Supposons que cela est vraie pour les périodes T. Alors, nous pouvons écrire :

$$\begin{aligned}
(T+1)\phi^\pi + D_\pi r_\pi - P_\pi^{T+1} D_\pi r_\pi &= T\phi^\pi + P_\pi^* r_\pi + D_\pi r_\pi - P_\pi^{T+1} D_\pi r_\pi \\
&= V_T^\pi + P_\pi^T D_\pi r_\pi + P_\pi^* - P_\pi^{T+1} D_\pi r_\pi \\
&= V_T^\pi + P_\pi^T [D_\pi + P_\pi^* - P_\pi D_\pi] r_\pi \\
&\quad (\textit{en utilisant la partie}(2)\ \textit{du théorème}2.27) \\
&= V_T^\pi + P_\pi^T r_\pi \\
&= V_{T+1}^\pi
\end{aligned}$$

□

L'expansion de la série de Laurent

La partie (2) du théorème 2.29 montre une relation entre la récompense pondérée et celle du moyenne quand le facteur d'escompte tend à 1. Cette relation est basée sur l'expansion de Laurent de V_π^α près de $\alpha = 1$, avec une politique déterministe comme exprimée dans le prochain théorème.

Théorème 2.30. *[12]*

Soit π une politique déterministe. Soit u_k^π, $k = -1, 0, ...$ définie par : $u_{-1}^\pi = P_\pi^* r_\pi$, $u_0^\pi = D_\pi r_\pi$ et $u_{k+1}^\pi = -D_\pi u_k^\pi$, $k \geq 0$.

Alors

$$\alpha V_\alpha^\pi = \sum_{k=-1}^{\infty} \rho^k u_k^\pi$$

pour $\alpha_{0_\pi} < \alpha < 1$, où $\rho = \frac{1-\alpha}{\alpha}$ et $\alpha_{0_\pi} = \frac{\|D_\pi\|}{1+\|D_\pi\|}$.

Démonstration. Soit

$$x_\pi = \frac{1}{\alpha} \sum_{k=-1}^{\infty} \rho^k u_k^\pi = \frac{\phi^\pi}{1-\alpha} + \frac{1}{\alpha} \sum_{k=0}^{\infty} \rho^k u_k^\pi$$

Puisque $u_k^\pi = D_\pi[-D_\pi]^k r_\pi$, pour $k > 0$, la série$\sum_{k=0}^{\infty} \rho^k u_k^\pi$ est bien définie si $\rho\|D_\pi\| < 1$ c'est à dire $\alpha = \frac{\|D_\pi\|}{1+\|D_\pi\|}$.

Puisque V_α^π est l'unique solution du système linéaire $\{I - \alpha P_\pi\}x = r_\pi$, il suffit de montrer que :

$\{I - \alpha P_\pi\}x_\pi = r_\pi$ c'est à dire $y_\pi := r_\pi - \{I - \alpha P_\pi\}x_\pi = 0$ Nous avons

$$\begin{aligned}
y_\pi &= r_\pi - \{I - \alpha P_\pi\}\frac{P_\pi^* r_\pi}{1-\alpha} - \{I - \alpha P_\pi\}\frac{D_\pi}{\alpha}\sum_{k=0}^{\infty}[-\rho D_\pi]^k r_\pi \\
&= r_\pi - P_\pi^* r_\pi - \{\alpha(I - P_\pi) + (1-\alpha)I\}\frac{D_\pi}{\alpha}\sum_{k=0}^{\infty}[-\rho D_\pi]^k r_\pi \\
&= (I - P_\pi^*)r_\pi - (I - P_\pi)D_\pi\sum_{k=0}^{\infty}[-\rho D_\pi]^k r_\pi - \frac{1-\alpha}{\alpha}D_\pi\sum_{k=0}^{\infty}[-\rho D_\pi]^k r_\pi \\
&= (I - P_\pi^*)r_\pi - \{I - P_\pi^*\}\sum_{k=0}^{\infty}[-\rho D_\pi]^k r_\pi + \sum_{k=0}^{\infty}[-\rho D_\pi]^{k+1} r_\pi \\
&= (I - P_\pi^*)r_\pi - \sum_{k=0}^{\infty}[-\rho D_\pi]^k r_\pi + P_\pi^* r_\pi + \sum_{k=1}^{\infty}[-\rho D_\pi]^k r_\pi \\
&= (I - P_\pi^*)r_\pi - r_\pi - \sum_{k=1}^{\infty}[-\rho D_\pi]^k r_\pi + P_\pi^* r_\pi + \sum_{k=1}^{\infty}[-\rho D_\pi]^k r_\pi = 0
\end{aligned}$$

□

Corollaire 2.31. *[12]*

Soit π une politique déterministe.

$$V_\alpha^\pi = \frac{\phi^\pi}{1-\alpha} + u_0^\pi + \epsilon(\alpha)$$

où $\epsilon(\alpha)$ satisfait :

$$\lim_{\alpha \to 1} \epsilon(\alpha) = 0$$

Démonstration. Du Théorème (2.30), il suit que

$$V_\alpha^\pi = \frac{\phi^\pi}{1-\alpha} + \frac{u_0^\pi}{\alpha} + \sum_{k=1}^{\infty} \frac{(1-\alpha)^k}{\alpha^{k+1}} u_k^\pi$$

Puisque

$$\frac{1}{\alpha} = \frac{1}{1-(1-\alpha)} = 1 + (1-\alpha) + (1-\alpha)^2 + ...$$

Nous pouvons écrire

$$V_\alpha^\pi = \frac{\phi^\pi}{1-\alpha} + u_0^\pi + \epsilon(\alpha)$$

où

$$\lim_{\alpha \to 1} \epsilon(\alpha) = 0$$

□

Equation d'optimalité pour le critère moyen

Dans le critère α-pondéré, la fonction de valeur optimale est la solution unique d'une équation d'optimalité. Pour le critère moyen, nous pouvons avoir un résultat semblable mais l'équation est plus compliquée.

Théorème 2.32. *Equation d'optimalité [12]*

Considérons le système :

$$\begin{cases} x(s) = \max_{a \in A_s} \sum_{s' \in S} p(s'|s,a) x(s') \\ x(s) + y(s) = \max_{a \in A(s,x)} \{ r(s,a) + \sum_{s' \in S} p(s'|s,a) y(s') \end{cases} \tag{2.10}$$

pour tout $s \in S$

avec $A(s,x) = \{a \in A_s | x(s) = \sum_{s' \in S} p(s'|s,a) x(s')\}$, $s \in S$.

Si (x,y) est une solution de (2.10), alors $x = \phi$ est la fonction de valeur optimale.

Démonstration. Soit (x, y) la solution de (2.10). Alors, pour un $\pi \in \Pi^{MD}$, $x \geq P_\pi x$ implique que $x \geq P_\pi^n x$ pour tout $n \in \mathbb{N}$, et par conséquent $x \geq P_\pi^* x$.
En outre, puisque $0 = P_\pi^*\{x - P_\pi\}$ et tous les éléments de P_π^* et $x - P_\pi$ sont non négatives, $p_\pi^*(s'|s)(x - P_\pi x)_{s'} = 0$ pour tout $(s, s') \in S \times S$. cela implique que $p_\pi^*(s|s)(x - P_\pi x)_s = 0$ pour tout $s \in S$.

Pour un état récurrent i, $p_\pi^*(s|s) > 0$, et par conséquent,

$$x(s) - \sum_{s' \in S} p_\pi(s'|s)x(s') = 0 \text{ c'est à dire } \pi(s) \in A(s, x)$$

et donc, par (2.10)

$$x(s) + y(s) \geq r_\pi(s) + \sum_{s' \in S} p_\pi(s'|s)y(s')$$

Les colonnes de P_π^* correspondant aux états transitoires sont zéro, cela implique

$$P_\pi^*(x + y) \geq P_\pi^*\{r_\pi + P_\pi y\} = \phi_\pi + P_\pi^* y$$

c'est à dire

$$\phi_\pi \leq P_\pi^* x \leq x \tag{2.11}$$

D'autre part, toute solution de système (2.10) donne une politique R laquelle satisfait

$$x = P_R x \text{ et } x + y = r_R + P_R y$$

Par conséquent, $x = P_R^* x$ et donc,

$$\phi_R = P_R^* r_R = P_R^*\{x + y - P_R y\} = x + P_R^*\{y - P_R y\} = x \tag{2.12}$$

Par (2.11) et (2.12) il suit que

$$x(s) = \max_{a \in A_s} \sum_{s' \in S} p(s'|s, a)x(s') = \phi(s) \ , \ s \in S.$$

□

Conclusion

Pour chacun des critères, nous avons ainsi déterminer la spécificité de chaque équation d'optimalité portant sur les fonctions de valeur et nous avons pu avoir la politique optimale. Nous avons eu des systèmes linéaires dans les équations d'optimalité pour chacun des critères. D'une part, dans le cadre du critère fini, les

politiques optimales sont donc de type markovien déterministe, mais non stationnaire c'est-à-dire le choix de la meilleure décision à prendre dépend de l'instant t. D'autre part, pour le critère α-pondéré, les politiques optimales sont de type stationnaire et déterministe. Pour le critère moyen, les politiques optimales sont aussi de type stationnaire et déterministe, en outre l'étude est un peu plus complexe que celle des autres, si on veut avoir une vision plus explicite, il faudra classifier les MDP. Cependant, dans la prochaine partie, nous allons nous intéresser uniquement au critère α- pondéré.

Chapitre 3

Formulation en programmation linéaire

Dans cette partie, nous allons formuler en programmation linéaire le MDP. La programmation linéaire peut être formulée en 2 formes : le programme linéaire primal et le programme linéaire dual. Afin d'aboutir à ces deux formulations, nous allons partir de la notion de dualité afin de comprendre la relation entre ces deux formulations.

Dans un problème de programmation linéaire, nous souhaitons optimiser : maximiser ou minimiser une fonction linéaire, sous contraintes formées par un ensemble d'inégalités ou d'égalités linéaires.

Nous souhaitons trouver un vecteur $\vec{x}$ avec n éléments qui maximise ou minimise la fonction objectif ou fonction économique :

$$\sum_{i=1}^{n} c_i x_i$$

sous m contraintes :

$$A\vec{x} \geq \vec{b} \text{ et pour tout } i, x_i \geq 0$$

avec : $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix}$ et $\vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_m \end{bmatrix}$ le vecteur second membre et $\vec{c} = \begin{bmatrix} c_1 \\ c_2 \\ . \\ . \\ c_n \end{bmatrix}$ le vecteur de coûts.

La fonction objectif peut s'écrire : $z = \vec{c}.\vec{x}$ (produit scalaire)
A est la matrice de contrainte dont les éléments sont formés par $(a_{ij})_{(i,j)\in\mathbb{N}\times\mathbb{N}}$, A est de taille $m \times n$.

Rappelons d'abord quelques définitions utiles pour la programmation linéaire.

Définition 3.1. *Solution [11]*

Nous appellons "solution", toutes valeurs spécifiques des variables décisionnelles $(x_1, x_2, ..., x_n)$.

Définition 3.2. *Solution réalisable [11]*

Une solution est appelée une solution réalisable si elle satisfait simultanément toutes les contraintes du problème.

Définition 3.3. *Solution extrême [11]*

Une solution extrême est une solution qui ne peut pas s'écrire comme une combinaison linéaire de 2 autres solutions.

Définition 3.4. *Solution extrême réalisable [11]*

Une solution extrême est réalisable si elle satisfait simultanément toutes les contraintes du problème.

Définition 3.5. *Solution optimale [11]*

Une solution optimale est une solution réalisable ayant la valeur de la fonction objectif la plus favorable.

Définition 3.6. *Base [11]*

Nous appellons "base" : toutes sous matrices carrés régulières extraites de A. Le complément de B dans A est le hors-base associé.

Définition 3.7. *Solution de base [11]*

Soit $A = [B, N]$, avec B la matrice de base et N la matrice hors base.
Soit $\vec{x} = [x_B, x_N]$, avec x_B les variables de base et x_N les variables hors base .
Nous appellons "Solution de base" associé à la matrice de base B la solution particulière de l'équation

$$Bx_B + Nx_N = b$$

en prenant $x_N = 0$ par conséquent $x_B = B^{-1}b$. Cette solution est donc :

$$\vec{x} = \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix}$$

Une solution de base est dite dégénérée si x_B a des composantes nulles.

3.1 Caractérisation des bases réalisables et des solutions des bases réalisables optimales

Soit $z(x) = \sum_{i=1}^{n} c_i x_i = \vec{c}.\vec{x}$, la fonction objectif. Soit $\vec{c} = [c_B, c_N]$, ainsi $z(x) = c_B.x_B + c_N.x_N$.

Théorème 3.8. *[4] - [11]*

Soit le vecteur $\Pi = (\Pi_1, \Pi_2, ..., \Pi_m)$, (m : nombre de lignes) tel que :

$$\Pi = c_B B^{-1}$$

Par définition le vecteur Π est appelé le vecteur des multiplicateurs de simplexe. Une condition nécessaire et suffisante pour que la matrice B soit une base réalisable optimale est que :

$$\bar{c}_N = c_N - \Pi N \geq 0$$

(c'est-à-dire toutes les composantes sont tous positives)

Démonstration. **Condition suffisante :**
Soient $A = [B, N]$, $x = [x_B, x_N]$. Nous avons
$Bx_B + Nx_N = b \Rightarrow Bx_B = b - Nx_N \Rightarrow x_B = B^{-1}b - B^{-1}Nx_N$

Nous avons $c = [c_B, c_N]$

$$\begin{aligned} z(x) &= \vec{c}.\vec{x} \\ &= c_B x_b + c_N x_N \\ &= c_B(B^{-1}b - B^{-1}Nx_N) + c_N x_N \\ &= c_B B^{-1}b - c_B B^{-1}Nx_N + c_N x_N \\ &= c_B B^{-1}b + (c_N - c_B B^{-1}N)x_N \\ &= c_B B^{-1}b + \bar{c}_N x_N \end{aligned}$$

or $\bar{c}_N x_N \geq 0$. Par conséquent $z(x) \geq c_B B^{-1}b = z_B$, cette valeur est atteinte en une solution réalisable :

$\vec{x} = [B^{-1}b, 0]$, et $z^*(x) = c_B B^{-1}b$.

