الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة 1
Université SAAD DAHLAB de BLIDA 1

كلية العلوم
Faculté des sciences

قسم الرياضيات
Département de mathématique

Mémoire de Master

Pour l'obtention du diplôme de master 2 en mathématiques
Spécialité modélisation stochastique et statistique

présenté par

Boumerzoug Loubna

& Serir Iméne

contrôle optimal d'un système de service stochastique à s serveurs par les processus de décision semi markovien

Devant le jury composé de :

Mr Tami Omar	**M.A.A**	**U.S.D.Blida 1**	**Président**
Mme Dahmane Zineb	**M.A.A**	**U.S.D.Blida 1**	**Promotrice**
Mme Oukid Nadia	**M.C.A**	**U.S.D.Blida 1**	**Examinatrice**

Année Universitaire 2019-2020

Remerciements

Tout d'abord, nous remercions le bon dieu, le tout puissant de nous avoir donné le courage et la volonté pour réaliser ce travail.

Nous remercions Mme Dahmane notre promotrice, pour avoir dirigé ce travail, nous tenons à lui remercier pour son aide précieuse, son objectivité, son disponibilité, son rigueur scientifique et ses conseils qui ont fait progresser ce travail.

Une pensée pleine de reconnaissance à Mdm N.Oukid et M.r Tami et à tous les enseignants du département de mathématique, pour leurs collaborations lors de notre cursus à l'université.

Nos remerciements vont aussi à l'encontre de toute personne qui a participé de prés ou de loin, directement ou indirectement à la réalisation de ce travail.

Dédicace

J'ai le grand honneur de dédier ce modeste travail :

Aux plus chères personnes de ma vie, mon mari et mes parents pour leur amour,

Son confiance et son orientation tout au long de mes études.

A toute la famille Serir, Halhal et ma petite famille Koudile et surtout à mon binôme boumerzoug loubna.

Serir imane

Dédicace

Avec un énorme plaisir, un cœur ouvert et une immense joie, que je dédie mon

Travail à mes très chers,

Respectueux et magnifiques parents qui m'ont soutenus tout Long ma vie.

A toute la famille Boumerzoug, Keddah et mes sincères remerciements à mon mari.

Et particulier remerciement à mon binôme serir imane.

Boumerzoug loubna

ملخص:

تستند هذه الأطروحة على موضوع العمليات العشوائية , و بصورة أدق عمليات نمذجة القرار لشبه ماركوف المستمدة من نظرية القرار والاحتمال .

من بين الأساليب المستخدمة في البحث الأمثل عن السياسات ، استشهدنا بخوارزميتين "تكرار القيمة" و "تكرار السياسة".

من أجل حل مشكله "السيطرة المثلى لنظام الخدمة العشوائية"، تمكنا من برمجة "تكرار القيمة" التي قمنا بتكييفها لمشكلتنا.

لسوء الحظ ، خوارزميه "تكرار السياسة" أكثر صعوبة في تطبيقه لحل هذا النوع من المشكلة.

Résumé :

Ce mémoire est inclus dans l'étude des processus stochastique, et plus précisément des processus de décision semi markovien (semi- makovian decison process SMDP), issu de la théorie de la décision et de la probabilité.

Parmi les méthode utilisées pour la recherche de politique optimale, nous avons cité deux algorithmes « value iteration » et «policy iteration » .

Pour résoudre le problème « contrôle optimal d'un système de service stochastique », nous avons réussi à programmer « value iteration » que nous avons adapté à notre problème.

Malheureusement, le deuxième algorithme « policy iteration » semble plus difficile à appliquer pour résoudre ce type de problème.

Abstract :

This memory is included in the study of stochastic processes, and more specifically semi-markovian decision-making processes (semi-makovian decison process SMDP), derived from the theory of decision and probability.

Among the methods used for optimal policy research, we cited two algorithms "value iteration" and "policy iteration"

In order to solve the problem of "optimal control of a stochastic service system", we were able to program "value iteration" which we adapted to our problem.

Unfortunately, the second "policy iteration" algorithm seems more difficult to apply to solve this type of problem.

Notation et abréviation

.MDP : Markov decision process.

.SMDP : Semi -Markov decision process.

.I : l'ensemble d'états.

. A : l'ensemble d'actions.

.A(i) : l'ensemble des actions possibles pour l'état « i ».

.R : politique stationnaire.

.x_n :l'état du système au $n^{éme}$ instant de décision.

.$p_{ij}(R_i)$: la probabilité qu'au prochain instant de décision le système soit à l'état « j », Si l'action «a» est choisie dans l'état actuel «i ».

.g_i(R) : l'espérance du coût moyen (si l'état initial est i).

.$\pi_j(R)$: Représente la distribution stationnaire d'équilibre de la chaine de Markov $\{x_n\}$.

.K (a, b) : un cout de commutation engagé lors du réglage du nombre de canaux activés de a à b.

.μ : Taux de service.

. λ: Taux d'arrivé.

<u>Si l'action «a» est choisie dans l'état actuel «i » on a les 03 notations suivantes :</u>

.P_{ij}(a):la probabilité qu'au prochain instant de décision le système soit à l'état j.

.τ_i (a): le délai prévu jusqu'au prochain instant de décision.

.C_i(a): les coûts attendus encourus jusqu'au prochain instant de décision.

Listes des figures

Table des matières

Chapitre 3 :

Application et programmation

INTRODUCTION GÉNÉRALE

Introduction générale

Le modèle des processus de décision markovien et semi markovien (MDP et SMDP) offrent un cadre général pour la résolution de problèmes de décision séquentielle dans l'incertain. Ses exploitations supposent une connaissance précise des valeurs des paramètres (probabilités et coûts).

Le modèle des processus de décision semi markovien (SMDP) est un modèle très étudié en intelligence artificielle, il offre un formalisme pour modéliser et résoudre beaucoup de problème dans l'incertain.

Dans de nombreuses situations en environnement incertain, les processus de décision markovien et semi markovien jouent un rôle important dans la résolution de ces problèmes, les transitions du système d'un état à un autre peuvent être contrôlés moyennant une séquence d'actions, le modèle correspondant est appeler processus séquentiel de décision markovien ou semi markovien qui sont à la base de la programmation dynamique stochastique.

Le problème est en général mis sous forme de « programme mathématique » ou à chaque décision possible a été associée une variable, la résolution de ce problème représente la détermination d'un ensemble de décision optimal.

Le processus de discision semi markovien (markovien decision process MDP) permet la modélisation d'un large éventail de problème d'optimisation. Il trouve des applications en gestion de stock, Maintenance, Allocation de ressources, Ordonnancement, Télécommunication et dans certain problème d'analyse de système informatique.

Ce mémoire comporte trois chapitres :

Le premier chapitre représente un petit résumé d'un mémoire de master 2 en modélisation statistique et stochastique réalisé par le binôme (Zekri YASSINE et LEKHAL SAMI) sous le thème général « les processus de décision markovien », Ce chapitre contient toutes les définitions et les éléments de base du modèle de décision Markovien.

Dans le deuxième chapitre, nous présentons les notions de base du model de décision Semi markovien, la méthode de transformation des donnés et les deux algorithmes de résolution « itération par valeur » et « itération par politique».

Finalement le troisième chapitre est consacré à la description, la modélisation et la résolution numérique du problème « contrôle optimal d'un système de service stochastique ».

CHAPITRE 01

PROCESSUS DE DÉCISION MARKOVIEN (CAS DISCRET)

Chapitre 01 : processus de décision Markovien (cas discret)

1.1 INTRODUCTION

Ce chapitre traite les systèmes dynamiques évoluant aux cours du temps, où la loi probabiliste du mouvement peut être contrôlée en prenant des décisions.

L'objectif est de trouver une règle de contrôle qui minimise le coût moyen par unité de temps.

Le modèle de décision markovien est un outil polyvalent et puissant pour l'analyse des processus de décision séquentiels probabiliste avec un horizon de planification infini.

Après quelques définitions de base, deux exemples sont présentés : un problème de maintenance et un problème de gestion d'une centrale électrique.

1.2 DEFINITIONS

1.2.1 Processus stochastique [2]

Un processus stochastique est une famille de variables aléatoire X(t) ou t est un paramètre réel prenant ses valeurs dans l'ensemble T.

En général, T est dénombrable et représente le temps, le processus est alors dit discret.

1.2.2 Chaine de Markov [3]

Le processus stochastique (x_n) $n \geq \mathbf{0}$ à valeurs dans E (ensemble fini ou dénombrable) est une chaine de Markov si :

$.P\ (x_{n+1} = j\ /x_n = i,\ x_{n-1} = i_{n-1}, \ldots\ldots\ldots\ldots,\ x_1 = i_1,\ x_0 = i_0) = p(x_{n+1} = j/x_n = i)$

$\forall \boldsymbol{n} \in \boldsymbol{N}\ ;\ \forall\ \boldsymbol{j}, \boldsymbol{i}_0,\ \ \boldsymbol{i}_1,\ \ \ldots\ldots\ldots \boldsymbol{i}_{n-1}\ \ \ , \boldsymbol{i} \in \boldsymbol{E}$.