D'autre part la valeur z_B de z est atteinte par la solution de base réalisable qui est $\vec{x^0} = [B^{-1}b, 0]$.

Condition nécessaire : Supposons que $\bar{c}_N < 0$.

Il existe une variable hors-base d'indice s telle que $\bar{c}_s < 0$. Si tel est le cas, nous allons mettre en évidence une solution de base réalisable dont le coût est inférieur à z_B.

Pour montrer qu'on n'a pas une base optimale, on va construire $\hat{x} = x^0 + \theta\vec{e}_s$.

Comme $\bar{c}_N$, il existe s tel que $\bar{c}_s < 0$ et z tel que $z(\hat{x}) < z_B = c_B B^{-1}b$.

On part de $x^0 = [B^{-1}b, 0]$.

$$\vec{e}_s = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ 1 \\ 0 \\ . \\ 0 \end{bmatrix}$$

Soit s l'indice ci-dessus. $\hat{x} = x^0 + \theta\vec{e}_s$. θ choisi convenablement.

Nous avons $\hat{x} = [B^{-1}b, \theta\vec{e}_s]$.

Or $z(\hat{x}) < z_B = c_B B^{-1}b$.

Choisissons θ de telle manière que $\hat{x}$ soit toujours une solution de base.

$A\hat{x} = b$.

$\Rightarrow Bx_B + N\theta\vec{e}_s = b$.

$\Rightarrow Bx_B = b - \theta N\vec{e}_s$.

$\Rightarrow x_B = B^{-1}b - \theta B^{-1}N\vec{e}_s$.

Posons $\bar{b} = B^{-1}b$ et $\bar{A}_s = B^{-1}A_s$.

Alors $x_B = \bar{b} - \theta\bar{A}_s$.

On suppose que $b > 0$ (hypothèse de non-dégénérescence).

On peut donc choisir θ assez petit de telle manière que x_B reste toujours ≥ 0 avec θ nombre positif.

Soit θ^* cette valeur convenable de θ, alors $\hat{x} = x^0 + \theta^*\vec{e}_s$.

Par conséquent :

$$\begin{aligned} z(\hat{x} &= z(x^0) + \bar{c}_N\theta^*\vec{e}_s \\ &= z(x^0) + \theta^*\bar{c}_s \end{aligned}$$

or $\theta^* \geq 0$ et $\bar{c}_s < 0$.

D'où $z(\hat{x}) < z(x^0)$ □

Définition 3.9. *Vecteur coûts réduits [11]*

Le vecteur $\bar{c}_N$ défini dans le théorème précédent s'appelle Vecteur coûts réduits.

3.2 Notion de Dualité en programmation linéaire

Rappelons que le problème est de trouver $\vec{x}$ avec n éléments qui minimise la fonction objectif :

$$\sum_{i=1}^{n} c_i x_i$$

sous m contraintes :

$$A\vec{x} \geq \vec{b} \text{ et pour tout } i, x_i \geq 0$$

Ce problème peut être écrit formellement comme un programme linéaire de la structure suivante appelée **programme linéaire primal** :

minimiser $\vec{c}.\vec{x}$

sous contrainte $A\vec{x} \geq \vec{b}$ et $\vec{x} \geq \vec{0}$.

A tout programme linéaire primal on peut associer un programme linéaire dit **dual** de façon qu'il existe des relations très fortes entre les solutions (variables et objectives) de l'un et de l'autre.
Donc chaque problème de minimisation peut facilement être transformé à un problème dual de maximisation de la forme :
maximiser $\vec{b}.\vec{y}$
sous contrainte ${}^{T}A\vec{y} \leq \vec{c}$ et $\vec{y} \geq \vec{0}$

3.2.1 Propriétés de la dualité

Propriété 1 [6]

Pour obtenir un problème dual à partir d'un problème primal, il suffit d'échanger le type d'optimisation (min $\Leftrightarrow$ max) et d'associer à chaque contrainte d'un problème une contrainte de l'autre et réciproquement de la façon suivante :

PRIMAL	**DUAL**
n variables, m contraintes	m variables, n contraintes
$\min \vec{c}.\vec{x}$	$\max \vec{y}.\vec{b}$
$A\vec{x} \geq \vec{b}$	$\vec{y} \geq 0$
$\vec{x} \geq 0$	$\vec{y}A \leq \vec{c}$

Propriété 2 [6]

La dualité est involutive, c'est-à-dire que le dual du dual est le primal.

Démonstration. La preuve est donnée par le tableau suivant :

PRIMAL	DUAL	forme équivalente	DUAL du DUAL	forme équivalente
$\min \vec{c}.\vec{x}$	$\max \vec{y}.\vec{b}$	$-\min -\vec{y}.\vec{b}$	$-\max -\vec{c}.\vec{z}$	$\min \vec{c}.\vec{z}$
$A\vec{x} \geq \vec{b}$	$\vec{y}.A \leq \vec{c}$	$-\vec{y}.A \geq -\vec{c}$	$-A\vec{z} \geq -\vec{b}$	$A\vec{z} \geq \vec{b}$
$\vec{x} \geq 0$	$\vec{y} \geq 0$	$\vec{y} \geq 0$	$\vec{z} \geq 0$	$\vec{z} \geq 0$

La dernière étape rend bien la forme primale. □

Remarque 3.10. *Les mots primal et dual ne sont que des étiquettes, chaque problème peut-être considéré comme un primal ou un dual mais l'usage courant est*

de considérer comme primal le problème de minimisation et dual le problème de maximisation.

3.2.2 Théorème fondamental de la dualité

Ce théorème précise les relations entre solutions du primal et du dual, il est précédé de deux lemmes :

Lemme 3.11. *[2]*

Si $\vec{x^*}$ est une solution réalisable du primal et $\vec{y^*}$ une solution réalisable du dual alors $\vec{c}.\vec{x^*} \geq \vec{y^*}.\vec{b}$.

Démonstration. Nous avons

Primal	$\min \vec{c}.\vec{x}$	Dual	$\max \vec{y}.\vec{b}$
	$A.\vec{x} \geq \vec{b}$		$\vec{y} \geq \vec{0}$
	$\vec{x} \geq \vec{0}$		$\vec{y}.A \leq \vec{c}$

Dans le primal, nous avons $A\vec{x^*} \geq \vec{b} \Rightarrow \vec{y^*}A\vec{x^*} \geq \vec{y^*}.\vec{b}$ vu que $\vec{y^*} \geq \vec{0}$
Dans le dual nous avons :$\vec{y^*}A \leq \vec{c} \Rightarrow \vec{y^*}A\vec{x^*} \Rightarrow \vec{c}.\vec{x^*}$ vu que $\vec{x^*} \geq \vec{0}$
D'où $\vec{y^*}\vec{b} \leq \vec{y^*}A\vec{x^*} \leq \vec{c}.\vec{x^*}$. □

Lemme 3.12. *[2]*

Si $\vec{x^*}$ est une solution réalisable du primal et $\vec{y^*}$ une solution réalisable du dual et si $\vec{c}.\vec{x^*} = \vec{y^*}.\vec{b}$.
Alors $\vec{x^*}$ et $\vec{y^*}$ sont des solutions optimales de leurs problèmes respectifs.

Démonstration. Si $\vec{x^*}$ n'était pas optimale, il existerait une solution $\vec{x^{**}}$ telle que $\vec{c}.\vec{x^{**}} < \vec{c}.\vec{x^*} = \vec{y^*}.\vec{b}$ ce qui est contraire au lemme3.11. De même si $\vec{y^*}$ n'était pas optimale, il existerait une solution $\vec{y^{**}}$ telle que $\vec{y^{**}}.\vec{b} > \vec{y^*}.\vec{b} = \vec{c}.\vec{x^*}$ contrairement au lemme3.11. □

Théorème 3.13. *Théorème fondamental [2]*

1. *Si le primal possède une solution optimale $\vec{x^*}$, alors le dual possède aussi une solution optimale $\vec{y^*}$ et $\vec{c}.\vec{x^*} = \vec{y^*}.\vec{b}$.*
2. *Si le dual possède une solution optimale $\vec{y^*}$, alors le primal possède aussi une solution optimale $\vec{x^*}$ et $\vec{c}.\vec{x^*} = \vec{y^*}.\vec{b}$.*

3. *Si la solution optimale du primal est non bornée (infinie) alors le dual n'a pas de solution réalisable.*
4. *Si la solution optimale du dual est non bornée (infinie) alors le primal n'a pas de solution réalisable.*
5. *Si le primal n'admet pas de solution réalisable alors le dual n'admet pas de solution réalisable ou admet une solution non bornée.*
6. *Si le dual n'admet pas de solution réalisable alors le primal n'admet pas de solution réalisable ou admet une solution non bornée.*

Démonstration. Etant donné l'involutivité de la dualité, on peut se contenter de démontrer les propositions impaires.

Soit $(x_{j_1}, x_{j_2}, ..., x_{j_m})$ les variables de base correspondantes.

Dénotons $\vec{c_B} =^T [c_{j_1}, c_{j_2}, ..., c_{j_m}]$, et Π le vecteur des multiplicateurs associés à la base optimale $\Pi = (\Pi_1, \Pi_2, ..., \Pi_m)$.

Rappelons que les coûts relatifs des variables sont définis comme suit :

$$\forall j \in \{1, ..., n\}, \bar{c}_j = c_j - \Pi A_j$$

où A_j dénote la j^{ème} colonne de la matrice A.

1. Si $\vec{x^*}$ est optimale du primal, nous avons

$$\forall j \in \{1, ..., n\}, \bar{c}_j = c_j - \Pi A_j \geq 0$$

nous obtenons

$$\Pi A_j \leq c_j, \forall j \in \{1, ..., n\}$$

soit matriciellement

$$\Pi A \leq \vec{c}$$

ou encore

$$^T A \Pi \leq \vec{c}$$

ce qui donne

$$\Pi \in \{\vec{y} :^T A\vec{y} \leq \vec{c}\}$$

On en conclut que Π vérifie les contraintes du dual (il n'y a pas de contrainte de positivité sur les variables duales).

Nous avons

$$\Pi = c_B B^{-1}$$

La fonction objectif du primal vaut :

$$z = c_B x_B = c_B \bar{b} = c_B B^{-1} \vec{b} = \Pi.\vec{b}.$$

La fonction objectif du dual pour $\vec{y} = \Pi$ vaut évidemment aussi $\Pi.\vec{b}$.
En vertu du lemme3.12, Π est la solution optimale du dual et le théorème est démontré.
3. Si le primal n'admet pas de solution bornée (inférieurement puisque le primal est un problème de minimum), $\forall \vec{K}$, il existe $\vec{x}$ telle que $\vec{c}.\vec{x} < \vec{K}$.
Si le dual admettait une solution $\vec{y^*}$, nous aurions (lemme3.11) $\forall \vec{K} : \vec{y^*}.\vec{b} \leq \vec{c}.\vec{x} < \vec{K}$ ce qui est absurde lorsque $\vec{K}$ tend vers $-\vec{\infty}$.
5.Ne nécessite pas de démonstration vu que 2) exclut la possibilité d'une solution finie. □

3.3 Formulation d'un MDP en programmation linéaire

3.3.1 Programme linéaire primal

Rappelons que la traduction générale du problème $x = \max\{a_1, ..., a_n\}$ à un programme linéaire est comme suit :
minimiser : x
sous contrainte $\forall i$, $x \geq a_i$
L'équation d'optimalité du problème de l'horizon infini pour le critère α-pondéré est :

$$V(s) = \max_{a \in A} \{r(s,a) + \alpha \sum_{s' \in S} p(s'|s,a) V(s')\}$$

avec $0 < \alpha < 1$.
En utilisant la traduction ci-dessus nous obtiendrions le programme linéaire suivant : déterminer $V(s)$ minimisant :

$$\sum_{s \in S} \beta(s) V(s)$$

sous-contrainte

$$\forall s \in S, \forall a \in A_s : \; V(s) \geq r(s,a) + \alpha \sum_{s' \in S} p(s'|s,a) V(s') \tag{3.1}$$

ou sous forme matricielle

$$(I - \alpha P_\pi)\vec{V} \geq \vec{r}_\pi$$

Voici donc le programme linéaire primal :
minimiser $\vec{\beta}\vec{V}$
sous-contrainte $(I - \alpha P_\pi)\vec{V} \geq \vec{r}_\pi$.

Avec $\beta(s)$ la pondération d'intérêt de l'état s (state relevance weight) telle que

- $\forall s \in S$, $\beta(s > 0)$
- $\sum_{s \in S} \beta(s) = 1$

$\beta(s)$ est la distribution de la probabilité de l'état initial de l'agent dans le MDP : $\beta(s) = Prob(s = s_0)$ (la probabilité pour que l'état s soit l'état initial).

Il est immédiat de vérifier que si $V \in W$ minimise la fonction $\sum_{s \in S} V(s)$ sous la contrainte $V \geq LV$, alors $V = V_\alpha^*$. En effet, nous avons montré au cours de la preuve du théorème 2.18 que $V \geq LV$ impliquait $V \geq V_\alpha^*$ et donc que $\beta V \geq \beta V_\alpha^*$, d'où $\sum_{s \in S} \beta(s)V(s) \geq \sum_{s \in S} \beta(s)V_\alpha^*(s)$.

3.3.2 Programme linéaire dual

Rappelons si nous avons comme programme linéaire primal :
minimiser $\vec{c}.\vec{x}$
sous contrainte :$A\vec{x} \geq \vec{b}$ et $\vec{x} \geq \vec{0}$

Le problème dual de maximisation associé est de la forme :
maximiser $\vec{b}.\vec{y}$
sous contrainte :${}^T A\vec{y} \leq \vec{c}$ et $\vec{y} \geq \vec{0}$ (avec ${}^T A$ est la transposée de la matrice A)

Or nous avons comme programme linéaire primal :
minimiser $\vec{\beta}.\vec{V}$
sous contrainte :$(I - \alpha P_\pi)\vec{V} \geq \vec{r}_\pi$ et $\vec{V} \geq \vec{0}$

Le programme linéaire dual associé est donc :
maximiser $\vec{r}_\pi.\vec{y}$
sous contrainte :${}^T(I - \alpha P_\pi)\vec{y} \leq \vec{\beta}$ et $\vec{y} \geq \vec{0}$

Plus explicitement, déterminer $y(s,a)$ qui maximise

$$\sum_{s\in S} r(s,a)y(s,a)$$

sous contrainte :

$$y(s',a) - \alpha \sum_{s\in S} p(s'|s,a)y(s,a) \leq \beta(s')$$

et $y(s,a) \geq 0$ pour tout $s' \in S$ et $a \in A$, telle que $y(s,a)$ est la probabilité d'être dans l'état s et d'effectuer l'action a, tout en tenant compte du facteur d'actualisation α.

Finalement, nous cherchons $y(s,a)$ qui maximise

$$\sum_{a\in A}\sum_{s\in S} r(s,a)y(s,a)$$

sous-contrainte

$$\forall s' \in S,\ \sum_{a\in A} y(s',a) - \sum_{a\in A}\sum_{s\in S} p(s'|s,a)y(s,a) \leq \beta(s') \tag{3.2}$$

et

$$y(s,a) \geq 0 \text{ pour tout } s \in S \text{ et } a \in A$$

Cette formulation est clairement un programme linéaire qui peut être résolu par la méthode de simplexe. Une fois $y(s,a)$ est obtenue, une politique optimale peut ensuite être calculée.
$\pi(a,s) = Prob[action = a|état = s]$ est facilement trouvé par :

$$\pi(a,s) = \frac{y(s,a)}{\sum_{a\in A} y(s,a)}$$

En effet,
soit $y(s,a) = Prob(action = a \cap état = s)$ et $q(s) = Prob(état = s)$
Nous avons :

$$P((A \cap B) = P(A/B)P(B)$$

D'où

$$Prob(action = a \cap état = s) = Prob(action = a/état = s)Prob(état = s)$$

ou

$$y(s,a) = \pi(a,s)q(s)$$

Or $y(s,a) = Prob(action = a \cap état = s)$ et

$$\begin{aligned}\sum_{a\in A} y(s,a) &= \sum_{a\in A} Prob(action = a \cap état = s) \\ &= Prob(état = s) = q(s)\end{aligned}$$

et nous avons

$$\begin{aligned}y(s,a) &= \pi(a,s)q(s) \\ &= \pi(a,s)\sum_{a\in A} y(s,a)\end{aligned}$$

D'où

$$\pi(a,s) = \frac{y(s,a)}{\sum_{a\in A} y(s,a)}$$

3.4 La méthode du simplexe

La méthode du simplexe est une méthode algébrique pour trouver la solution optimale d'un problème de programmation linéaire. Ainsi nous pouvons trouver la politique optimale avec la méthode du simplexe.

Rappelons que la fonction objectif est de la forme $z = \vec{c}.\vec{x}$

avec $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix}$ et $\vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ b_m \end{bmatrix}$ le vecteur second membre et $\vec{c} = \begin{bmatrix} c_1 \\ c_2 \\ . \\ . \\ c_n \end{bmatrix}$ le vecteur de coûts.

Et A la matrice de contrainte de taille $m \times n$ dont les éléments sont formés par $(a_{ij})_{ij}$.

Prenons la structure suivante :

$$\text{maximiser} z = \vec{c}.\vec{x}$$

sous contrainte

$$A\vec{x} = \vec{b}$$

et $\vec{x} \geq 0$ (c'est-à-dire tous les éléments sont tous positifs)

3.4.1 Principes de la méthode du simplexe

- **Principe 1** [4]

  On calcule une suite de sommets du polyèdre des contraintes (solutions de base admissibles), chacune d'elles étant meilleure (pour la fonction économique) que la précédente.

- **Principe 2** [4]

  (a) S'il existe exactement une solution optimale, alors c'est une solution extrême réalisable.

  (b) S'il existe plusieurs solutions optimales, alors au moins deux d'entre elles sont des solutions extrêmes réalisables adjacentes.

- **Principe 3** [4]

  Il existe un nombre fini de solution extrêmes réalisables.

- **Principe 4** [4]

  Si une solution extrême réalisable est meilleure que toutes les autres solutions extrêmes réalisables qui sont adjacentes, alors elle est meilleure que toute autre solution extrême réalisable.

3.4.2 Algorithme du simplexe

Algorithme 1. *[11]*

Nous supposons que nous disposons au départ d'une base réalisable.

1. *Soit* B^0 *une base réalisable*

   k=0

2. $k = k + 1$

3. *A l'itération k, soit B la base courante*

   Soit $x = x_B, x_N$, *la solution de base correspondante*

   Calculer $\bar{b} = B^{-1}b$ *(les valeurs des variables de base)*

   $\Pi = c_B B^{-1}$ *(Multiplicateurs du simplexe)*

   $\bar{c}_N = c_N - \Pi N$ *(Les couts réduits)*

4. *Si* $\bar{c}_N \geq 0$ *Stop, l'optimum est atteint*

   Sinon il existe s tel que alors $\bar{c}_s < 0$ *alors*

Soit A_s la colonne s de A

Calculer $\bar{A}_s = B^{-1}A_s$

Si $\bar{a}_{is} \leq 0$, $i = 1, ..., m$ Stop optimum $= -\infty$

Sinon calculer

$$\hat{x}_s = \frac{\bar{b}_r}{\bar{a}_{rs}} = \min_{i/\bar{a}_{is}>0} \frac{\bar{b}_i}{\bar{a}_{is}}$$

5. *Soit x_t la variable correspondant à la $r^{ième}$ ligne de la base c'est-à-dire tq*

$$B^{-1}A_t = \vec{e}_r = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ 1 \\ . \\ 0 \\ 0 \end{bmatrix} \text{ la } r^{ième} \text{ ligne.}$$

Alors

-la variable s rentre en base ($\hat{x}_s > 0$)

-la variable t sort de la base ($x_t = 0$)

La nouvelle solution courante $\hat{x}$ correspond à la nouvelle base réalisable

$$\hat{B} = B + A_s - A_t$$

Calculer $\hat{B}^{-1}$ et retourner en (2).

Conclusion

La formulation linéaire d'un MDP nous permet de calculer une solution optimale. Cependant, il y a de la complexité car la résolution est coûteuse en temps de calcul et en mémoire. En effet, dans le programme linéaire de l'équation 3.2, le nombre exponentiellement croissant d'états possibles dans le problème se retrouve à la fois dans le nombre de variables à déterminer, dans le nombre de termes de la somme à maximiser, le nombre de contraintes à satisfaire et le nombre de termes dans le produit de chaque contrainte. De plus les fonctions de transition P_π et de récompense r_π doivent être connues.

Chapitre 4

Les Algorithmes "Policy-Iteration" et "Value Iteration"

Lorsque les fonctions de transition et de récompense sont connues, la programmation dynamique est une approche alternative à la programmation linéaire pour calculer la fonction de valeur optimale d'un MDP. Nous allons voir les deux algorithmes du programmation dynamique les plus connus et les plus utilisés pour résoudre le MDP : "Value Iteration", décrit dans [3] et "Policy Iteration" introduit trois ans plus tard par Howard en 1960. Rappelons que nous allons nous intéresser au critère α-pondéré pour toute la suite.

La programmation dynamique désigne un ensemble d'algorithmes permettant de calculer les politiques optimales d'un MDP fini. Ces algorithmes reposent sur les hypothèses suivantes :
1-la fonction de transition est connue ;
2-la fonction de récompense est connue.
Les algorithmes de programmation dynamique permettent donc de trouver l'ensemble des solutions d'un MDP uniquement si celui-ci est parfaitement connu. Généralement, deux étapes composent un algorithme de programmation dynamique : l'évaluation d'une politique et l'amélioration d'une politique.

4.1 Algorithme d'itération sur les valeurs ou "Value Iteration"

L'algorithme d'itération sur les valeurs ou "Value Iteration" (voir [15]), est issu de la programmation dynamique, est l'un des algorithmes standards des Processus Décisionnels Markoviens.

4.1.1 Principe de l'algorithme de "Value Iteration"

Cet algorithme consiste à calculer itérativement la valeur de chaque état. Nous avons vu que la valeur d'un état est le gain obtenu par l'exécution d'une action auquel nous ajoutons les valeurs des différents états qu'il est possible d'atteindre en exécutant cette même action tout en prenant en compte le facteur d'actualisation α permettant d'introduire un compromis entre action à court terme et action à long terme.

Cet algorithme se base sur la résolution directe de l'équation d'optimalité de Bellman , en utilisant pour cela une méthode itérative de type point fixe, d'où son nom anglais de "Value Iteration" [15].

Rappelons que

$$\forall V \in W, \forall s \in S, \quad LV(s) = \max_{a \in A}\{r(s,a) + \alpha \sum_{s' \in S} p(s'|s,a)V(s')\}$$

ou en notation vectorielle

$$\forall V \in W, \quad LV = \max_{\pi \in D}\{r_\pi + \alpha P_\pi V\}$$

Le Théorème2.15 : (Equation de Bellman-optimalité des fonctions de valeur pour le critère -pondéré), nous a montré que la solution de l'équation de Bellman est obtenue comme limite de la suite $V_{n+1} = LV_n$, quelle que soit l'initialisation de V_0.

4.1.2 Algorithme de "Value Iteration"

Algorithme 2. *[20]*

Entrée : MDP et paramètres ϵ et α

1. *Choisissez une fonction de valeur initiale V_0 (en choisissant un nombre pour tout $s \in s$)*

   $n \leftarrow 0$

2. *Assignez la prochaine fonction de valeur*

$$V_{n+1}(s) = \max_{a \in A}\{r(s,a) + \alpha \sum_{s' \in S} p(s'|s,a)V_n(s')\}, \quad \forall s \in S,$$

3. *Si $\|V_{n+1} - V_n\| \leq \epsilon\frac{1-\alpha}{2\alpha}$ STOP l'optimum est atteint.*

   Choisissez la politique sortante telle que :

$$\pi(s) = arg\max_{a \in A}\{r(s,a) + \alpha \sum_{s' \in S} p(s'|s,a)V_n(s')\}, \quad \forall s \in S$$

   Sinon retourner en 2 avec $n = n + 1$.

4.1.3 Convergence de l'algorithme de "Value Iteration"

Dans le théorème suivant, nous prouvons que l'algorithme trouve la politique ϵ- optimale dans un nombre finie d'étape. Notons que la politique choisie pourrait en fait être la politique optimale, mais nous n'avons aucune manière de savoir cela à l'avance.

Théorème 4.1. *[20]*

Pour la suite $(\vec{V_n})_n$ et la politique π_ϵ calculé par "Value Iteration", nous avons :

1. $\lim_{n\to\infty} V_n = V_\alpha^*$

2. *Il existe N , tel que pour tout $n > N$ $\|V_{n+1} - V_n\| < \epsilon\frac{1-\alpha}{2\alpha}$*

3. *La politique π_ϵ est optimale*

4. *Si $\|V_{n+1} - V_n\| < \epsilon\frac{1-\alpha}{2\alpha}$ alors $\|V_{n+1} - V_\alpha^*\| < \frac{\epsilon}{2}$*

Démonstration. La partie (1) et (2) provient directement de la propriété de la suite $V_{n+1} = LV_n$.

Partie (3) et (4) :

Supposons que $\|V_{n+1} - V_n\| < \epsilon\frac{1-\alpha}{2\alpha}$ comme c'est le cas quand l'algorithme s'arrête, et montrons que

$\|V_\alpha^{\pi_\epsilon} - V_\alpha^*\| < \epsilon$, ce qui fait la politique π_ϵ est ϵ-optimal.

Soit

$$\|V_\alpha^{\pi_\epsilon} - V_\alpha^*\| < \|V_\alpha^{\pi_\epsilon} - V_{n+1}\| + \|V_{n+1} - V_\alpha^*\| \tag{4.1}$$

Nous allons majorer maintenant chaque partie de la somme individuellement :

$$\begin{aligned}\|V_\alpha^{\pi_\epsilon} - V_{n+1}\| &= \|L_{\pi_\epsilon}V_\alpha^{\pi_\epsilon} - V_{n+1}\|(\text{car}V_\alpha^{\pi_\epsilon}\text{est un point fixe de}L_{\pi_\epsilon}) \\ &\leq \|L_{\pi_\epsilon}V_\alpha^{\pi_\epsilon} - LV_{n+1}\| + \|LV_{n+1} - V_{n+1}\|\end{aligned}$$

Quand π_ϵ est maximum sur les actions utilisant V_{n+1}, celà implique que $L_{\pi_\epsilon}V_{n+1} = LV_{n+1}$ et nous concluons que :

$$\begin{aligned}\|V_\alpha^{\pi_\epsilon} - V_{n+1}\| &\leq \|L_{\pi_\epsilon}V_\alpha^{\pi_\epsilon} - L_{\pi_\epsilon}V_{n+1}\| + \|LV_{n+1} - LV_n\| \\ &\leq \alpha|V_\alpha^{\pi_\epsilon} - V_{n+1}\| + \alpha\|V_{n+1} - V_n\|\end{aligned}$$

De cette inégalité, il suit que :

$$\|V_\alpha^{\pi_\epsilon} - V_{n+1}\| \leq \frac{\alpha}{1-\alpha}\|V_{n+1} - V_n\|$$

Pour la deuxième partie du somme, nous opérons de la même façon :

$$\|V_{n+1} - V_\alpha^* \leq \|\frac{\alpha}{1-\alpha}\|V_{n+1} - V_n\|$$

D'où la partie (4) du théorème.

$$\frac{\alpha}{1-\alpha}\|V_{n+1} - V_n\| + \frac{\alpha}{1-\alpha}\|V_{n+1} - V_n\|$$

En retournant vers l'inégalité (4.1), il suit que :

$$\|V_\alpha^{\pi_\epsilon} - V_\alpha^*\| \leq \frac{2\alpha}{1-\alpha}\|V_{n+1} - V_n\| < \epsilon$$

Par conséquent la politique sélectionnée π_ϵ est $\epsilon-$optimale. □

Dans la suite, nous allons montrer que la convergence à la politique optimale d'algorithme "Value iteration" est monotone, et que la convergence est exponentiellement rapide dans le paramètre α.