1.2.3 Processus de décision markovien(MDP) (Markov decision process)

Un processus de décision markovien est un modèle stochastique définie par :

- un ensemble d'états I qui peut être fini, ou infini dénombrable.
- un ensemble d'actions possibles.
- Des probabilités de transition $p_{ij}(a)$.
- une fonction de récompense (coût) $C_i(a)$.

1.2.4 Système dynamique

En mathématique, en chimie ou en physique un système dynamique est la donné d'un système est d'une loi décrivant l'évolution de ce système. Ce peut être l'évolution d'une réaction chimique au cours du temps, le mouvement des planètes dans le système solaire ou encore l'évolution de la mémoire d'un ordinateur sous l'action d'un programme informatique.

1.2.5 Programmation dynamique

La programmation dynamique Est une procédure récursive permettant de calculer les valeurs optimales à partir d'une équation fonctionnelle.

En faite, la programmation dynamique c'est une approche informatique permettant d'analyser les processus de décision séquentiels avec un horizon de planification fini.

Les idées de base de la programmation dynamique sont :

- .les états.
- .le principe d'optimalité.
- .les équations fonctionnelles.

1.2.6 Les équations fonctionnelles [6]

Une équation fonctionnelle est une équation dont l'inconnu est une fonction.

En général, lorsqu'on parle d'équations, on s'intéresse à trouver les valeurs qui satisfont une égalité de deux expression algébrique.ici, on se propose ce chercher les expressions algébrique qui font que l'équation et satisfaite pour certaines valeurs données.

1.2.7 Politique [2]

Une politique ou stratégie (choix d'une action ou décision) pour chaque état, c'est un ensemble de règles « if state then action ».

1.2.8 Politique stationnaire [2]

Une politique R est dite stationnaire si elle n'est pas aléatoire, et l'action choisie à l'instant t dépend seulement de l'état du processus en cet instant.

1.2.9 Politique optimale [2]

Une politique optimale est celle qui maximise (ou minimise) les récompenses (les coûts).

1.3. MODELE

De nombreux problèmes de contrôle pratique peuvent être modélisés comme un processus de décision markovien par un choix approprié de l'espace d'états et l'ensemble d'actions.

Considérons un système dynamique avec des instants équidistants t=0,1,…….

A chaque instant de décision le système est classé dans l'un des nombres possibles d'états, et par la suite une décision doit être prise.

On dit qu'un système est de Markov à temps discret si et seulement si les propriétés d'un modèle markovien sont satisfaites.

Si à un instant de décision l'action « a » est choisie dans l'état i , alors les événements suivants se produisent indépendamment de l'historique du système :

- Un coût immédiat $C_i(a)$ est engagé.

- au prochain instant de décision, le système sera dans l'état j avec une probabilité $p_{ij}(a)$ tq $\sum_{j\in I} p_{ij}(a) = 1$, i∈ I.

On note que le paiement en une étape coûte $c_i(a)$, et les probabilités de transitions en une étape $p_{ij}(a)$ sont supposées être homogènes dans le temps.

Dans des problèmes spécifiques, les coûts « immédiats » $c_i(a)$ représenteront les coûts attendus jusqu'au prochain instant de décision, Lorsque l'action a est choisie dans l'état i.

De plus, il convient de souligner que le choix de l'espace d'états et l'ensemble d'actions dépend de la structure de coûts du problème spécifique considéré.

Le modèle de décision markovien à de nombreuses applications potentielles dans le contrôle des stocks, maintenance, fabrications, télécommunications,…..

Dans la plupart des applications il n'est pas possible de trouver une politique optimale en calculant le coût moyen de chaque politique stationnaire, par exemple si le nombre d'états est N et qu'il y'a deux actions dans chaque états, alors le nombre de politiques stationnaires possibles est 2^N et ce nombre augmente rapidement au-delà de toute limite pratique. [2]

Cependant, plusieurs algorithmes peuvent être donnés qui conduisent de manière efficace à une politique optimale, « policy iteration » et « value iteration » sont les algorithmes les plus utilisées. [2]

L'algorithme d'itération par valeur apparemment complexe est en fait très simple, il permet de calculer itérativement la valeur de chaque état. La valeur d'un état est en fait le gain obtenu par l'exécution d'une action auquel on ajoute les valeurs des différents états qu'il est possible d'atteindre en exécutant cette même action tout en prenant en compte le facteur permettant d'introduire un compromis entre action à court terme et action à long terme. Plus formellement, nous pouvons définir la valeur d'un état à l'instant t. [4]

L'algorithme d'itération par politique différent du précédent est plus complexe. Il s'agit également d'un algorithme itératif applicable à chaque pas de calcul jusqu'à obtenir la politique optimale. Contrairement à l'algorithme d'itération par valeur qui calcule une politique sans connaissance à priori, cet algorithme nécessite l'initialisation d'une première politique quelconque, ensuite améliorer par itérations successives. Chaque itération se décompose en deux phases : tout d'abord une première phase d'évaluation de la politique courante revient à la résolution d'un système de |S| équations à |S| inconnues. Enfin, la seconde phase consiste à améliorer cette politique courante en déterminant un peu comme dans l'algorithme d'itération par valeur pour chaque état une action améliorant la valeur courante. La politique optimale est obtenue lorsque deux politiques successives sont identiques et nécessairement optimales. [4]

1.3.1 Problème de maintenance [1]

Au début de chaque journée, un équipement est inspecté pour relever son état de fonctionnement réel, l'équipement sera trouvé dans l'une des conditions de travail i=1,…... N, où les conditions de travail i est meilleure que la condition de travail i+1. L'équipement détériore avec le temps.

Si la condition de travail actuelle est i et aucune réparation n'est effectuée, alors au début de jour suivant, l'équipement sera dans la condition de fonctionnement j(l'état j) avec une probabilité q_{ij}. On suppose que $q_{ij}=0$ pour $j<i$ et $\sum_{j>i} \boldsymbol{q}_{ij} = \mathbf{1}$.

La condition de travail i=N représente un dysfonctionnement nécessitant une réparation forcée prenant deux jours. Pour les états intermédiaires i avec $1<i<N$.

On a le choix entre réparer préventivement le matériel ou laisser l'équipement fonctionner pendant une journée, une réparation préventive ne prend qu'un seul jour.

Un système réparé a la condition de fonctionnement i=1, le coût d'une réparation forcée en cas de panne est $\boldsymbol{C_f}$ et le coût d'une réparation préventive en état de marche i est $\boldsymbol{C_{pi}}$.

On souhaite déterminer une règle de maintenance qui minimise le coût moyen de réparation par jour à long terme.

Ce problème peut être placé dans le cadre d'un processus de décision markovien à temps discret.

De plus, étant donné qu'une réparation forcée prend deux jours et que l'état du système doit être défini au début de chaque jour, on à besoin d'un état auxiliaire pour la situation dans laquelle une réparation forcée est en cours depuis déjà un jour, ainsi l'ensemble d'états possibles du système est choisi comme suit :

I= {1,2,…. N, N+1}

L'état i avec $1 \leq \mathbf{i} \leq \mathbf{N}$ correspond à la situation dans laquelle une inspection révèle une condition de travail i, tendis que l'état N+1 correspond à la situation dans laquelle la réparation forcée est déjà en cours depuis un jour.

Les actions :

$$A=\begin{cases} \mathbf{0} & \textbf{si aucue réparation n'est faite} \\ \mathbf{1} & \textbf{si une réparation préventive est effectuée} \\ \mathbf{2} & \textbf{si une réaration forcée est effectuée} \end{cases}$$

L'ensemble d'actions possible dans l'état i est :

A(1)= {0}, A(i)= {0,1} pour 1<i<N, A(N)=A (N+1)= {2}.

Les probabilités de transition en une étape $p_{ij}(a)$ sont donnés par :

$P_{ij}(0)= q_{ij}$ pour $1 \leq \mathbf{i} < \boldsymbol{N}$

$\mathbf{P_{i1}}(1)= 1$ pour $1 < \mathbf{i} <$ N

$\boldsymbol{P_{M,M+1}}(2) = \boldsymbol{P_{M+1,1}}(2) = 1$

$p_{ij}(a) = 0$ ailleurs (sinon)

Les coûts en une étape $c_i(a)$ sont donnés par $c_i(0) = 0$, $c_i(1)=c_{pi}$, $c_n(2)=\boldsymbol{C_f}$ et $\boldsymbol{C_{n+1}}(2)=0$ (pour plus de détails voir chapitre 6 page 235 [1]).

1.3.2 Problème de contrôle électrique [2]

Considérons une centrale électrique a deux générateurs j=1,2 pour produire de l'électricité. La quantité d'électricité nécessaire pendant la journée est variable, les 24 heures d'une journée sont subdivisées en six périodes consécutives de 4 heures chacune. La quantité d'électricité requise au cours de la période k est d_k , k=1,…..,6. D'autre part, le générateur j a une capacité de générer de c_j kWH par période pour j=1, 2, un excès d'électricité produit pendant une période ne peut pas être utilisé pour la prochaine période.