Lemme 4.2. *Monotonie de la convergence [20]*

1. *Si* $V \geq U$ *alors* $LV \geq LU$
2. *Si* $LV_n \geq LU_n$ *alors* $\forall m \geq 0$, $V_{n+m+1} \geq V_{n+m}$

Démonstration. **Partie 1** Soit

$$\delta \in arg \max_{\pi \in \Pi^{MD}} \{r_\pi + \alpha P_\delta U\}$$

Comme $P_\delta \geq 0$, il suit que $P_\delta V \geq P_\delta U$. Par conséquent

$$LU = r_\delta + \alpha P_\delta U \leq r_\delta + \alpha P_\delta V \leq \max_{\pi \in \Pi^{MD}} \{r_\pi + \alpha P_\pi V\} = LV$$

Cela prouve la partie (1).

Partie 2 De la partie (1) il suit que si $V \geq U$, alors $\forall m \geq 1, L^m V \geq L^m U$.

$$V_{n+m+1} = L^m L V_n \geq L^m V_n = V_{n+m}$$

□

Théorème 4.3. *[20]*

Soit V_n *la suite de valeurs calculée par l'algorithme "Value Iteration"*

1. $\forall n, \|V_n - V_\alpha^*\| \leq \frac{\alpha^n}{1-\alpha}\|V_1 - V_0\|$
2. $\forall n, \|V_\alpha^{\pi_n} - V_\alpha^*\| \leq \frac{2\alpha^n}{1-\alpha}\|V_1 - V_0\|$ *où* π_n *est la politique définie par* V_n

Démonstration. **Partie 1** Nous allons utiliser le résultat de la partie (4) du théorème 4.1 :

Si $\|V_{n+1} - V_n\| < \epsilon\frac{1-\alpha}{\alpha}$ alors $\|V_{n+1} - V_\alpha^*\| < \epsilon$.

Pour l'utiliser, nous allons majorer :
$\|V_n - V_{n-1}\| = \|L^{n-1}V_1 - L^{n-1}V_0\| \leq \alpha^{n-1}\|V_1 - V_0\|$ car (L contractante).

Soit $\epsilon = \frac{\alpha^n}{1-\alpha}\|V_1 - V_0\|$, de sorte que nous puissions employer le théorème4.1 pour conclure que :

$$|V_n - V_\alpha^*\| \leq \frac{\alpha^n}{1-\alpha}\|V_1 - V_0\|$$

Ceci prouve la partie 1.

Partie 2 Pour prouver la partie 2, nous allons majorer

$$\|V_\alpha^{\pi_n} - V_\alpha^*\| \leq \|V_\alpha^{\pi_n} - V_n\| + \|V_n - V_\alpha^*\|$$

La majoration suivante dérive (a) de la même résultat que du théorème 4.1 et (b) du partie (1) résultat de ce théorème, respectivement :

$$\begin{aligned}\|V_\alpha^{\pi_n} - V_\alpha^*\| &\leq \underbrace{\frac{\alpha}{1-\alpha}\|V_n - V_{n-1}\|}_{(a)} + \underbrace{\frac{\alpha^n}{1-\alpha}\|V_1 - V_0\|}_{(b)} \\ &\leq \frac{2\alpha^n}{1-\alpha}\|V_1 - V_0\|\end{aligned}$$

La dernière inégalité est obtenu car L est un opérateur contractante. □

De ce théorème, il découle que chaque itération de l'algorithme "Value itération" est plus près de V_α^* par un facteur de α. Quand α approche 1, le taux de convergence diminue. Les majorations que nous avons montrées peuvent être employées pour déterminer le nombre d'itérations requises pour un problème spécifique.

4.1.4 Complexité de l'algorithme

L'efficacité de l'algorithme dépend de deux facteurs : la complexité d'une itération, et le nombre d'itérations nécessaire pour converger. Chaque itération consiste à calculer la valeur de transition entre les états, pour chaque action, d'après [19] : cela nécessite $|A||S|^2$ opérations. L'algorithme de "Value Iteration" est donc de complexité polynomial. Le nombre d'itérations nécessaire est plus difficile à déterminer. [13] montre que l'algorithme est polynomial selon $|S|$, $|A|$, α et B, où B est le nombre de bits nécessaires à la représentation des données du problème.

4.2 Algorithme d'itération sur les politique ou "Policy iteration"

Cet algorithme de résolution est constitué des méthodes itérant sur les politiques elles-mêmes.

4.2.1 Principe de l'algorithme de "Policy iteration"

Le principe de l'algorithme d'itération sur les politiques est très simple : partant d'une politique initiale quelconque, l'algorithme cherche à maximiser la valeur de chaque état, et itère jusqu'à obtenir deux politiques successives identiques, qui sont alors optimales.
Plus formellement :
Soit la politique π_n à l'itération n.
Dans une première étape, on résout le système d'équations linéaires : $V_n = L_{\pi_n} V_n$

$$V_n(s) = \max_{a \in A} \{ r_{\pi_n}(s,a) + \alpha \sum_{s' \in S} p_{\pi_n}(s'|s,a) V_n(s') \}$$

pour tout $V \in W$ et $s \in S$.
Puis, dans le deuxième étape, nous améliorons la politique courante en posant :

$$\pi_{n+1} \in arg \max_{\delta \in D} \{ r_\delta + \alpha P_\delta V_\alpha^\pi \}$$

On stoppe l'algorithme lorsque : $\pi_{n+1} = \pi_n$.

La suite V_n, croissante et bornée par V_α^*, converge. Comme il y a un nombre fini de politiques, la suite π_n converge alors en un nombre fini d'itération. A la limite $V_n = V_\alpha^*$, et π_n est optimale.

4.2.2 Algorithme de "Policy Iteration"

Algorithme 3. *[20]*

Entrée MDP et α.

1. *Initialiser :* $\pi_0 \in \Pi^{MD}$, $n \leftarrow 0$

2. *determination des valeurs ou "value determination"*

   Trouver V_n (la fonction de valeur de π_n) en résolvant le système de /S/ équations avec /S/ inconnues valeur de $V_n(s)$.

$$V_n(s) = r_{\pi_n}(s, \pi_n(s)) + \alpha \sum_{s' \in S} p(s'|s, \pi_n(s)) V_n(s')$$

   ou en notation matricielle

$$(I - \alpha P_{\pi_n})V = r_{\pi_n}$$

3. *amélioration de la politique ou "Policy improvement"*

   Choisissez la politique suivante :

$$\pi_{n+1} \in \arg \max_{\pi \in \Pi^{MD}} \{r_\pi + \alpha P_\pi V_{\pi_n}\}$$

4. *Si $\pi_{n+1} = \pi_n$ STOP.*

   Sinon $n \leftarrow n + 1$, retourner à l'étape (2).

4.2.3 Convergence de l'algorithme de "Policy Iteration"

Nous verrons que quand le nombre d'états |S|, et le nombre des actions |A| sont finis, il n'y a pas deux itérations consécutives avec la même politique, à moins que nous ayons une politique optimale. Par conséquent π_n converge vers une politique optimale dans un nombre fini d'étapes. La clé de la convergence de π_n est la monotonie de $\{V_n\}$.

Lemme 4.4. *[20]*

Soient V_n, V_{n+1} les valeurs d'itérations consécutives de l'algorithme ci-dessus, alors

$$V_n \leq V_{n+1} \leq V_\alpha^*$$

Démonstration. Soit π_{n+1}, la politique dans l'étape "Policy improvement" , alors $r_{\pi_{n+1}} + \alpha P_{\pi_{n+1}} V_n \geq r_{\pi_n} + \alpha P_{\pi_n} V_n = V_n$ (Par définition de V_n).
Par conséquent :

$$r_{\pi_{n+1}} \geq (I - \alpha P_{\pi_{n+1}})V_n$$

En multipliant par $(I - \alpha P_{\pi_{n+1}})^{-1}$ on obtient :

$$V_{n+1} = (I - \alpha P_{\pi_{n+1}})^{-1} r_{\pi_{n+1}} \geq V_n$$

□

Théorème 4.5. *[20]*

Soient S et A finie, alors l'algorithme de "Policy Iteration" converge vers une politique optimale après tout au plus $|A|^{|S|}$ *itérations.*

Démonstration. Clairement,$|A|^{|S|} \geq |\Pi^{MD}|$. D'après le Lemme4.4, dans chaque étape $V_{n+1} \geq V_n$ sauf pour la dernière étape dans laquelle $V_{n+1} = V_n$. Par conséquent aucune politique $\pi \in \Pi^{MD}$ peut apparaître dans deux itérations différentes. Par conséquent le nombre d'itérations est $\leq |\Pi^{MD}| \leq |A|^{|S|}$. □

4.2.4 Complexité de l'algorithme

Chaque itération de l'algorithme consiste en deux opérations : la résolution du système d'équations, qui nécessite un peu moins de $|S|^3$ opérations d'après [19], et la phase d'amélioration, qui est effectuée en $|A||S|^2$ opérations d'après [19]. Donc, l'algorithme de "Policy Iteration" est aussi de complexité polynomial. Comme précédemment, le nombre d'itérations nécessaire à la convergence de l'algorithme est plus difficile à déterminer.[13] donne le même résultat que pour "Value Iteration".

4.3 Comparaison des deux algorithmes "Value Iteration" et "Policy Iteration"

Regardons maintenant le taux de la convergence des algorithmes "Value Iteration"et "Policy Iteration". Nous allons montrer que, en supposant que les deux algorithmes commencent avec la même valeur rapprochée, l'algorithme du "Policy Iteration" converge plus vite à la valeur optimale.

Théorème 4.6. *[20]*

Soient U_i *la suite crée par l'algorithme de "Value Iteration" (où* $U_{n+1} = LU_n$ *) et soit* V_i *la suite crée par l'algorithme de "Policy Iteration".*
Si $U_0 = V_0$*, alors pour tout n*

$$U_n \leq V_n \leq V_\alpha^*.$$

Démonstration. Nous utiliserons la récurrence pour prouver le théorème.
Nous assumons que $U_0 = V_0$.

V_0 est la valeur retournée du politique spécifique, et par conséquent il est clair qu'il est inférieur à la valeur optimale retournée. Par conséquent :$U_0 \leq V_0 \leq V_\alpha^*$ Donc le théorème est vrai pour $n = 0$.
Supposons maintenant qu'il est vrai pour n, et montrons le pour $n = n + 1$.

$$U_{n+1} = LU_n = L_{p_n} U_n$$

où

$$p_n \in arg \max_{\pi \in \Pi^{MD}} \{r_\pi + \alpha P_\pi U_n\}$$

Par hypothèse de récurrence $U_n \leq V_n$, et L_{p_n} comme est monotone il suit que :

$$L_{p_n} U_n \leq L_{p_n} V_n$$

Puisque L prend le maximum au-dessus de toutes les politiques :

$$L_{p_n} U_n \leq L V_n$$

Nous noterons par π_n la politique déterminée par l'algorithme de "Policy Iteration" à l'itération n et par conséquent $LV_n = L_{\pi_n} V_n$.
Par l'équation d'optimalité on obtient :

$$L_{\pi_n} V_n \leq L_{\pi_n} V_\alpha^{\pi_n}$$

Par la définition de V_{n+1}, nous avons :

$$L_{\pi_n} V_\alpha^{\pi_n} = V_{n+1}$$

Et on obtient :

$$U_{n+1} \leq V_{n+1}$$

Puisque $V_{n+1} \leq V_\alpha^*$ par conséquent

$$U_{n+1} \leq V_{n+1} \leq V_\alpha^*$$

□

Conclusion

Nous avons présenté dans cette partie les algorithmes classiques du contrôle optimal stochastique "Value Iteration" et "Policy Iteration". Ces deux algorithmes ont un fonctionnement très différent :

– d'une part "Value Iteration" procède par de petites améliorations successives de la fonction de valeur. En pratique, cet algorithme nécessite un grand nombre d'itérations pour converger, mais chaque itération est très rapide.
– d'autre part "Policy Iteration" améliore beaucoup la fonction de valeur à chaque itération. Généralement, le nombre d'itérations nécessaire à la convergence est faible, mais chaque itération est très coûteuse.

Du théorème 4.6, il découle que, assumant le même point de départ, l'algorithme de "Policy Iteration" exige moins d'étapes que l'algorithme "Value Iteration" pour converger à la politique optimale. Cependant, il devrait noter que chaque étape simple de "Policy Iteration" exige une solution d'un ensemble d'équations linéaires (l'étape d'évaluation de politique) et donc il est informatique plus cher qu'une étape simple de l'algorithme de "Value Iteration".
Soulignons le fait que tous ces algorithmes sont des formes d'itération sur les politiques qui diffèrent par la taille du pas effectue en direction de la valeur de la politique courante à chaque itération. Dans la partie suivante, nous allons appliquer ces algorithmes dans un étude de cas pour avoir une vision plus explicite du MDP et sa résolution.

Chapitre 5

Etude de cas

Les stratégies de maintenance de machines et leurs évaluations demeurent une préoccupation particulièrement forte au sein des entreprises aujourd'hui. Les enjeux économiques dépendant de la compétitivité de chacune d'entre elles sont de plus en plus étroitement liés à l'activité et à la qualité des interventions de l'entretien des machines. C'est pourquoi, nous allons alors utiliser l'approche markovienne comme une stratégie de maintenance. Nous allons voir ici explicitement et numériquement l'application d'un MDP dans le cadre de l'optimisation des revenus d'une machine. D'abord, on va modéliser la maintenance d'une machine par un MDP. Ensuite, nous allons utiliser les deux algorithmes de la programmation dynamique "Value iteration" et "Policy iteration" pour la résolution sans oublier la programmation linéaire. Dans tous les cas, nous utiliserons le langage de programmation C++ pour implémenter ces algorithmes.

5.1 Formalisation du problème

Nous souhaitons alors maximiser le revenu provenant de la production d'une machine. Or, le niveau de production dépend de l'état de la machine, et l'état de la machine dépend de son entretien. Nous considérons ainsi la maintenance, en temps discret, de la machine dont l'état de fonctionnement est modélisé par une chaîne de Markov à états en nombre fini. Une politique de maintenance consiste à décider les instants où l'on entretient la machine. On décide d'un niveau de dégradation inacceptable qui lorsqu'elle est atteinte entraine la réparation. On

suppose qu'après chaque réparation (qui est instantanée) la machine revient dans l'état de fonctionnement parfait.

5.1.1 Définition de l'ensemble d'états

Nous modélisons ainsi l'évolution de l'état de la machine par une chaîne de Markov. Chaque état représente le niveau de dégradation. Notons 0 l'état de fonctionnement parfait et S l'état d'épave inutilisable. Les états intermédiaires sont représentés par un entier compris entre 1 et S.

Notons X_t l'état de la machine. $X_t \in \{0, 1, ..., S-1\}$.

La machine peut être donc dans une des $E+1$ états :

- état 0 : signifie que la machine est neuve
- état 1 : signifie que la machine est en bonne état, dégradation de niveau 0 ;
- état 2 : signifie que la machine en mauvaise état avec un niveau de dégradation 1 ;
- ... ...
- état S-1 : signifie que la machine est dans l'état avec un niveau de dégradation d
- état S : signifie que la machine est dans l'état d'épave inutilisable.

5.1.2 Définition de l'ensemble d'actions

Face à ce problème, on a 3 décisions (ou actions) possibles :

- action 1 : entretenir la machine, qui la renvoie à l'état 0. Plus précisément, nous appelons cette politique : politique de maintenance. Après chaque réparation, la machine revient dans l'état de fonctionnement parfait. Le coût de l'entretien dépend du niveau de dégradation de la machine.
- action 2 : laisser la machine dans son état, sans politique de maintenance. C'est-à-dire, on utilise la machine sans se soucier de son état jusqu'à ce qu'elle soit devenue une épave.

– action 3 : remplacer la machine à chaque dégradation, qui la renvoie à l'état neuve.

Notons A l'ensemble des actions possibles.

$$A = \{action1, action2, action3\}$$

5.1.3 Définition des récompenses

Le rendement par période est noté $n(s)$ c'est-à-dire le nombre d'objet produit par la machine dépend de l'état de la machine. Nous avons

– à l'état état neuf $n(s) = p$, c'est le rendement maximum.

– à l'état d'épave inutilisable $n(s) = 0$, la machine ne produit plus aucun objet.

– dans les états intermédiaires, le nombre de service diminue au fur et à mesure que l'état de la dégradation augmente.

Le revenu par objet produit est $r_0(s)$.

Ainsi, le gain par période est donc $g(s) = r_0(s).n(s)$.

Par conséquent, le revenu total est le reste de des gains en enlevant le coût de l'entretien qui dépend de l'état de la machine et de la décision prise :

$$r(s, a) = g(s) - c(s, a)$$

5.2 Application numérique

Nous considèrerons ici des politiques markoviennes déterministes et stationnaires. Utilisons les données du problème suivantes :
Prenons 4 états, les états sont : 0 (neuf), 1 (bon état), 2 (mauvais état) et 3 (en panne).

$$S = \{0, 1, 2, 3\}$$

Prenons comme rendements respectifs (par période) 30, 15, 5 et 0. Les actions possibles sont : entretenir (1), ne rien faire (2), remplacer (3).

$$A = \{1, 2, 3\}$$

Supposons que le revenu par objet produit est de 100$.

Prenons comme taux d'actualisation $\alpha = 0.8$.

Les tableaux suivants fournissent pour chaque état les informations sur les revenus associés aux différentes combinaisons d'états et de décision, ainsi que les probabilités de transition.

5.2.1 état s=0

Le rendement dans l'état 0 est de 30.Donc le gain obtenu est de 3000$.

action	cout	Prob de transition	Nouvel état s'	revenu $r(s,a)$
1	500	3/4=0.75	0	2500
1	500	1/4=0.25	1	2500
1	500	0	2	2500
1	500	0	3	2500
2	0	0	0	3000
2	0	4/5=0.8	1	3000
2	0	0	2	3000
2	0	1/5=0.2	3	3000
3	3000	1	0	0
3	3000	0	1	0
3	3000	0	2	0
3	3000	0	3	0

5.2.2 état s=1

Le rendement dans l'état 1 est 15, doù le gain obtenu est de 1500$.

action	cout	Prob de transition	Nouvel état s'	revenu $r(s,a)$
1	1000	0	0	500
1	1000	4/7=0.57	1	500
1	1000	2/7=0.29	2	500
1	1000	1/7=0.14	3	500
2	0	0	0	1500
2	0	0	1	1500
2	0	4/5=0.8	2	1500
2	0	1/5=0.2	3	1500
3	3000	1	0	-1500
3	3000	0	1	-1500
3	3000	0	2	-1500
3	3000	0	3	-1500

5.2.3 état s=2

Le rendement dans l'état 2 est 5, doù le gain obtenu est de 500$.

action	cout	Prob de transition	Nouvel état s'	revenu $r(s,a)$
1	1000	0	0	-1000
1	1000	0	1	-1000
1	1000	3/4=0.75	2	-1000
1	1000	1/4=0.25	3	-1000
2	0	0	0	500
2	0	0	1	500
2	0	1/2=0.5	2	500
2	0	1/2=0.5	3	500
3	3000	1	0	-2500
3	3000	0	1	-2500
3	3000	0	2	-2500
3	3000	0	3	-2500

5.2.4 état s=3

Le rendement dans l'état 3 est 0, doù le gain obtenu est null.

action	cout	Prob de transition	Nouvel état s'	revenu $r(s,a)$
1	∞	0	0	$-\infty$
1	∞	0	1	$-\infty$
1	∞	0	2	$-\infty$
1	∞	1	3	$-\infty$
2	∞	0	0	$-\infty$
2	∞	0	1	$-\infty$
2	∞	0	2	$-\infty$
2	∞	1	3	$-\infty$
3	3000	1	0	-3000
3	3000	0	1	-3000
3	3000	0	2	-3000
3	3000	0	3	-3000

Remarque 5.1. *Nous avons des nombres ∞ puisque laisser la machine dans un état inopérable sans politique de maintenance ou le réparer quand c'est inopérable induit une valeur de coût infini.*

5.3 Résolution du problème

5.3.1 Résolution par l'Algorithme de "Policy Iteration"

La première étape de l'algorithme de "Policy Iteration" réclame choisir une politique arbitrairement. Choisissons alors la politique π_1 qui réclame de :

– entretenir une machine neuve

$$\pi_1(0) = 1$$

– ne rien faire si la machine est en bon état

$$\pi_1(1) = 2$$

– entretenir une machine en mauvaise état

$$\pi_1(2) = 1$$

– remplacer une machine en panne

$$\pi_1(3) = 3$$

La matrice de transition associée à cette politique π_1 est donnée par :

état	0	1	2	3
0	0.75	0.25	0	0
1	0	0	0.8	0.2
2	0	0	0.75	0.25
3	1	0	0	0

Les revenus obtenus après la politique π_1 sont indiqués par :

état	$r(s, \pi_1(s))$
0	2500
1	1500
2	-1000
3	-3000

Résolvons alors le système d'équation :

$$V_\alpha^{\pi_1}(s) = r(s, \pi_1(s)) + \alpha \sum_{s \in S} p(s'|s, \pi_1(s)) V_\alpha^{\pi_1}(s')$$

$$\text{avec } s \in 0, 1, 2, 3$$

La solution simultanée à ce système d'équation rapporte :

– $V_\alpha^{\pi_1}(0) = 6782.18$

– $V_\alpha^{\pi_1}(1) = 1064.36$

– $V_\alpha^{\pi_1}(2) = -1287.13$

– $V_\alpha^{\pi_1}(3) = 2425.74$

L'étape suivante peut être maintenant appliquée. Nous allons chercher une politique améliorée π_2 maximisant les expressions :

$$(0) \qquad r(0, \pi_2(0)) + 0.8 \sum_{s'=0}^{3} p(s'|0, \pi_2(0)) V_\alpha^{\pi_1}(s')$$

$$(1) \qquad r(1, \pi_2(1)) + 0.8 \sum_{s'=0}^{3} p(s'|1, \pi_2(0)) V_\alpha^{\pi_1}(s')$$

$$(2) \qquad r(2, \pi_2(2)) + 0.8 \sum_{s'=0}^{3} p(s'|2, \pi_2(0)) V_\alpha^{\pi_1}(s')$$

$$(3) \qquad r(3, \pi_2(3)) + 0.8 \sum_{s'=0}^{3} p(s'|3, \pi_2(0)) V_\alpha^{\pi_1}(s')$$

Pour trouver $\pi_2(0)$, la meilleure décision quand la machine est dans l'état 0, évaluons la première expression pour toutes les décisions possibles. Rappelons que la probabilité de transition appropriée et les revenus dépendent sur les décisions prises. Pour chaque état $s = \{0, 1, 2, 3\}$ améliorons la politique π_1. Nous obtenons alors le résultat suivant :

– **état 0**

actions	$p(0\|0,a)$	$p(1\|0,a)$	$p(2\|0,a)$	$p(3\|0,a)$	$r(0,a)$	$V_\alpha^{\pi_2}(0)$
1	0.75	0.25	0	0	2500	6782.18
2	0	0.8	0	0.2	3000	4069.31
3	1	0	0	0	0	5425.74

La valeur maximum à l'état 0 est 6782.18.

La décision correspondante à ce maximum est la décision 1.

– **état 1**

actions	$p(0\|1,a)$	$p(1\|1,a)$	$p(2\|1,a)$	$p(3\|1,a)$	$r(1,a)$	$V_\alpha^{\pi_2}(1)$
1	0	0.57	0.29	0.14	500	958.416
2	0	0	0.8	0.2	1500	1064.36
3	1	0	0	0	-1500	3925.74

La valeur maximum à l'état 1 est 3925.74.

La décision correspondante à ce maximum est la décision 3.

– **état 2**

actions	$p(0\|2,a)$	$p(1\|2,a)$	$p(2\|2,a)$	$p(3\|2,a)$	$r(2,a)$	$V_\alpha^{\pi_2}(2)$
1	0	0	0.75	0.25	-1000	-1287.13
2	0	0	0.5	0.5	500	955.446
3	1	0	0	0	-2500	2925.74

La valeur maximum à l'état 2 est 2925.74.

La décision correspondante à ce maximum est la décision 3.

– **état 3**

actions	$p(0\|3,a)$	$p(1\|3,a)$	$p(2\|3,a)$	$p(3\|3,a)$	$r(3,a)$	$V_\alpha^{\pi_2}(3)$
1	0	0	0	1	$-\infty$	$-\infty$
2	0	0	0	1	∞	$-\infty$
3	1	0	0	0	-3000	2425.74

La valeur maximum à l'état 3 est 2425.74.

La décision correspondante à ce maximum est la décision 3.

Ainsi, la politique π_2 obtenue est définie par

– entretenir une machine neuve

$$\pi_2(0) = 1$$

– remplacer si la machine est en bon état

$$\pi_2(1) = 3$$

– remplacer une machine en mauvaise état

$$\pi_2(2) = 3$$

– remplacer une machine en panne

$$\pi_2(3) = 3$$

Comme π_2 n'est pas égale à π_1, on recommence l'algorithme en utilisant la politique π_2. La matrice de transition associée à la politique π_2 est donnée par :

état	0	1	2	3
0	0.75	0.25	0	0
1	1	0	0	0
2	1	0	0	0
3	1	0	0	0

Les revenus obtenus après la politique π_2 sont indiqués par :

état	$r(s, \pi_1(s))$
0	2500
1	-1500
2	-2500
3	-3000

Résolvons alors le système d'équation :

$$V_\alpha^{\pi_2}(s) = r(s, \pi_2(s)) + \alpha \sum_{s \in S} p(s'|s, \pi_2(s)) V_\alpha^{\pi_2}(s')$$

$$\text{avec } s \in 0, 1, 2, 3$$

Voici les solutions de ce système linéaire :

- $V_\alpha^{\pi_2}(0) = 9166.67$
- $V_\alpha^{\pi_2}(1) = 5833.33$
- $V_\alpha^{\pi_2}(2) = 4833.33$
- $V_\alpha^{\pi_2}(3) = 4333.333$

Cherchons alors la politique π_3, voici les résultats des calculs correspondants :

état	$V_\alpha^{\pi_3}(0)$	$V_\alpha^{\pi_3}(1)$	$V_\alpha^{\pi_3}(2)$	$V_\alpha^{\pi_3}(3)$
1	9166.67	4766.67	2766.67	$-\infty$
2	7426.67	5286.67	4166.67	$-\infty$
3	7333.33	5833.33	4833.33	433.33

La valeur maximum à l'état 0 est 9166.67. La décision correspondante à ce maximum est la décision 1.
La valeur maximum à l'état 1 est 5833.33 . La décision correspondante est la décision 3.
La valeur maximum à l'état 2 est 4833.33. La décision correspondante est la décision 3.
La valeur maximum à l'état 3 est 433.33. La décision correspondante est la décision 3.
Nous obtenons ainsi la politique π_3 obtenue est définie par

- entretenir une machine neuve

$$\pi_3(0) = 1$$

- remplacer si la machine est en bon état

$$\pi_3(1) = 3$$

- remplacer une machine en mauvaise état

$$\pi_3(2) = 3$$

- remplacer une machine en panne

$$\pi_3(3) = 3$$

La politique π_3 est identique à la politique π_2 . Puisque les politiques sur deux itérations successives sont identiques, la politique optimale a été obtenue : entretenir la machine à l'état neuve autrement la remplacer.