Au début de chaque période k il faut déterminer les générateurs à utiliser pour cette période (un seul générateur, lequel ? ou les deux générateurs ?)

Les coûts imposés sont les suivants :

- un coût d'exploitation r_j du générateur j pour chaque période d'utilisation.
- un coût d'installation s_j est engendré à chaque fois que le générateur j est activé (mis en marche) après avoir été inactif pendant un certain temps.

Nous souhaitons déterminer une règle de contrôle qui minimise le coût moyen à long terme par jours.

Modélisation :

Ce problème peut être mis dans le cadre d'un modèle de décision markovien à temps

discret.

Dans ce cas, l'état du système peut être décrit par le couple (k, y) telle que :

La première variable k indique la période en cours (k=1,2, ... 6) et la seconde variable y indique les générateur opérationnels (y ∈ {1,2,3})

Y=1 veut dire que le premier générateur est opérationnel, y=2 veut dire que le deuxième générateur est opérationnel et y=3 veut dire que les deux générateurs sont opérationnels

Donc l'espace des états S s'écrit :

S= {(k, y) / k=1,…,6 et y=1,2, 3}

Les actions

$$\begin{cases} 1 & si\ seulement\ le\ premiere\ générateur\ est\ utilisé \\ 2 & si\ seulement\ le\ deuxiéme\ générateur\ est\ utilisé \\ 3 & si\ les\ deux\ générateurs\ sont\ utilisé \end{cases}$$

L'ensemble des actions possibles à l'état (k, y) :

A {(k, y)}= {y} si Cy $\geq d_k$ pour y=1,2 ,3 et k=1,2,.....,6 (selon la demande de chaque période). (Pour plus de détails voir chapitre 2 page 34 [2]).

CHAPITRE 02

PROCESSUS DE DÉCISION SEMI MARKOVIEN

Chapitre 2 : Processus de décision semi markovien

2.1. INTRODUCTION

Dans le chapitre précédent, nous avons donné un bref aperçu sur les processus de décision markoviens et leurs applications, selon ce modèle, les décisions ne peuvent être prises qu'à des instants fixes t=0,1....

Cependant, il ya beaucoup de problèmes de contrôle stochastique où les temps entre les instants de décision sont aléatoire, le modèle de décision semi Markovien est un outil possible pour analyser ce type de problèmes.

Dans la section (2.2) et (2,3) nous discutons les éléments de base de ce modèle.

Dans la section (2,4) nous exposons une méthode de transformation de données par laquelle le modèle de décision semi markovien peut être converti en un modèle de décision markovien à temps discret, cette méthode nous permet d'appliquer la méthode récursive d'itération par valeur au modèle de décision semi markovien.

La section (2.5) résume divers algorithmes pour trouver une politique optimale (i.e coût moyen optimal).

Dans la section (2.6) nous discutons l'algorithme d'itération par valeur pour un modèle de décision semi-markovien ou les temps entre les instants de décision sont exponentiellement distribués, pour ce cas particulier l'effort de calcul de cet algorithme peut être considérablement réduit en introduisant des instants de décisions fictifs.

Cette astuce très simple consiste de créer des matrices de transitions conduisant à un algorithme d'itération par valeur beaucoup plus efficace.

Chapitre 2 : Processus de décision semi markovien

2.2. Le modèle de décision semi markovien

Considérons un système dynamique dont l'état est examiné à des instants aléatoires.

À ces instants, une décision doit être prise et des coûts sont induits à la suite de la décision prise.

L'ensemble d'états possibles est noté I tq $\forall i \in I$ l'ensemble A(i) des actions possible est disponible.

Supposons que l'espace d'état I et l'ensemble d'actions A(i) $i \in I$ sont finis.

Ce système dynamique contrôlé est appelé processus de décision semi-markovien lorsque les propriétés markoviennes suivantes sont satisfaites :

- Si la distribution conditionnelle de probabilité des états futurs, étant donnés les états passés et l'état présent, ne dépend, en fait que de l'état présent et non pas des états passés "absence de mémoire".

En outre, les coûts induits jusqu'au prochain instant de décision ne dépendent que de l'état actuel et l'action choisie dans cet état.

Aussi les coûts consistent généralement du coût engagé lors de la décision.

Comme exemple on considère un système d'inventaire mono produit où le processus de poisson et la position d'inventaire peut être reconstituée à tout moment par exemple les instants de la décision sont les instants de la demande et ils se produisent de manière aléatoire dans le temps [chapitre 07 page280 [1]).

2.3. Le coût moyen à long terme par unité du temps

Le coût moyen permet de déterminer une zone de bénéfice pour l'activité de l'organisation.

Le coût moyen à long terme par unité du temps est pris comme critère d'optimalité, donc le modèle de décision semi -markovien est déterminé par les caractéristiques suivantes :

- $P_{ij}(a)$: c'est la probabilité que le système soit en état j (sachant que l'action a est choisie dans l'état actuel i).
- $\tau_i(a)$: Le délai prévu jusqu'au prochain instant de décision (Si l'action a est choisie dans l'état actuel i).
- $C_i(a)$: Les coûts attendus encourus jusqu'au prochain instant de décision (Si l'action a est choisie dans l'état actuel i).

On suppose que $\tau_i(a)>0 \ \forall i \in I$ et $\forall a \in A(i)$.

une politique stationnaire R est une règle qui ajoute à chaque état i une seule action $R_i \in A(i)$ et recommande toujours de prendre cette mesure a chaque fois que le système est observé dans l'état i .

Étant donné un espace d'états fini, on peut montrer que sous chaque politique stationnaire le nombre de décisions prises dans un intervalle de temps fini est fini avec une probabilité 1.

Il s'ensuit que sous une politique stationnaire R le processus stochastique incluse $\{x_n\}$ est une chaine de Markov à temps discret avec des probabilités de transition en une étape $p_{ij}(R_i)$.

On fixe maintenant une politique stationnaire R, notons par :

$V_n(i, R)$: l'espérance du coût total sur les n premiers instants de décision, quand la politique R est utilisée à l'état initial i.

$g_i(R)$: l'espérance du coût moyen par unité de temps à long terme.

g_i(R) =$\lim_{n\to\infty} \frac{V_n(\mathrm{i,R})}{n}$ i∈ I. Cette limite existe d'après le théorème (7.1.1)

2.4. La méthode de transformation de données

Choisissez d'abord un nombre τavec 0<$\tau \leqslant \min_{i,a} \tau_i(a)$.

Considérons maintenant le modèle de décision markovien à temps discret dont les éléments de base sont donné par :

$\bar{I}$= I , Ā(i) = A(i) , i ∈ Ī.

$\bar{C}_i$(a) = $C_i(a)/\tau_i(a)$, a ∈ Ā(i) et i ∈ Ī.

$$\bar{p}_{ij}(a)=\begin{cases} \frac{\tau}{\tau_i(a)} p_{ij}(a) & j \neq i \ , a \in \bar{A}(i) \ \text{ et } \ i \in \bar{I} . \\ \frac{\tau}{\tau_i(a)} p_{ij}(a) + \left[1 - \frac{\tau}{\tau_i(a)}\right] & j = i \ , a \in \bar{A}(i) \ \text{ et } \ i \in \bar{I} . \end{cases}$$

Ce modèle de décision markovien à temps discret a la même classe de politique stationnaires comme le modèle de décision semi-markovien.

Pour chaque politique stationnaire R, g_i(R) représente le coût moyen à long terme par unité de temps dans le modèle à temps discret.

Lorsque la politique R est utilisée et l'état initial égal à i et pour chaque politique stationnaire R on a : g_i(R)= $\bar{g}_i$(R) i∈I …..(**1**)

Alors ce résultat ne nécessite aucune hypothèse sur la structure des chaines de Markov associées aux politiques stationnaires. Cependant, nous prouvons le résultat (**1**) seulement pour le cas unichain assumption (Annexe 1).

Fixons d'abord une règle stationnaire R et supposons que la chaine de Markov immergée $\{\bar{X}_n\}$ vérifié l'hypothèse de la chaine unique.

Les probabilités d'équilibre $\bar{\pi}_j$(R) de la chaine de Markov $\{\bar{X}_n\}$ satisfont les équations d'équilibres :

$\bar{\pi}_j$ (R) = $\sum_{i\in I} \bar{\pi}_i(R)\ \bar{P}_{ij}(R_i) = \sum_{i\in I} \bar{\pi}_i(R) \frac{\tau}{\tau_i(R_i)} P_{ij}(R_i) + [1 - \frac{\tau}{\tau_j(R_j)}]\ \bar{\pi}_j(R)$, j∈ I .

Ces équations sont précisément les équations d'équilibre $\pi_j(R)$ de la chaine de Markov $\{X_n\}$ dans le modèle semi markovien.