5.3.2 Résolution par l'Algorithme de "Value Iteration"

Rappelons que cette algorithme consiste à chercher la prochaine fonction de valeur de façon que : pour tout $s \in S$

$$V_{n+1}(s) = \max_{a \in A}\{r(s,a) + \alpha \sum_{s' \in S} p(s'|s,a)V_n(s')\}$$

Choisissons $\epsilon = 0.1$ Et nous avons donc comme test d'arrêt :

$$\|V_{n+1} - V_n\| = \max_{s \in S} |V_{n+1}(s) - V_n(s)| < \epsilon . \frac{1-\alpha}{2\alpha}$$

Ici S=0,1,2,3 et A=1,2,3 et $\alpha = 0.8$.

Posons alors $K = \epsilon . \frac{1-\alpha}{2\alpha} = 0.0125$ le test d'arrêt.

La première étape dans cette algorithme est de choisir arbitrairement un ensemble de valeurs $V_0(0), V_0(1), V_0(2), V_0(3)$.

A l'itération 0, soient alors

$$V_0(0) = 0$$

$$V_0(1) = 0$$

$$V_0(2) = 0$$

$$V_0(3) = 0$$

Nous obtenons alors à l'itération 1 :

$$V_1(0) = \max_{a \in A}\{r(0,a)\} = 3000$$

$$V_1(1) = \max_{a \in A}\{r(0,a)\} = 1500$$

$$V_1(2) = \max_{a \in A}\{r(0,a)\} = 500$$

$$V_1(3) = \max_{a \in A}\{r(0,a)\} = 3000$$

Ainsi la première approximation réclame de prendre la décision 2 (sans maintenance) quand la machine est dans l'état 0,1 ou 2. Quand la machine est dans l'état 3, la décision 3 (remplacement) est prise.

	$V_1(s)$	$V_0(s)$	$\|V_1(s) - V_0(s)\|$
s=0	3000	0	3000
s=1	1500	0	1500
s=2	500	0	500
s=3	3000	0	3000

D'après le tableau précédent, nous avons $\|V_1(s) - V_0(s)\| = 3000$ est supérieur à $K = 0.0125$, nous continuons l'algorithme.

La deuxième itération nous donne :

$$V_2(0) = \max\{2500 + 0.8[\frac{3}{4}(3000) + \frac{1}{4}(1500)],$$
$$3000 + 0.8[\frac{4}{5}(1500) + \frac{1}{5}(-3000)], 0.8[1(3000)]\}$$
$$= \max\{4600, 3480, 2400\} = 4600$$

$$V_2(1) = \max\{500 + 0.8[\frac{4}{7}(1500) + \frac{2}{7}(500) + \frac{1}{7}(-3000)],$$
$$1500 + 0.8[\frac{4}{5}(500) + \frac{1}{5}(-3000)], -1500 + 0.8[1(3000)]\}$$
$$= \max\{957, 1340, 900\} = 1340$$

$$V_2(2) = \max\{-1000 + 0.8[\frac{3}{4}(500) + \frac{1}{4}(-3000)],$$
$$500 + 0.8[\frac{1}{2}(500) + \frac{1}{2}(-3000)], -2500 + 0.8[1(3000)]\}$$
$$= \max\{-1300, -500, -100\} = -100$$

$$V_2(3) = \max\{-\infty + 0.8[1(-3000)]$$
$$-\infty + 0.8[1(-3000)], -3000 + 0.8[1(3000)]\}$$
$$= -600$$

Cette deuxième approximation réclame à entretenir la machine comme elle est dans l'état 0, la laisser sans maintenance quand elle est dans l'état 1, la remplacer quand elle est dans l'état 2 ou 3.

	$V_2(s)$	$V_1(s)$	$\lvert V_2(s) - V_1(s)\rvert$
s=0	4600	3000	1600
s=1	1340	1500	-160
s=2	-100	500	-600
s=3	-600	3000	-3600

Comme précédement, nous avons $\|V_2(s) - V_1(s)\| = 1600$ est supérieure à $K = 0.0125$, on continue l'algorithme.

A l'itération 3, nous obtenons la politique telle que : entretenir la machine comme elle est dans l'état 0, sinon la remplacer dans les autres états tel que :

	$V_3(s)$	$V_2(s)$	$\lvert V_3(s) - V_2(s)\rvert$
s=0	5528	4600	928
s=1	2180	1340	840
s=2	1180	-100	1280
s=3	680	-600	1280

Nous avons toujours $\|V_3(s) - V_2(s)\| = 1280 > K = 0.0125$, nous continuons l'algorithme.

A l'itération 4, nous obtenons la même politique qu'à l'itération 3, c'est-à-dire entretenir la machine comme elle est dans l'état 0, autrement la remplacer tel que :

	$V_4(s)$	$V_3(s)$	$\lvert V_4(s) - V_3(s)\rvert$
s=0	6252.8	5528	724.8
s=1	2922.4	2180	742.4
s=2	1922.4	1180	742.4
s=3	1422.4	680	742.4

Nous avons toujours $\|V_4(s) - V_3(s)\| = 742.4 > K = 0.0125$, on continue l'algorithme.

....

A l'itération 54, obtient la même politique qu'à l'itération 3 tel que :

	$V_{54}(s)$	$V_{53}(s)$	$\lvert V_{54}(s) - V_{53}(s)\rvert$
s=0	9166.63	9166.62	0.01
s=1	5833.29	5833.28	0.01
s=2	4833.29	4833.28	0.01
s=3	4333.29	4333.28	0.01

En outre nous avons $\|V_{54}(s) - V_{53}(s)\| = 0.01 < K = 0.0125$. Nous stoppons donc l'algorithme. Finalement, nous obtenons la solution optimale qui est d'entretenir l'auto quand elle dans l'état neuf autrement le remplacer. Le coût total pondéré maximum attendu du système au cours des 54 périodes, si le système commence à l'état , est donné par 9166.63, 5833.29, 4833.29, et 4333.29 respectivement.

5.3.3 Résolution par la programmation linéaire

Nous avons montré que la déclaration de problème de la programmation linéaire est de choisir $y(s,a)$ qui maximise

$$\sum_{a\in A}\sum_{s\in S} r(s,a)y(s,a)$$

sous contraintes

$$\sum_{a\in A} y(s',a) - \alpha \sum_{a\in A}\sum_{s\in S} p(s'|s,a)y(s,a) \leq \beta(s')$$

pour $s' \in S$ et

$$y(s,a) \geq 0 \quad \forall s \in S, a \in A$$

telle que la politique optimale sortante est :

$$\pi(s,a) = \frac{y(s,a)}{\sum_{a\in A} y(s,a)}$$

Nous avons donc la formulation du problème par le programme linéaire suivant :

$$\begin{aligned} maximiser & 2500y(0,1) + 3000y(0,2) + 0(y,0,3) \\ & + 500y(1,1) + 1500y(1,2) - 1500y(1,3) \\ & - 1000y(2,1) + 500y(2,2) - 2500y(2,3) \\ & - c_1 y(3,1) - c_2 y(3,2) - 3000y(3,3) \end{aligned}$$

(où c_1 et c_2 sont pris comme des nombres assez grands)

sous contraintes :

$$\sum_{k=1}^{3} y(0,k) - 0.8[0.75y(0,1) + y(0,3) + y(1,3) + y(2,3) + y(3,3)] \leq \frac{1}{4}$$

$$\sum_{k=1}^{3} y(1,k) - 0.8[0.25y(0,1) + 0.8y(0.2) + 0.57y(1,1)] \leq \frac{1}{4}$$

$$\sum_{k=1}^{3} y(2,k) - 0.8[0.29y(1,1) + 0.8y(1,2) + 0.75y(2,1) + 0.5y(2,2)] \leq \frac{1}{4}$$

$$\sum_{k=1}^{3} y(3,k) - 0.8[0.2y(0,2) + 0.14y(1,1) + 0.2y(1,2) + 0.25y(2,1) + 0.5y(2,2)$$
$$+ y(3,1) + y(3,2)] \leq \frac{1}{4}$$

Ainsi la solution du programme linéaire donne :

y(0,1)=3.542	y(0,2)=0	y(0,3) =0
y(1,1)=0	y(1,2)=0	y(1,3)=0.958
y(2,1)=0	y(2,2)=0	y(2,3)=0.25
y(3,1)=0	y(3,2)=0	y(3,3)=0.25

Calculons alors $\sum_{a=1}^{3} y(s,a)$ pour chaque état s :

$$\sum_{a=1}^{3} y(0,a) = 3.54167$$
$$\sum_{a=1}^{3} y(1,a) = 0.9583333$$
$$\sum_{a=1}^{3} y(2,a) = 0.25$$
$$\sum_{a=1}^{3} y(3,a) = 0.25$$

Nous obtenons la probabilité d'être à l'état s en prenant la décision a :

$\pi(0,1) = 1$	$\pi(0,2) = 0$	$\pi(0,3) = 0$
$\pi(1,1) = 0$	$\pi(1,2) = 0$	$\pi(1,3) = 1$
$\pi(2,1) = 0$	$\pi(2,2) = 0$	$\pi(2,3) = 1$
$\pi(3,1) = 0$	$\pi(3,2) = 0$	$\pi(3,3) = 1$

La politique optimale sortant est donc d'entretenir la machine quand elle est neuve autrement la remplacer.

Conclusion

Cette approche présentée a le grand intérêt de permettre au responsable de la machine de trouver la politique optimale de gérer la dégradation de la machine. En utilisant les deux méthodes : programmation linéaire et programmation dynamique, nous avons pu prouver que la politique optimale est la même : entretenir la machine quand elle est neuve autrement la remplacer. Cependant, nous avons trouvé la rapidité de l'algorithme de "Policy itération" avec 3 itérations par rapport à "Value itération" qui a nécessité 53 itérations. Mais chaque itération de "Value itération" est très rapide sans le souci de résoudre des systèmes linéaires. De même la résolution par la programmation linéaire est plus chère en temps de calcul, cependant elle nous mène aussi à la solution optimale.

Conclusions générales

Pour conclure ce mémoire, nous commencerons par un bref résumé soulignant les apports, avant d'indiquer les différentes perspectives que nous aimerions développer.

Nous avons présenté dans ce rapport une étude de l'utilisation de modèles stochastiques pour planifier les décisions ou actions à prendre évoluant dans un environnement aléatoire ou déterministes. Notre apport dans le cadre des MDP se situe à plusieurs niveaux :

- dans le chapitre 2, nous avons montré l'importance de chaque critère étudié en MDP : le critère fini qui est facile à étudié, le critère $\alpha-$pondéré qui présente des équations d'optimalité assez robuste et enfin le critère moyen qui est le plus complexe.
- dans le chapitre 3 et 4, nous avons étudié les méthodes de résolution des MDP. La formulation en programmation linéaire nous mène à résoudre le MDP par l'algorithme de simplexe. La programmation dynamique a montré les 2 algorithmes les plus utilisés pour résoudre le MDP. Néanmoins, le résultat reste le même quelque soit l'algorithme utilisé. Mais nous avons pu constater l'ordre de complexité de chaque algorithme.
- dans le chapitre 5, nous avons présenté une application des MDP numériquement. Nous avons pu confirmé les cadres théoriques présentés dans le chapitre 2 et 3, à savoir la convergence des algorithmes de la programmation dynamique. Nous avons pu trouver aussi la politique optimale par la programmation linéaire.

Les Processus décisionnels de Markov constituent donc un puissant outil de modélisation dans le domaine économique. D'où la thématique de recherche de ce mémoire de DEA s'est avérée très intéressante et le travail accompli a été égale-

ment très enrichissant pour moi.

Toutefois, le cadre théorique que nous avons exposé au cours de ce mémoire n'est pas exempt de limites en termes théoriques. D'abord, nous avons supposé que l'agent est en parfaite connaissance des fonctions de transition et de récompense qui définissent le problème auquel il est confronté. Ensuite, nous avons supposé qu'il n'y a qu'un seul agent au niveau du décideur. De plus, pour des environnements de taille complexe, les algorithmes que nous avons utilisés pour la résolution des MDP ont un temps de calcul très important, ce qui les rend difficilement adaptables dans le cadre d'une application en temps-réel.
De ce fait, de nombreuses extensions se présentent à nous. D'une part, nous pouvons notamment développer par la suite au niveau du décideur, la possibilité de mettre plusieurs agent pour choisir la stratégie : le MDP à multi-agents. En effet, nombreux problèmes requièrent la modélisation de plusieurs agents évoluant et agissant ensemble au sein du même environnement. D'autre part, nous pouvons entrer dans les Processus Décisionnels de Markov Partiellement Observables (POMDP) permettant de modéliser et contrôler les systèmes dynamiques incertains dont l'état n'est que partiellement connu. Et enfin, nous pouvons dépasser les limites des algorithmes de programmation dynamique par d'autres méthodes présentées dans [19], nous pouvons citer par exemple : les représentations factorisées, l'optimisation de politiques paramétrées ou encore l'optimisation de décision en ligne.