Selon le théorème (7.1.1) (annexe 01) $\bar{g}(R)=g(R)$ on peut donc conclure qu'une politique optimale de coût moyen dans le modèle semi markovien peut être obtenue en résolvant un modèle de décision markovien approprié à temps discret.

Cette conclusion est particulièrement utile par rapport à l'algorithme d'itération par valeur, en appliquant l'itération par valeur aux modèle transformé, il n'ya aucune restriction à supposer que pour chaque politique stationnaire la chaine de Markov immergé $\{x_n\}$ est apériodique.

En choisissant strictement la constante $\tau < \min_{i,a}(\tau_i(a))$ on a toujours $p_{ii}(a)>0$ ∀ i, a .

Et donc la chaine de Markov est toujours apériodique.

2.5. Algorithmes pour une politique optimale

Dans cette section nous décrivons comment les algorithmes pour le modèle de décision markovien à temps discret peuvent être étendus au modèle de décision semi-markovien.

2.5.1. Algorithme d'itération par politique [1]

L'algorithme d'itération par politique sera décrit sous l'hypothèse (unichain assumption) (annexe 1), cette hypothèse nécessite pour chaque politique stationnaire la chaine de Markov $\{x_n\}$ forme une seule chaine.

Supposons que g(R) et $v_i(R)$ i∈I soient le coût moyen et les valeurs relatives d'une politique stationnaire R .

Si une politique stationnaire R est construite de telle sorte que pour chaque état i∈I

$C_i(\bar{R}_i)$-g(R) $\tau_i(\bar{R}_i) + \sum_{j\in I} p_{ij}(\bar{R}_i)\ )v_j(R\ ) \leq v_i(R\ )$ **(2)**

Alors $g(\overline{R}) \leq$ g(R).

De plus, si le signe de l'inégalité stricte dans (**2**) pour certains états i récurrents sous R

On a $g(\overline{R}) <$g(R).

Ces affirmations peuvent être vérifiées par les mêmes arguments que ceux utilisés dans la deuxième partie de la preuve du théorème (chapitre 6.2.1 page 238) [1].

Sous l'hypothèse (unichain assumption) (annexe 1) nous pouvons maintenant formuler l'algorithme d'itération par politique :

- **.**Étape 00 : (initialisation) choisissez une politique stationnaire R.
- **.**Étape 01 : (étape de détermination de la valeur) pour la politique actuelle R, calculez le coût moyen g(R) et les valeurs relatives v_i(R) i∈I comme solution unique aux équations linéaires.

$v_i = c_i(R_i) - g\,\tau_i(R_i) + \sum_{j\in I} p_{ij}(R_i)v_j$, i∈ I.

v_s=0.

Ou s est un état choisi arbitrairement.

- **.**étape 02 : (étape d'amélioration de la politique) pour chaque état i∈ I déterminez une action a_i donnant le minimum :

$$\min_{a\in A(i)} \{ c_i(a) - g(R)\tau_i(a) + \sum_{j\in I} p_{ij}(a)v_j(R)\}$$

La nouvelle politique stationnaire $\bar{R}$ s'obtient en choisissant $\bar{R}_i=a_i$ pour tout i∈ I avec la convention que $\bar{R}_i$ est choisi égal à l'ancienne action R_i lorsque cette action minimise la quantité d'amélioration de la politique.

- **.**Étape 03 : (teste de convergence)

Si la nouvelle politique $\bar{R}$ =R alors l'algorithme est arrêté avec la politique R sinon passez à

l'étape 01 avec R remplacer par$\bar{R}$.

De la même manière que pour le modèle de décision markovien à temps discret il peut être montré que l'algorithme converge en un nombre fini d'itérations à une politique optimale de cout moyen , aussi et comme conséquence de la convergence de l'algorithme il existe des nombres g∗ et v_i* Satisfaisant l'équation d'optimalité des couts moyens

$$v_i^* = \min_{a\in A(i)} \{c_i a) - g^*\tau_i(a) + \sum_{j\in I} p_{ij}(a)v_j^*\} \quad , \quad i \in I \quad \ldots\ldots \textbf{(3)}$$

La constante g∗ est uniquement déterminée comme le cout moyen minimal par unité de temps.

De plus chaque politique stationnaire dont les actions minimisent le coté droit de (**3**)

∀ i∈I est optimale en termes de coût moyen.

2.5.2. Algorithme d'itération par valeur [1]

Pour le modèle de décision semi-markovien la formulation d'un algorithme d'itération par valeur n'est pas simple une relation de récursivité pour les couts minimaux attendus sur les n premiers instants de décisions ne prennent pas en compte les temps de transitons non identiques.

Ces couts donc ne peuvent pas être liés au coût moyen minimal par unité de temps.

Cependant par la méthode de transformation de données de la section 2.4 nous pouvons convertir le modèle de décision semi markovien à un modèle de décision markovien à temps discret tel que les deux modèles ont le même coût moyen pour chaque politique fixe.

Il n'ya aucune restriction à supposer que tous $\overline{c_i}$(a)=c_i(a)/$\tau_i a)$ sont positives, Sinon ajoutez une constante positive suffisamment grande à chaque $\overline{c_i}$(a)

Les résultats de la méthode récursive pour le modèle de décision semi-markovien sont les suivants :

- .Étape 00 : choisissez $V_{\circ}(i)$ tel que $0 \leqslant V_{\circ}(i) \leqslant \min_{a} \{c_i(a)/\tau_i(a)\}$ $\forall i \in I$

Choisissez un nombre τ avec $0 < \tau \leqslant \min_{i,a} \tau_i(a)$ soit n :=1

- Étape 01 : calculer la fonction $V_n(i)$ $i \in I$ à partir de

$$V_n(i)=\min_{a\in A(i)}\{\frac{c_i(a)}{\tau_i(a)}+\frac{\tau}{\tau_i(a)}\sum_{j\in I} p_{ij}(a)v_{n-1}(j)+(1-\frac{\tau}{\tau_i(a)})v_{n-1}(i)\} \quad \text{.....(4)}$$

Soit R(n) une politique stationnaire dont les actions minimisent le coté droit de **(4)**

- Étape 02 : calculer les limites

$$m_n=\min_{j\in I}\{V_n(j) - V_{n-1}(j)\} ,$$

$$M_n =\max_{j\in I}\{V_n(j) - V_{n-1}(j)\}.$$

L'algorithme est arrêté avec la politique R(n) lorsque $0 \leqslant (M_n - m_n) \leqslant \varepsilon\, m_n$, ou ε est le nombre de tolérance pré spécifié.

Sinon passez à l'étape 3.

- Étape 03 : n=n+1 est passez à l'étape 1

Supposons que l'hypothèse unichain faible de la section 6.5 [1] soit satisfaite pour les chaines de Markov incluses $\{x_n\}$ associées aux politiques stationnaires.

Il n y'a aucune restriction à supposer que les chaine de Markov $\{x_n\}$ dans le modèle transformé sont a périodiques , puis l'algorithme s'arrête après un grand nombre d'itérations avec une politique R(n)

Dont la fonction de coût moyen $g_i(R(_n))$ satisfait

$$0 \leq \frac{g_i(R(n)) - g^*}{g^*} \leq \varepsilon, \quad i \in I.$$

Ou g^* : désigne le coût moyen minimal par unité de temps

En ce qui concerne le choix de τ dans l'algorithme il est recommandé de prendre τcomme

suit :

$$\tau = \begin{cases} \tau = \min_{i,a} \tau_i(a) & \text{si les chaines de markov } \{x_n\} \\ \text{dans le modéle semi markovien sont aperiodiques} & \\ \tau = \frac{1}{2}\min_{i,a} \tau_i(a) & \text{sinon} \end{cases}$$

C'est un choix raisonnable

2.6. Itération par valeur et les décisions fictives :

La méthode d'itération par valeur est souvent la méthode la plus recommandée pour calculer une politique optimale de coût (presque) moyen.

À chaque itération de la méthode, les bornes inferieur et supérieur indiquent dans qu'elle mesure le coût moyen de la politique actuelle s'écarte du coût moyen minimal.

La charge de calcul pour l'algorithme d'itération par valeur dépend non seulement du nombre d'états mais aussi de la densité des probabilités de transition non nulles $p_{ij}(a)$.

Par la nature même d'algorithme d'itération par valeur il est fastidieux en calcul d'avoir beaucoup de $p_{ij}(a)$.

Dans les applications avec des temps distribués de manière exponentielle entre les instants de décision l'effort de calcul de l'algorithme d'itération par valeur peut souvent être Considérablement réduit en incluant les instants de décisions fictives. [1]

L'état du système est laissé inchangé aux instants de décisions fictives. L'inclusion d'instants de décision fictives ne change pas la nature markovienne du processus de décision car les temps entre les transitons d'état sont distribués de manière exponentielle (propriété sans mémoire).

Chapitre 2 : Processus de décision semi markovien

L'idée d'introduire les instants de décision fictive réduit non seulement l'effort de calcul, mais simplifie également la formulation de l'algorithme d'itération par valeur.

CHAPITRE 03

APPLICATION ET PROGRAMMATION DANS LES PROCESSUS SEMI MARKOVIENNE

Chapitre 3 : Application Et programmation

3.1. Optimisation des files d'attente :

Le modèle semi-markovien est un outil naturel et puissant pour l'optimisation des files d'attente, de nombreux problèmes de file d'attente en télécommunication demandent le calcul d'une règle de contrôle optimale pour une mesure de performance donnée. Si la règle de contrôle est déterminée par un ou deux paramètres, on peut d'abord utiliser l'analyse en chaine de Markov pour calculer la mesure de performance pour des valeurs données des paramètres de contrôle, puis utiliser une procédure d'optimisation standard pour rechercher les valeurs optimales des paramètres de contrôle. Cependant ce n'est pas toujours l'approche la plus efficace.

Nous donnons un exemple de système de file d'attente, pour lequel l'approche de décision semi-markovienne est non seulement plus élégante mais est également plus efficace qu'une procédure de recherche directe, Dans cette application le nombre d'états est illimitée. Cependant, en exploitant la structure du problème, nous sommes en mesure de transformer le problème en un modèle de décision de Markov avec un espace d'états fini.

3.1.1. Contrôle optimal d'un système de service stochastique

Un système de service stochastique a S canaux identiques disponibles pour fournir le service, où le nombre de canaux en fonctionnement peut être contrôlé en activant ou désactivant les canaux, par exemple les canaux de service peuvent être des caisses dans un super marché ou des machines de production dans une usine.

Les demandes arrivent selon un processus de poisson de taux λ , chaque demande de service qui arrive est autorisé à entrer dans le système, et attend en ligne jusqu'à ce qu'un canal d'exploitation soit fourni.

Le temps de service de chaque requête est distribué de manière exponentielle avec une valeur moyenne $1/\mu$.

On suppose que le taux d'arrivée moyen λ est inferieur au taux de service maximal sμ.

Un canal activé ne peut gérer qu'une requête à la fois.

À tout instant les canaux peuvent être activés ou désactivés en fonction du nombre de demandes de service dans le système.

Un coût de commutation non négatif K (a, b) est induit lors du réglage du nombre de canaux activés de a à b.

Pour chaque canal activé, il y'a un coût d'exploitation de taux r>0 par unité de temps, aussi pour chaque demande un coût de possession de h>0 est induit pour chaque unité de temps, pendant laquelle la demande est dans le système jusqu'à ce que son service soit terminé [1].

L'objectif est de trouver une politique pour contrôler le nombre de canaux activés, de sorte que le coût moyen à long terme par unité de temps soit minimal.

3.2. Modélisation [1] :

Comme le processus de poisson et la distribution exponentielle sont sans mémoire, l'état du système à tout moment est décrit par la paire (i, t) ou :

- I : le nombre de demandes de service présentes (nombre de clients).
- T : le nombre des canaux activés.

Les instants de décisions sont les instants d'arrivée d'une nouvelle demande ou les instants de fin de service.

Dans cet exemple le nombre d'états possible est illimité. Car la variable d'état i prend les valeurs 0,1….

Une approche naturelle aboutirait à une formulation de modèle de décision semi markovien, dont la variable d'état i est délimitée par un nombre U suffisamment grand, de sorte que la probabilité qu'il y ait plus que U demandes dans le système est négligeable. Lorsque le taux d'arrivée λ est proche du débit de service maximal sμ alors cette approche conduirait à un très grand espace d'états.

Une formulation de modèle de décision markovien plus efficace est obtenue en restreignant la classe des règles de contrôle plutôt qu'en tronquant l'espace d'états.

Il est intuitivement évident que sous chaque règle de contrôle raisonnable tous les canaux seront activés lorsque le nombre de demandes dans le système est suffisamment grand.

En d'autres termes, en choisissant un entier suffisamment grand M avec M ≥ s, tel que pour chaque i≥ M la seule action réalisable consiste à activer tout les canaux.

Cependant, cela implique que nous pouvons restreindre le contrôle du système aux instants d'arrivée et d'achèvement du service, aux quels il ne reste pas plus que M demandes de service dans le système.

Ce faisant, on obtient une formulation de processus de décision semi –markovien avec l'espace d'états :

i= {(i, t) / 0≤ $i \leq M$, 0≤ $t \leq s$}

L'ensemble d'actions :

A (i, t) = $\begin{cases} \{a \backslash a = 0, \ldots\ldots, s\} & 0 \leq i \leq M-1 \ , \quad 0 \leq t \leq s \\ \{s\} & \mathrm{I} = \mathrm{M} \ , \quad 0 \leq t \leq s \end{cases}$

Ici l'action a dans l'état (i, t) signifie que le nombre de canaux est ajusté de t à a. Cette formulation de processus de décision semi – markovien implique la stipulation suivante :

Si l'action a=s est prise à l'état (M, T) alors le prochain instant de décision est défini comme le premier instant d'achèvement de service au quel M au M-1 demandes sont en attente.

En outre, si l'action a=s est effectuée dans l'état (M, t) les coûts induits en une étape jusqu'au prochain instant de décision sont définis comme la somme des coûts de commutation k(t, s) et les coûts d'attente et d'exploitation réalisés pendant la période allant jusqu'au prochain instant de décision.

On désigne par la variable aléatoire T_M(s) le temps jusqu'au prochain instant de décision ou l'action a=s est prise à l'état (M, T). La variable aléatoire T_M(s) est la somme de deux composantes, la première est le temps jusqu'à ce que le prochain achèvement de service se réalise, ou bien le temps jusqu'à la prochaine arrivée. La deuxième composante est zéro si un achèvement de service se produit en premier, sinon il est distribué comme le temps

nécessaire pour réduire le nombre de demandes de service présentes de M+1 à M.

La formulation en modèle de décision semi-markovien avec un espace d'états inclus (immergé) est réalisable uniquement s'il est possible de calculer l'espérance du temps moyen de transition en une étape $\tau_{(M,t)}(s)$ ainsi que l'espérance du coût en une étape $C_{(M,t)}(s)$.

Le calcul de ces quantités est facile, puisque les services sont complétés selon un processus de poisson au taux sμ tant que tous les canaux sont occupés. En d'autres termes chaque fois que M ou plus de demandes sont dans le système, nous pouvons imaginer de manière équivalente qu'un seul « super Channel » traite les demandes une par une à un taux exponentiel de sμ.

En prenant n=1 et en remplaçant la moyenne $1/\mu$ par $1/s\mu$ dans les formules {(2.6.2) et (2.6.3) annexe 1} nous constatons que le temps attendu nécessaire pour réduire le nombre de demandes présentes de M+1 à M, étant donné que tous les canaux sont activés est

$$\frac{1/s\mu}{1-\lambda/s\mu} = \frac{1}{s\mu-\lambda}$$

Et les coûts de détention et d'exploitation prévu pendant le temps nécessaire pour réduire le nombre de demandes présentes de M+1 à M, étant donné que tous les canaux sont activés est :

$$\frac{hM}{s\mu-\lambda} + \frac{hs\mu}{s\mu-\lambda}\left\{\frac{1}{s\mu} + \frac{\lambda}{s\mu(s\mu-\lambda)}\right\} + \frac{rs}{s\mu-\lambda} = \frac{h(M+1)+rs}{s\mu-\lambda} + \frac{h\lambda}{(s\mu-\lambda)^2}$$

Ici, le terme hM/(sμ-λ) représente les coûts de possession attendus pour les M demandes de service qui sont présentent en permanence pendant les temps nécessaire pour réduire le nombre dans le système de M+1 à M.

Si tous les canaux S sont occupés, alors le délai jusqu'au prochain événement (achèvement du service ou nouvelle arrivée) est distribué de manière exponentielle avec la moyenne $1/\lambda$+sμ et l'événement suivant est généré par une arrivée avec une probabilité $\lambda/\lambda + s\mu$.

On trouve :

$$\tau_{(M,t)}(s) \quad = \frac{1}{\lambda+s\mu} + \lambda/\lambda+s\mu\left(\frac{1}{s\mu-\lambda}\right) = \frac{s\mu}{(\lambda+s\mu)(s\mu-\lambda)}$$

Et

$$c_{(M,t)}(s) = k(t, s) + \frac{hM+rs}{\lambda+s\mu} + \frac{\lambda}{\lambda+s\mu}\left\{\frac{h(M+1)+rs}{s\mu-\lambda} + \frac{h\lambda}{(s\mu-\lambda)^2}\right\}$$

Nous avons également les probabilités :

$P_{(M,t)(M-1,s)}(S) = \frac{s\mu}{(\lambda+s\mu)}$ et $P_{(M,t)(M,s)}(S) = \frac{\lambda}{\lambda+s\mu}$

Pour les autres états, les éléments de base du modèle de décision semi –markovien sont facilement spécifies, on a

$\tau_{(i,t)}(a) = \frac{1}{\lambda+\min(i;a)\mu}$ $0 \leqslant i \leqslant M-1$, $0 \leqslant a \leqslant s$

Et

$C_{(i,t)}(a) = k(t, a) + \frac{hi+ra}{\lambda+\min(i,a)\mu}$ $0 \leqslant i \leqslant M-1$, $0 \leqslant a \leqslant s$

3.2.1 Résolution avec l'algorithme d'itération par valeur [1]

Nous allons utiliser l'algorithme d'itération par valeur (chapitre 2) pour chercher une politique optimale

- <u>Etape00</u> : choisissez $V_0(i)$ tel que $0 \leqslant V_0(i) \leqslant \min_a \{C_{i,t}(a)/\tau_{i,t}(a)\}$ $\forall i \in I$, $0 \leq t \leq s$.