Annexe

La programmation dynamique

La programmation dynamique est une méthodologie générale pour concevoir des algorithmes qui permettent de résoudre efficacement certains problèmes d'optimisation. Un problème d'optimisation admet un grand nombre de solutions. Chaque solution a une certaine valeur, et on veut identifier une solution dont la valeur est optimale (minimale ou maximale). Elle a été désignée par ce terme pour la première fois dans les années 1940 par Richard Bellman.

Principe

La programmation dynamique s'appuie sur un principe simple : toute solution optimale s'appuie elle-même sur des sous-problèmes résolus localement de façon optimale. Concrètement, cela signifie que l'on va pouvoir déduire la solution optimale d'un problème en combinant des solutions optimales d'une série de sous problèmes. Les solutions des problèmes sont étudiées "de bas en haut", c'est-à-dire qu'on calcule les solutions des sous-problèmes les plus petits pour ensuite déduire petit à petit les solutions de l'ensemble.

Code source Programme sur Dev C++

Résolution de MDP par l'algorithme de "Policy Iteration"

Code source C++

```
#include <stdio.h>
#include <conio.h>
```

```
#include <math.h>
#include <string.h>
#include <stdlib.h>
#include <iostream.h>
#include <process.h>
#include <stdlib.h>
#include <conio.h>
void main()
{
int x0=0;
float alpha;
int x,d,M,i,j,k,n,p;
int D[100];
double e[100][100],s[100], P[100][100], P0[100][100], C0[100][100],C[100],Z[100][100];
int V[100];
double ee,aa;
double init[100],B[100],W[100],Q[100][100],D2[100];
int kd[100];
system ("cls");
system ("color F1");
cout«"\n\n";
cout«"\n\n\t\t\t MDP ALGORITHME DE POLICY ITERATION";
cout«"\n\n\n";
cout«" Donner le nombre de decision d=";
cin»d;
cout«"\n\n";
cout«" Donner le nombre d etat M=";
cin»M;
cout«"\n \n";
cout«" Donner la valeur de alpha=";
cin»alpha;
cout«"\n \n";
```

```
cout«"Donner la politique a prendre : \n\n";
for(p=0;p<M;p++) {
cout«"Decision a l'etat "«p«" = ";
cin»D[p];
cout«"\n\n"; }
cout«"\n\n\t\t\t Entrer la matrice contenant tous les couts\n";
for(p=0;p<M;p++) { cout«"\n\n\t\t\t Etat"«p«"\n";
for(i=0;i<d;i++) {
cout«"C0["«i«"]["«p«"]=";
cin»C0[i][p]; }
cout«"\n"; }
cout«"\n";
cout«"C0[i][j]=";
cout«"\n";
for (i=0;i<d;i++) {
for (j=0;j<M;j++) {
cout«C0[i][j]«" "; }
cout«"\n"; }
cout«"\n";
int M2=M*M;
cout«"\t Entrer tous les matrices de transition \n";
for (i=0;i<d;i++) {
for (j=0;j<M2;j++) {
cout«"P0["«i«"]["«j«"]=";
cin»P0[i][j]; } }
cout«"\n";
cout«"P0[i][j]=";
cout«"\n";
for (i=0;i<d;i++) {
for (j=0;j<M2;j++) {
cout«P0[i][j]«" "; }
cout«"\n"; }
```

```
int k2=0 ;
do
{
cout«"\n \n \t \t Entrer la matrice de transition de cette politique : \n \n" ;
for (i=0 ;i<M ;i++) {
for ( j=0 ;j<M ;j++) {
cout«"P["«i«"]["«j«"]= " ;
cin»P[i][j] ; } }
cout«"\n" ;
for (i=0 ;i<M ;i++)
{
for ( j=0 ;j<M ;j++) {
cout«P[i][j] ;
cout«" " ; }
cout«"\n" ; }
cout«"\n" ;
for ( i=0 ;i<M ;i++) {
cout«"C["«i«"]=" ;
cin»C[i] ; }
cout«"\n" ;
for ( i=0 ;i<M ;i++) {
cout«C[i] ;
cout«"\n" ; }
cout«"\n" ;
for ( i=0 ;i<M ;i++) {
for( j=0 ;j<M ;j++) {
if (i==j) { Z[i][j]=1-(alpha*P[i][j]) ; }
else { Z[i][j]= -(alpha*P[i][j]) ; } } }
for ( i=0 ;i<M ;i++) {
Z[i][M]=C[i] ; }
n=M ;
cout«"\n" ;
```

```
for (i=0 ;i<n ;i++) {
for (p=0 ;p<n ;p++) {
e[p][i]=Z[i][p] ; }
e[n][i]=Z[i][n] ; }
int y=0 ;
double var1=0,var2=0 ;
double temp ;
int a,t ;
for(int x=0 ;x<n-1 ;x++){
for(a=1+x ;a<n ;a++){
temp=e[x][a] ;
for (t=x ;t<n+1 ;t++){
e[t][a]=e[t][a]*e[x][x]-e[t][x]*temp ; } } }
int af ;
s[n-1]=e[n][n-1]/e[n-1][n-1] ;
e[n][n-1]=0 ;
e[n-1][n-1]=0 ;
for (int ligne=1 ;ligne<=n ;ligne++){
for (int sol=2 ;sol<=n ;sol++){
e[n-ligne][n-sol]*=s[n-ligne] ;
e[n][n-sol]-=e[n-ligne][n-sol] ;
e[n-ligne][n-sol]=0 ;
}
s[n-(ligne+1)]=e[n][n-(ligne+1)]/e[n-(ligne+1)][n-(ligne+1)] ;
}
cout«"\t V 0 = "«s[0]«"\n\n" ;
for (af=1 ;af<n ;af++) {
cout«"\t V "«af«" = "«s[af]«"\n\n" ;
}
for(i=0 ;i<M ;i++) {
B[i]=s[i] ; }
for (p=0 ;p<M ;p++) {
```

```
cout«"\n\n\t\t\t State"«p«"\n";
for(i=0;i<d;i++) {
init[i]=0;
for(j=0;j<M;j++) {
init[i]=init[i]+P0[i][(M*p)+j]*B[j]; } }
cout«"calcul des valeurs de chaque expression \n";
for (j=0;j<d;j++) {
W[j]=C0[j][p]+alpha*init[j];
cout« W[j]«"\n"; }
cout« "\n";
for (i=0;i<d;i++) {
D2[i]=i+1; }
for (i=0;i<d;i++) {
Q[i][0]=D2[i];
Q[i][1]=W[i]; }
cout«" La 1ere colonne la decision, la 2nde la valeur correspondant :\n";
for (i=0;i<d;i++) {
for (j=0;j<2;j++) {
cout«Q[i][j];
cout«" "; }
cout«"\n"; }
cout«"\n";
aa=Q[0][1];
for (i=0;i<d-1;i++) {
if (aa>Q[i+1][1]) { ee=aa; }
else {
ee=Q[i+1][1]; }
aa=ee; }
for (i=0;i<d;i++) {
if(ee==Q[i][1]) {
k=i+1; } }
cout«"Le maximum dans l'etat " « p«" est : " « ee;
```

```
cout«"\n\n";
cout«"la decision est : "«k;
cout«"\n\n";
kd[p]=k;
}
cout«"\n";
cout«"voici les decisions a chaque etat : \n ";
for (p=0;p<M;p++) {
cout«"\n";
cout«kd[p];
cout«"\n"; }
cout«"\n";
int E[100],x1,max0;
for (p=0;p<M;p++) {
if (kd[p]!=D[p]) {
x1=x1+1; }
else
{
x1=0; }
E[p]=x1; }
max0= E[0];
for (p=0;p<M-1;p++) {
if (max0>E[p+1]) {
x0=max0; }
else
{
x0=E[p+1]; }
max0=x0; }
cout«"\n\n\t\t ITERATION Numero : "«k2+1;
cout«"\n\n\t\t La politique obtenue est : \n";
for (p=0;p<M;p++) {
D[p]=kd[p];
```

```
cout«"La decision a l'etat "«p«" est la decision : "«D[p]«"\n";
}
cout«"\n";
k2=k2+1;
}
while(x0!=0);
cout«"\n\n\t\t VOICI LA POLITIQUE OPTIMALE : \n";
cout«"\n";
for (p=0;p<M;p++) {
cout«"Decision a l'etat "«p«" = Decision "«D[p]«"\n";
}
cout«"\n";
getch();
return(0);
}
```

Résolution de MDP par l'algorithme de "Value Iteration"

Code source C++

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>
#include <iostream.h>
#include <process.h>
#include <stdlib.h>
#include <conio.h>
void main()
{
int x0=0;
float alpha,epsilon,z[100],norme,max0;
```

```
int x,d,M,i,j,k,n,p,k2;
double E;
int D[100]; double P[100][100],C[100][100],Z[100][100], V[100][100], ee[100];
double aa;
double init[100],W[100],Q[100][100],D2[100]; int kd[100];
system ("cls");
system ("color F1");
cout«"\n\n\t\t\t MDP ALGORITHME DE VALUE ITERATION";
cout«"\n\n\n";
cout«" Donner le nombre de decision d=";
cin»d;
cout«"\n\n";
cout«" Donner le nombre d etat M=";
cin»M;
cout«"\n\n";
cout«" Donner la valeur de alpha=";
cin»alpha;
cout«"\n\n";
cout«" Donner la valeur de epsilon=";
cin»epsilon;
cout«"\n\n";
int M2=M*M;
for(i=0;i<M;i++)
{
V[i][0]=0;
}
E=epsilon*((1-alpha)/(2*alpha));
k2=0;
cout«"\n\n\t\t\t Entrer la colonne du cout\n";
for(p=0;p<M;p++)
{ cout«"\n\n\t\t\t Etat"«p«"\n";
for(i=0;i<d;i++) {
```

```
cout«"C["«i«"]["«p«"]=";
cin»C[i][p]; }
cout«"\n"; }
cout«"C[i][j]=";
cout«"\n";
for (i=0;i<d;i++) {
for (j=0;j<M;j++) {
cout«C[i][j]«" "; }
cout«"\n"; }
cout«"\t Entrer la matrice de transition \n";
for (i=0;i<d;i++) {
for (j=0;j<M2;j++)
    {
cout«"P["«i«"]["«j«"]=";
cin»P[i][j]; } }
cout«"\n";
cout«"P[i][j]=";
cout«"\n";
for (i=0;i<d;i++) {
for (j=0;j<M2;j++) {
cout«P[i][j]«" "; }
cout«"\n"; }
do
{
for (p=0;p<M;p++)
{ cout«"\n\n\t\t\t Etat"«p«"\n";
for(i=0;i<d;i++)
{
init[i]=0;
for(j=0;j<M;j++) {
init[i]=init[i]+P[i][(M*p)+j]*V[j][k2];
} }
```

```
cout«"calcul des valeurs de chaque expression \n" ;
for (j=0 ;j<d ;j++) {
W[j]=C[j][p]+alpha*init[j] ;
cout« W[j]«"\n" ; }
cout« "\n" ;
for (i=0 ;i<d ;i++) {
D2[i]=i+1 ; }
for (i=0 ;i<d ;i++) {
Q[i][0]=D2[i] ;
Q[i][1]=W[i] ; }
cout«" La 1ere colonne la decision, la 2nde la valeur correspondante :\n" ;
for (i=0 ;i<d ;i++) {
for (j=0 ;j<2 ;j++) {
cout«Q[i][j] ;
cout«" " ; }
cout«"\n" ; }
cout«"\n" ;
aa=Q[0][1] ;
for (i=0 ;i<d-1 ;i++) {
if (aa>Q[i+1][1]) {
ee[p]=aa ;

    }
else
{
ee[p]=Q[i+1][1] ;
}
aa=ee[p] ;
}
for (i=0 ;i<d ;i++) {
if(ee[p]==Q[i][1]) {
k=i+1 ;
```

```
}
}
cout«"Le maximum dans l'etat " « p«" est : " « ee[p] ;
cout«"\n\n" ;
cout«"la decision est : "«k ;
cout«"\n\n" ;
kd[p]=k ;
}
for (p=0 ;p<M ;p++) {
z[p]= fabs(ee[p]-V[p][k2]) ;
}
max0= z[0] ;
for (p=0 ;p<M-1 ;p++) {
if (max0>z[p+1]) {
norme=max0 ; }
else {
norme=z[p+1] ; }
max0=norme ;
}
cout«"\n norme= " ;
cout«norme ;
cout«"\n" ;
cout«"\n E= " ;
cout«E ;
cout«"\n" ;
if (norme>E) {
x0=x0+1 ; }
else
{
x0=0 ; }
cout«"\n x0=" ;
cout«x0 ;
```

```
cout«"\n\n\t\t ITERATION Numero : "«k2+1 ;
cout«"\n\n\t\t La politique obtenue est : \n" ;
for (p=0 ;p<M ;p++)
{
D[p]=kd[p] ;
cout«"La decision a l'etat "«p«" est la decision : "«D[p]«"\n" ;
cout«"La valeur a l'etat "«p«" est : "«ee[p]«"\n" ;
}
for (p=0 ;p<M ;p++) {
V[p][k2+1]=ee[p] ;
}
k2=k2+1 ;
getch() ;
}
while(x0 !=0) ;
cout«"\n\n\t\t VOICI LA POLITIQUE OPTIMALE : \n" ;
for (p=0 ;p<M ;p++)
{
cout«"Decision a l'etat "«p«" = Decision "«D[p]«"\n" ;
}
cout«"\n" ;
getch() ;
return(0) ;
}
```

Résolution de MDP par l'algorithme de Simplexe

Code source C++

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <string.h>
```

```
#include <stdlib.h>
#include <iostream.h>
#include <process.h>
#include <stdlib.h>
#include <conio.h>
#define NMAX 100
#define MMAX 100
#define VARMAX 20
int simplexe() ;
int entrant(double a[MMAX][NMAX],int hb[NMAX],int m,int n,int phase) ;
int sortant(double a[MMAX][NMAX],int m,int k) ;
void pivotage(double a[MMAX][NMAX],int db[MMAX],int hb[NMAX], int m,int
n,int l,int k) ;
void affich(double a[MMAX][NMAX],int db[MMAX],int hb[NMAX],
int m,int n,int phase) ;
int simplexe_calcul(double a[MMAX][NMAX],double sol[VARMAX], int ineq1,int
ineq2, int eq,int n) ;
void main()
{
int x=1 ;
do
{
system ("cls") ;
system ("color F1") ;
simplexe() ;break ;
}
while(x !=0) ;
}
int resolutionSimplexe()
{
system("cls") ;
int i,j,ineq1,ineq2,eq,n,err ;
```

```
int k=1 ;
double a[MMAX][NMAX],sol[VARMAX] ;
cout«"\n\t\t\t Resolution MDP par la Methode Simplexe" ;
cout«"\n\n\t\t\t Max z = " ;
cout«"\n\t\t\t S.C : <= " ;
double y[100] ;
int u,a0,MI,DI,p ;
double som[100] ;
cout«"\n" ;
cout«"\n" ;
cout«"Donner le nombre d etat : " ;
cin»MI ;
cout«"\n" ;
cout«"Donner le nombre de decision : " ;
cin»DI ;
cout«"\n" ;
n=MI*DI ;
printf("Donner les elements de la fonction economique : \n") ;
a[0][0]=0 ;
for(k=1 ;k<=n ;k++) {
cout«"\n a[0]["«k«"] = " ;
cin»a[0][k] ;
}
printf("Donner le nombre d'equations en <= : ") ;
cin»ineq1 ;
printf("Donner les elements l'equations en <= :\n") ;
for(i=1 ;i<=ineq1 ;i++) {
for(j=0 ;j<=n ;j++) {
printf("\na[%d][%d] = ",i,j) ;
cin»a[i][j] ;
} }
ineq2=0 ;
```

```
eq=0 ;
err=simplexe_calcul(a,sol,ineq1,ineq2,eq,n) ;
if(err==1)printf("Solution infinie\n") ;
else
if(err==2)printf("Domaine vide\n") ;
else
{
printf("\n Solution optimale :\n\n") ;
for(i=1 ;i<=ineq1+ineq2+n ;i++)
{
cout«"x["«i«"]= "«sol[i] ;
cout«"\n" ; }
printf("\nValeur optimale : z=%23.16e\n",-a[0][0]) ;
for(p=0 ;p<MI ;p++) {
som[p]=0 ;
for(a0=0 ;a0<DI ;a0++) {
u=(p*DI)+a0+1 ;
som[p]=som[p]+sol[u] ; //som sur k de y(i,k) }
cout«"\n" ; }
double S[100][100] ;
for (p=0 ;p<MI ;p++) {
cout«"etat : "«p ;
cout«"\n" ;
for (i=0 ;i<DI ;i++) {
u=(p*DI)+i ;
S[p][i]=som[p] ;
cout«"S["«p«"]["«i«"]="« S[p][i] ;
cout«"\n" ; } }
double opt[100][100] ;
cout«"\n" ;
cout«"\t\t Voici la politique optimale : " ;
cout«"\n" ;
```

```
for (p=0 ;p<MI ;p++) {
for (i=0 ;i<DI ;i++) {
u= (p*DI)+i+1 ;
opt[p][i]=sol[u]/S[p][i] ;
cout«"la prob a l etat "«p«" en prenant la decision "«i«" est : "«opt[p][i] ;
cout«"\n" ; }
cout«"\n" ; } }
return (0) ;
}
int entrant(double a[MMAX][NMAX],int hb[NMAX],int m,int n,int phase)
{
int i,j,k,l ;
double d,s,max ;
k=0 ;
max=0.0 ;
if(phase==2)l=0 ;
else l=m+1 ;
for(j=1 ;j<=n ;j++) {
d=a[l][j] ;
s=0.0 ;
if((d>0)&&(hb[j] !=n+m)) {
for(i=1 ;i<=m ;i++) s+=fabs(a[i][j]) ;
d/=s ;
if(d>max)
{
max=d ;
k=j ; } } }
return(k) ;
}
int sortant(double a[MMAX][NMAX],int m,int k)
{
int i,l ;
```

```
double rap,min ;
min=1e308 ; //min= 0+
l=0 ;
for(i=1 ;i<=m ;i++) if(a[i][k]>0) {
rap=a[i][0]/a[i][k] ;
if(rap<min)
{
min=rap ;
l=i ;
} }
return(l) ;
}
void pivotage(double a[MMAX][NMAX],int db[MMAX],int hb[NMAX], int m,int
n,int l,int k)
{
int i,j ;
double pivot,coef ;
pivot=a[l][k] ;
for(i=0 ;i<=m ;i++) if(i !=l) {
coef=a[i][k]/pivot ;
a[i][k]=-coef ;
for(j=0 ;j<=n ;j++)
if(j !=k) //
a[i][j]=a[i][j]-coef*a[l][j] ;
}
coef=1/pivot ;
a[l][k]=coef ;
for(j=0 ;j<=n ;j++) if(j !=k) a[l][j]=coef*a[l][j] ;
i=db[l] ;
db[l]=hb[k] ;
hb[k]=i ;
} int simplexe_calcul(double a[MMAX][NMAX],double sol[VARMAX], int ineq1,int
```

```
ineq2, int eq,int n)
{
int i,j,k,l,phase,m,m1 ;
int db[MMAX],hb[NMAX] ;
double min ;
m=ineq1+ineq2+eq ;
for(i=ineq1+1 ;i<=ineq1+ineq2 ;i++)
for(j=0 ;j<=n ;j++)
a[i][j]=-a[i][j] ;
for(i=1 ;i<=ineq1+ineq2 ;i++)
db[i]=n+i ;
for(i=ineq1+ineq2+1 ;i<=m ;i++)
db[i]=0 ;
for(j=1 ;j<=n ;j++)
hb[j]=j ;
if(eq !=0)
{
for(i=ineq1+ineq2+1 ;i<=m ;i++) {
l=i ;
k=0 ;
for(j=1 ;j<=n ;j++) if(a[i][j] !=0)k=j ;
if(k==0)
{ if(a[i][0] !=0)return(2) ; }
else
{
printf("var.entrante : x%d\n",hb[k]) ;
pivotage(a,db,hb,m,n,l,k) ;
hb[k]=hb[n] ;
for(j=0 ;j<=m ;j++) a[j][k]=a[j][n] ;
n-=1 ;
} } }
n+=1 ;
```

```
m1=m ;
hb[n]=n+m ;
phase=2 ;
l=0 ;
min=0 ;
for(i=1 ;i<=m ;i++)
if(a[i][0]<min)
{
min=a[i][0] ;
l=i ;
}
if(l !=0)phase=1 ;
k=1 ;
if(phase==1)
{
m1=m+1 ;
for(j=0 ;j<n ;j++)
a[m1][j]=0 ;
for(i=1 ;i<=m ;i++)
if(a[i][0]<0)
a[i][n]=-1 ;
else a[i][n]=0 ;
a[0][n]=0 ;
a[m1][n]=-1 ;
pivotage(a,db,hb,m1,n,l,n) ;
}
while(phase<=2) {
do {
k=entrant(a,hb,m,n,phase) ;
if(k !=0)
{
l=sortant(a,m,k) ;
```

```
if(l==0)return(1) ;
pivotage(a,db,hb,m1,n,l,k) ;
} }
while(k !=0) ;
if(phase==1)
{
l=0 ;
for(i=1 ;i<=m ;i++)
if(db[i]==n+m)l=i ;
if(l !=0)
{
if(fabs(a[l][0])>1e-15)return(2) ;
else
{
for(j=1 ;j<=n ;j++) if(a[l][j] !=0)
k=j ;
pivotage(a,db,hb,m1,n,l,k) ; } } }
phase+=1 ;
m1-=1 ;
}
for(i=1 ;i<m+n ;i++) sol[i]=0 ;
for(i=1 ;i<=m ;i++) sol[db[i]]=a[i][0] ;
return(0) ;
}
int simplexe()
{
resolutionSimplexe() ;
system("PAUSE") ;
return(0) ;
}
```

Bibliographie

[1] Antonini C., J.-F. Quint, P. Borgnat, *Les Mathematiques pour l'Agregation*, 2002.

[2] Aurelien ESNARD, *Notes de Cours ENSEIRB Informatique et Recherche operationnelle*, 2000.

[3] Bellman, *Dynamic programming*, Princeton University Press, Princeton, 1957.

[4] Bernoussi S. El, *Programmation lineaire-Methode du simplexe*, 2010.

[5] Blackwell, D. and T. Ferguson , *Annals of Mathematical Statistics* 39, 1968.

[6] Ferland, *Dualite en programmation lineaire*, 2012.

[7] Gregory VIAL - Eric DARRIGRAND , *Algebre lineaire numerique*, 2007.

[8] Harison Victor, *Cours Probabilité- Maîtrise Mathématique*, Université d'Antananarivo, 2010.

[9] Harison Victor, *Cours Fondements mathématiques de la Finance-AEA Mathématique*, Université d'Antananarivo, 2011.

[10] Jean François & Delmas- Benjamin Jourdain, *Modèles aléatoires pour l'ingenieur*, 2003.

[11] Julien Amédée RABOANARY, *Cours optimisation-AEA Mathématiques*, Université d'Antananarivo, 2011-2012.

[12] Lodewijk KALLENBERG,*Markov Decision Processes*, University of Leiden.

[13] Littman M., *On the complexity of solving Markov decision problems*, Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence , Montreal, Québec, Canada, 1995.

[14] Marc TOUSSAINT, *Markov Decision Processes*, Machine Learning & Robotics group, Berlin-Germany, 2009.

[15] Puterman, *Markov decision processes*, Wiley, New York, 1994.

Webographie

[16] http ://fr.wikipedia.org/wiki/Chaîne_de_Markov.

[17] http ://fr.wikipedia.org/wiki/Processus_de_Markov.

[18] http ://fr.wikipedia.org/wiki/Processus_stochastique.

[19] http ://www.damas.ift.ulaval.ca/ coursMAS/ComplementsH10/MDP-Garcia-PDMIA.pdf

[20] http ://www.math.tau.ac.il/ mansour/rl-course.

Résumé

Titre : Contribution à l'analyse du Processus Décisionnel de Markov et ses algorithmes de résolution

Dans un environnement stochastique, nous cherchons à modéliser la dynamique de l'état d'un système soumis au contrôle d'un agent. Le processus décisionnel de Markov connu sous l'acronyme MDP nous permet cette modélisation. Nous nous sommes intéressés ainsi à une étude théorique des Processus Décisionnels de Markov et à la résolution de ceux-ci. Cette résolution consiste à trouver la politique optimale : minimiser les coûts ou maximiser les gains.

Nous avons deux méthodes pour résoudre le problème des MDP. D'une part, comme le MDP peut être formulé en programmation linéaire, la méthode de simplexe garantit donc sa résolution. D'autre part, les algorithmes de la programmation dynamique comme "Policy iteration" et "Value iteration" peuvent être identifiés pour déterminer la politique optimale dans un MDP. Nous avons pu montrer les limites et les avantages de ces approches. Tous ces algorithmes ont été implémentés sur le langage de programmation C++. Le code source est disponible dans l'annexe de ce mémoire.

En guise d'illustration de tout ceci, nous avons ainsi pu appliquer le MDP à un problème d'économie : la gestion de maintenance de machine pour maximiser le revenu de production d'une entreprise.

Par ailleurs, de nombreuses perspectives se présentent à nous, nous pouvons citer par exemple la possibilité de mettre plusieurs agents décideurs lors de l'exécution de l'action mais aussi l'amélioration des algorithmes de résolution face à des environnements assez grands.

Mots-clés : fonction de valeur, horizon fini et infini, méthode du simplexe, politique optimale, programmation dynamique, programmation linéaire, Processus Décisionnels de Markov, "Policy Iteration", "Value Iteration".

Abstract

Title : Contribution to the analysis of the Markovian Decision Processes and its algorithms of resolution.

In a stochastic environment, we seek to model dynamics state of a system subjected to the control of an agent. The Markovian Decision Processes known as the MDP acronym allows us this modelisation. Thus, we are interested in a theoretical study of the Markovian Decision Processes and its resolution. This resolution consists of finding the optimal policy : minimize the costs or maximize the profits.

We have two methods to solve the problem of the MDP. The first one, as the MDP can be formulated in linear programming, method of simplex guarantees thus its resolution. The second one, algorithms of the dynamics programming like "Policy iteration" and "Value iteration" can be identified to determine the optimal policy in a MDP. We have been able to show the limits and the advantages of these approaches. All these algorithms were implemented using the programming language C++. The source code is available in the appendix of this thesis.

To illustrate, we could apply the MDP to an economics problem : "the management of maintenance of a machine", "maximize the income of production of a company".

In future works, we can put several agents decision-makers at the time of the execution of the action, and we can improve the resolution algorithms with respect to a large environment.

Keywords : finite and infinite horizon, method of simplex, optimal policy, dynamic programming, linear programming, Markovian Decision Processes, " Policy Iteration ", " Value Iteration ", value vector.