  Dans la transformation de données on prend $\tau = 1/\lambda+s\mu$.

Soit n =1.

- <u>Étape 01 :</u> calculer la fonction $V_n(i,t)$ $i \in I$, $0 \leq t \leq s$ à partir de :

$$V_n(i,t) = \min_{0 \leq a \leq s}\left[\{\lambda + \min(i,a)\mu\}k(t,a) + hi + ra + \frac{\lambda}{\lambda+s\mu} v_{n-1}((i+1,a)) + \frac{\min(i;a)\mu}{\lambda+s\mu} v_{n-1}((i-1,a)) + \{1 - \frac{\lambda+\min(i,a)\mu}{\lambda+s\mu}\} v_{n-1}((i,t))\right],$$

Pour les états (i, t) avec $0 \leqslant i \leqslant m-1$, $0 \leqslant t \leqslant s$

Et

$V_n((M,t))=1/s\mu\ (\lambda+s\mu)\ (s\mu-\lambda)\ k(t,s)+\frac{h\lambda}{s\mu-\lambda}+hM+rs+\frac{s\mu-\lambda}{\lambda+s\mu}v_{n\_1}((M-1,s))+\{1-\frac{\lambda(s\mu-\lambda)}{s\mu(\lambda+s\mu)}\}v_{n\_1}((M,s))+\{1-\frac{s\mu-\lambda}{s\mu}\}v_{n\_1}((M,t))$.

Pour les états (M, t) avec , $0\leqslant t\leqslant s$

Avec la convention $v_{n\_1}((-1,t))=0$.

- Étape 02 : calculer les limites :

$m_n=\min_{i\in I}\{V_n(i,t)-V_{n\_1}(i,t)\}$, $M_n=\max_{i\in I}\{V_n(i,t)-V_{n\_1}(i,t)\}$.

L'algorithme est arrêté avec la politique R(n) (une politique stationnaire) lorsque $0\leqslant(M_n-m_n)\leqslant\varepsilon\ m_n$, ou ε est un numéro de précision pré spécifié.

Sinon passez à l'étape 3.

- Étape 03 : n=n+1 est passez à l'étape 1.

a. Résultat numérique

On considère la fonction de coût de commutation k (a, b)= k|a-b| et les données numériques suivantes :

S=10, $\mu=1$, r=30, et h=10.

Le taux d'arrivée λ prend les valeurs 7 et 8,

Le coût de commutation k prend les deux valeurs 10 et 25, dans chaque exemple nous prenons la borne M=20.

Pour les états (i, t) avec $i\geqslant M$ tous les canaux s sont toujours activés.

L'algorithme d'itération par valeur est lancé avec $V_\circ(i,t)=0$ pour tous les états (i, t).

On choisit le nombre de tolérance $\varepsilon=10^{-3}$ pour le critère d'arrêt.

Nous avons $(M+1)\times(s+1)$ équations, une fois les valeurs calculées, elles seront

stockées dans une matrice de dimension (M+1)(s+1).

Donc :

$$V_n\ (i, t) = \begin{pmatrix} v_n(0,0) & v_n(0,1) & \dots & v_n(0,s) \\ v_n(1,0) & v_n(1,1) & \cdots & v_n(1,s) \\ \vdots & \vdots & \ddots & \vdots \\ v_n(M,0) & v_n(M,1) & \cdots & v_n(M,s) \end{pmatrix}$$

- Etape 00 : On choisit v_0 (i, t) comme suit pour tous les cas :

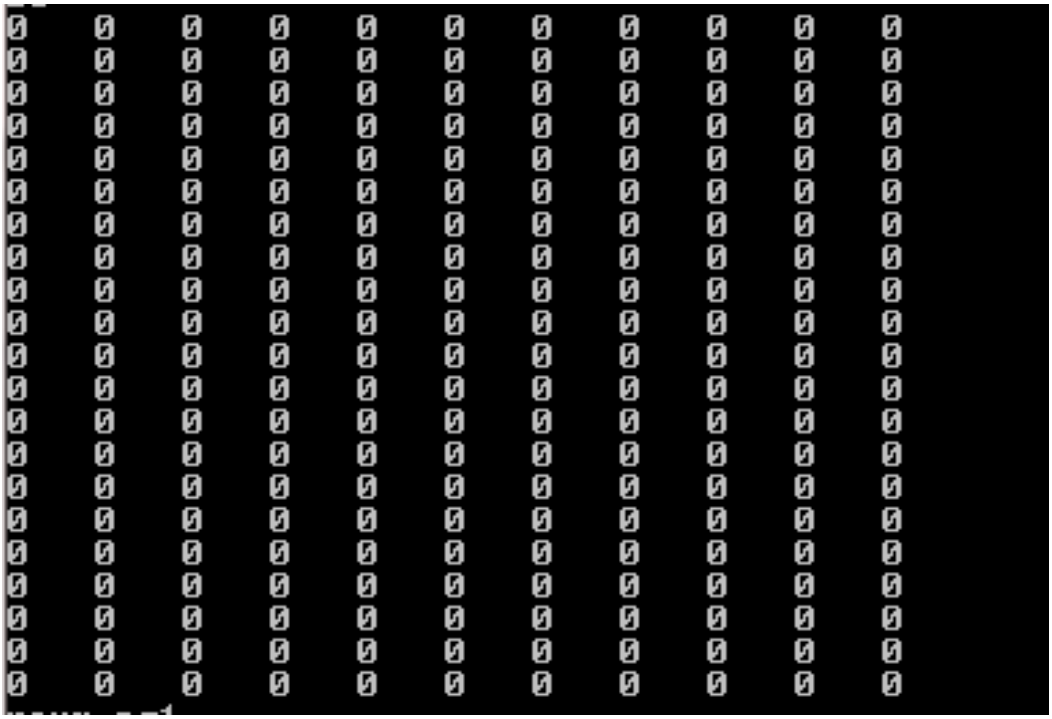

Figure 0 : l'affichage de v (i, t) pour n=0

1er cas $\lambda = 7$; $k = 10$

Soit n=1 :

- Etape 01 : v_1 (i, t)=

```
pour n=1
0   30   60   90   120   150   180   210   240   270   300
10   40   70   100   130   160   190   220   250   280   310
20   50   80   110   140   170   200   230   260   290   320
30   60   90   120   150   180   210   240   270   300   330
40   70   100   130   160   190   220   250   280   310   340
50   80   110   140   170   200   230   260   290   320   350
60   90   120   150   180   210   240   270   300   330   360
70   100   130   160   190   220   250   280   310   340   370
80   110   140   170   200   230   260   290   320   350   380
90   120   150   180   210   240   270   300   330   360   390
100   130   160   190   220   250   280   310   340   370   400
110   140   170   200   230   260   290   320   350   380   410
120   150   180   210   240   270   300   330   360   390   420
130   160   190   220   250   280   310   340   370   400   430
140   170   200   230   260   290   320   350   380   410   440
150   180   210   240   270   300   330   360   390   420   450
160   190   220   250   280   310   340   370   400   430   460
170   200   230   260   290   320   350   380   410   440   470
180   210   240   270   300   330   360   390   420   450   480
190   220   250   280   310   340   370   400   430   460   490
900   864   828   792   756   720   684   648   612   576   540

( mn , Mn )=(0 , 900)
```

Figure1 : l'affichage de v (i, t) pour n=1

- Etape 02 : Le critère d'arrêt $0 \leq M_n - m_n \leq \varepsilon * m_n$ n'est pas vérifiée car (900≠0) donc on passe a l'étape suivante.

- Etape 03 : n=2 (on fait le même travail pour les autres itérations jusqu'a obtenir une politique optimale R(n)).

Pour n=2 :

```
pour n=2
4.11765  64.1176  124.118  184.118  244.118  304.118  364.118  424.118  484.118
 544.118  604.118
24.1176  83.5294  143.529  203.529  263.529  323.529  383.529  443.529  503.529
 563.529  623.529
44.1176  103.529  162.941  222.941  282.941  342.941  402.941  462.941  522.941
 582.941  642.941
64.1176  123.529  182.941  242.353  302.353  362.353  422.353  482.353  542.353
 602.353  662.353
84.1176  143.529  202.941  262.353  321.765  381.765  441.765  501.765  561.765
 621.765  681.765
104.118  163.529  222.941  282.353  341.765  401.176  461.176  521.176  581.176
 641.176  701.176
124.118  183.529  242.941  302.353  361.765  421.176  480.588  540.588  600.588
 660.588  720.588
144.118  203.529  262.941  322.353  381.765  441.176  500.588  560  620  680  74
0
164.118  223.529  282.941  342.353  401.765  461.176  520.588  580  639.412  699
.412  759.412
184.118  243.529  302.941  362.353  421.765  481.176  540.588  600  659.412  718
.824  778.824
204.118  263.529  322.941  382.353  441.765  501.176  560.588  620  679.412  738
.824  798.235
224.118  283.529  342.941  402.353  461.765  521.176  580.588  640  699.412  758
.824  818.235
244.118  303.529  362.941  422.353  481.765  541.176  600.588  660  719.412  778
.824  838.235
264.118  323.529  382.941  442.353  501.765  561.176  620.588  680  739.412  798
.824  858.235
284.118  343.529  402.941  462.353  521.765  581.176  640.588  700  759.412  818
.824  878.235
304.118  363.529  422.941  482.353  541.765  601.176  660.588  720  779.412  838
.824  898.235
324.118  383.529  442.941  502.353  561.765  621.176  680.588  740  799.412  858
.824  918.235
344.118  403.529  462.941  522.353  581.765  641.176  700.588  760  819.412  878
.824  938.235
364.118  423.529  482.941  542.353  601.765  661.176  720.588  780  839.412  898
.824  958.235
727.255  753.314  779.373  805.431  831.49  857.549  883.608  909.667  935.725
961.784  987.843
1907.78  1821.08  1734.38  1647.68  1560.98  1474.28  1387.58  1300.88  1214.18
 1127.48  1040.78

( mn , Mn )=(4.11765 , 874.451)
```

Figure 02 : l'affichage de v (i, t) pour n=2

Le critère d'arrêt $0 \leq M_n - m_n \leq \varepsilon * m_n$ n'est pas vérifiée car $870.33 > 0.00411$.

<u>Pour n=174</u>

```
pour n=174
46712.1  46615  46601.1  46686.7  46856.7  47026.7  47196.7  47366.7  47536.7  4
7706.7  47876.7
47487.7  47317.7  47230.9  47243.7  47385.6  47555.6  47725.6  47895.6  48065.6
 48235.6  48405.6
48266.5  48096.5  47926.5  47855.9  47921  48091  48261  48431  48601  48771  48
941
49046.4  48876.4  48706.4  48539.3  48509.5  48621.4  48791.4  48961.4  49131.4
 49301.4  49471.4
49826.3  49656.3  49486.3  49316.3  49173  49187.1  49339.2  49509.2  49679.2  4
9849.2  50019.2
50619.1  50449.1  50279.1  50109.1  49939.1  49824.5  49893.6  50063.6  50233.6
 50403.6  50573.6
51419.6  51249.6  51079.6  50909.6  50739.6  50569.6  50506.6  50617.9  50787.9
 50957.9  51127.9
52217.5  52047.5  51877.5  51707.5  51537.5  51367.5  51224.7  51212.9  51351.1
 51521.1  51691.1
53028.7  52858.7  52688.7  52518.7  52348.7  52178.7  52008.7  51903.5  51937.1
 52097.8  52267.8
53844.9  53674.9  53504.9  53334.9  53164.9  52994.9  52824.9  52665.4  52605.2
 52682.5  52852.5
54664.3  54494.3  54324.3  54154.3  53984.3  53814.3  53644.3  53474.3  53343  5
3335.5  53441.4
55496.1  55326.1  55156.1  54986.1  54816.1  54646.1  54476.1  54306.1  54136.1
 54052.2  54086.9
56328.1  56158.1  55988.1  55818.1  55648.1  55478.1  55308.1  55138.1  54968.1
 54826.4  54789
57180.3  57010.3  56840.3  56670.3  56500.3  56330.3  56160.3  55990.3  55820.3
 55650.3  55547.6
58044  57874  57704  57534  57364  57194  57024  56854  56684  56514  56362.8
58934.4  58764.4  58594.4  58424.4  58254.4  58084.4  57914.4  57744.4  57574.4
 57404.4  57234.4
59862.5  59692.5  59522.5  59352.5  59182.5  59012.5  58842.5  58672.5  58502.5
 58332.5  58162.5
60847.2  60677.2  60507.2  60337.2  60167.2  59997.2  59827.2  59657.2  59487.2
 59317.2  59147.2
61888.3  61718.3  61548.3  61378.3  61208.3  61038.3  60868.3  60698.3  60528.3
 60358.3  60188.3
62986  62816  62646  62476  62306  62136  61966  61796  61626  61456  61286
64140.2  63970.2  63800.2  63630.2  63460.2  63290.2  63120.2  62950.2  62780.2
 62610.2  62440.2

( mn , Mn )=(319.34 , 319.652)

dans cet exemple l'algorithme convergeapres174iteration avec le cout optimale il
 est entre  les deux borne mn et Mn319.34   319.652
```

Figure 03 : l'affichage de v (i, t) pour n=174 et le coût moyen optimal

Le critère d'arrêt est vérifiée après 174 itérations $0 \leq M_n\text{-}m_n \leq \varepsilon * m_n$ $(\varepsilon = 0.001)$ $(0.312<0.319)$

Le coût moyen minimal est compris entre $(M_n.m_n) = (313{,}34 ; 319{,}652)$.

L'algorithme d'itération par valeur est arrêté avec la politique stationnaire R(174)

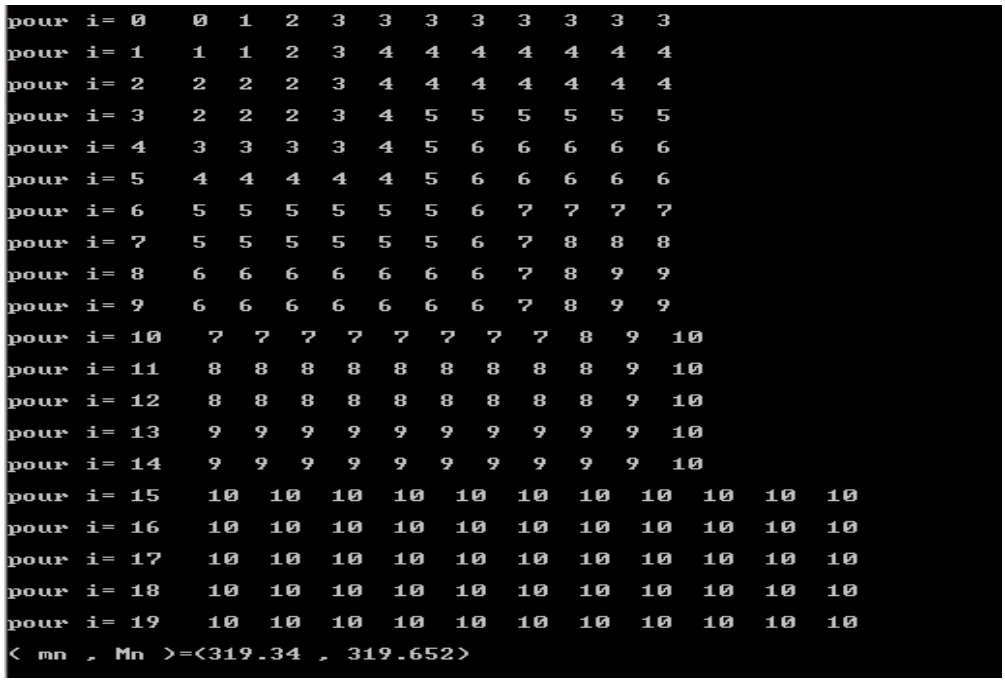

Figure 04 : affichage de la politique optimale

Cette politique optimale représente la décision qu'on doit prendre pour avoir un coût optimal, c'est à dire elle indique pour tous les états (i,t) $0 \leq i \leq M$ le bon choix de nombre de carnaux activées.

t / i	2	5	7	9	10
2	2	4	4	4	4
8	6	6	7	9	9
12	8	8	8	9	10
18	10	10	10	10	10

Tableau 05 : représente le nombre de carnaux activés pour un coût optimal

D'après le tableau :

Pour l'état (2,2) : la décision qui donne le coût optimal est a=2 (càd le nombre de canaux allumées reste telle qu'il est).

Pour l'état (8,10) : La décision optimale est a=9 (on ferme un seul carneau)

Pour l'état(18,7) : On voit que le coût est optimal pour a=10 (on ouvre 03 carnaux).

2 éme cas $\lambda = 8$; $k = 10$

Pour n=1 : on aura v_1 (i, t)=

```
pour n=1
0  30  60  90  120  150  180  210  240  270  300
10  40  70  100  130  160  190  220  250  280  310
20  50  80  110  140  170  200  230  260  290  320
30  60  90  120  150  180  210  240  270  300  330
40  70  100  130  160  190  220  250  280  310  340
50  80  110  140  170  200  230  260  290  320  350
60  90  120  150  180  210  240  270  300  330  360
70  100  130  160  190  220  250  280  310  340  370
80  110  140  170  200  230  260  290  320  350  380
90  120  150  180  210  240  270  300  330  360  390
100  130  160  190  220  250  280  310  340  370  400
110  140  170  200  230  260  290  320  350  380  410
120  150  180  210  240  270  300  330  360  390  420
130  160  190  220  250  280  310  340  370  400  430
140  170  200  230  260  290  320  350  380  410  440
150  180  210  240  270  300  330  360  390  420  450
160  190  220  250  280  310  340  370  400  430  460
170  200  230  260  290  320  350  380  410  440  470
180  210  240  270  300  330  360  390  420  450  480
190  220  250  280  310  340  370  400  430  460  490
900  864  828  792  756  720  684  648  612  576  540

( mn , Mn )=(0 , 900)
```

Figure 6 : l'affichage de v (i, t) pour n=1.

Le critère d'arrêt $0 \leq M_n - m_n \leq \varepsilon * m_n$ n'est pas vérifiée car ($900 \neq 0$) donc on passe a l'étape suivante

Pour n=2 :

```
pour n=2
4.44444  64.4444  124.444  184.444  244.444  304.444  364.444  424.444  484.444
 544.444  604.444
24.4444  83.8889  143.889  203.889  263.889  323.889  383.889  443.889  503.889
 563.889  623.889
44.4444  103.889  163.333  223.333  283.333  343.333  403.333  463.333  523.333
 583.333  643.333
64.4444  123.889  183.333  242.778  302.778  362.778  422.778  482.778  542.778
 602.778  662.778
84.4444  143.889  203.333  262.778  322.222  382.222  442.222  502.222  562.222
 622.222  682.222
104.444  163.889  223.333  282.778  342.222  401.667  461.667  521.667  581.667
 641.667  701.667
124.444  183.889  243.333  302.778  362.222  421.667  481.111  541.111  601.111
 661.111  721.111
144.444  203.889  263.333  322.778  382.222  441.667  501.111  560.556  620.556
 680.556  740.556
164.444  223.889  283.333  342.778  402.222  461.667  521.111  580.556  640  700
  760
184.444  243.889  303.333  362.778  422.222  481.667  541.111  600.556  660  719
.444  779.444
204.444  263.889  323.333  382.778  442.222  501.667  561.111  620.556  680  739
.444  798.889
224.444  283.889  343.333  402.778  462.222  521.667  581.111  640.556  700  759
.444  818.889
244.444  303.889  363.333  422.778  482.222  541.667  601.111  660.556  720  779
.444  838.889
264.444  323.889  383.333  442.778  502.222  561.667  621.111  680.556  740  799
.444  858.889
284.444  343.889  403.333  462.778  522.222  581.667  641.111  700.556  760  819
.444  878.889
304.444  363.889  423.333  482.778  542.222  601.667  661.111  720.556  780  839
.444  898.889
324.444  383.889  443.333  502.778  562.222  621.667  681.111  740.556  800  859
.444  918.889
344.444  403.889  463.333  522.778  582.222  641.667  701.111  760.556  820  879
.444  938.889
364.444  423.889  483.333  542.778  602.222  661.667  721.111  780.556  840  899
.444  958.889
695.556  725.667  755.778  785.889  816  846.111  876.222  906.333  936.444  966
.556  996.667
1722.44  1657.64  1592.84  1528.04  1463.24  1398.44  1333.64  1268.84  1204.04
 1139.24  1074.44

( mn , Mn )=(4.44444 , 822.444)
```

Figure 07 : l'affichage de v (i, t) pour n=2

Le critère d'arrêt $0 \leq M_n - m_n \leq \varepsilon * m_n$ n'est pas vérifiée car $817.9995 > 0.0044$

Pour n=312 :

```
pour n=312
103938  103826  103789  103839  104002  104182  104362  104542  104722  104902
105082
104764  104584  104480  104462  104558  104738  104918  105098  105278  105458
105638
105595  105415  105235  105141  105160  105304  105484  105664  105844  106024
106204
106429  106249  106069  105889  105822  105889  106069  106249  106429  106609
106789
107266  107086  106906  106726  106559  106530  106646  106826  107006  107186
107366
108113  107933  107753  107573  107393  107248  107265  107415  107595  107775
107955
108973  108793  108613  108433  108253  108073  107960  108025  108204  108384
108564
109843  109663  109483  109303  109123  108943  108768  108700  108811  108991
109171
110717  110537  110357  110177  109997  109817  109637  109487  109471  109606
109786
111610  111430  111250  111070  110890  110710  110530  110350  110236  110260
110414
112515  112335  112155  111975  111795  111615  111435  111255  111086  111011
111069
113436  113256  113076  112896  112716  112536  112356  112176  111996  111850
111814
114385  114205  114025  113845  113665  113485  113305  113125  112945  112765
112649
115362  115182  115002  114822  114642  114462  114282  114102  113922  113742
113574
116388  116208  116028  115848  115668  115488  115308  115128  114948  114768
114588
117492  117312  117132  116952  116772  116592  116412  116232  116052  115872
115692
118686  118506  118326  118146  117966  117786  117606  117426  117246  117066
116886
119970  119790  119610  119430  119250  119070  118890  118710  118530  118350
118170
121343  121163  120983  120803  120623  120443  120263  120083  119903  119723
119543
122807  122627  122447  122267  122087  121907  121727  121547  121367  121187
121007
124360  124180  124000  123820  123640  123460  123280  123100  122920  122740
122560

( mn , Mn )=(367.062 , 367.422)

dans cet exemple l'algorithme convergeapres312iteration avec le cout optimale il
 est entre  les deux borne mn et Mn367.062   367.422
```

Figure 08 : l'affichage de v (i, t) pour n=312 et le coût moyen optimal

Le critère d'arrêt est vérifiée après 312 itérations $0 \leq M_n - m_n \leq \varepsilon * m_n$ $(\varepsilon = 0.001)$ $(0.36<0.367)$.

Le coût moyen minimal est compris entre $(M_n ; m_n) = (367{,}062 ; 367{,}422)$.

La politique optimale : voir (Annexe 2).

3 éme cas $\lambda = 7$; $k = 25$

Pour n=1 : on aura v_1 (i, t)=

```
pour n=1
0  30  60  90  120  150  180  210  240  270  300
10  40  70  100  130  160  190  220  250  280  310
20  50  80  110  140  170  200  230  260  290  320
30  60  90  120  150  180  210  240  270  300  330
40  70  100  130  160  190  220  250  280  310  340
50  80  110  140  170  200  230  260  290  320  350
60  90  120  150  180  210  240  270  300  330  360
70  100  130  160  190  220  250  280  310  340  370
80  110  140  170  200  230  260  290  320  350  380
90  120  150  180  210  240  270  300  330  360  390
100  130  160  190  220  250  280  310  340  370  400
110  140  170  200  230  260  290  320  350  380  410
120  150  180  210  240  270  300  330  360  390  420
130  160  190  220  250  280  310  340  370  400  430
140  170  200  230  260  290  320  350  380  410  440
150  180  210  240  270  300  330  360  390  420  450
160  190  220  250  280  310  340  370  400  430  460
170  200  230  260  290  320  350  380  410  440  470
180  210  240  270  300  330  360  390  420  450  480
190  220  250  280  310  340  370  400  430  460  490
1798.33  1670.83  1543.33  1415.83  1288.33  1160.83  1033.33  905.833  778.333
 650.833  523.333

( mn , Mn )=(0 , 1798.33)
```

Figure 09: l'affichage de v (i, t) pour n=1

Le critère d'arrêt $0 \leq M_n - m_n \leq \varepsilon * m_n$ n'est pas vérifiée car (1798,33≠0) donc on passe a l'étape suivante

Pour n=2 :

```
pour n=2
4.11765  64.1176  124.118  184.118  244.118  304.118  364.118  424.118  484.118
 544.118  604.118
24.1176  83.5294  143.529  203.529  263.529  323.529  383.529  443.529  503.529
 563.529  623.529
44.1176  103.529  162.941  222.941  282.941  342.941  402.941  462.941  522.941
 582.941  642.941
64.1176  123.529  182.941  242.353  302.353  362.353  422.353  482.353  542.353
 602.353  662.353
84.1176  143.529  202.941  262.353  321.765  381.765  441.765  501.765  561.765
 621.765  681.765
104.118  163.529  222.941  282.353  341.765  401.176  461.176  521.176  581.176
 641.176  701.176
124.118  183.529  242.941  302.353  361.765  421.176  480.588  540.588  600.588
 660.588  720.588
144.118  203.529  262.941  322.353  381.765  441.176  500.588  560  620  680  74
0
164.118  223.529  282.941  342.353  401.765  461.176  520.588  580  639.412  699
.412  759.412
184.118  243.529  302.941  362.353  421.765  481.176  540.588  600  659.412  718
.824  778.824
204.118  263.529  322.941  382.353  441.765  501.176  560.588  620  679.412  738
.824  798.235
224.118  283.529  342.941  402.353  461.765  521.176  580.588  640  699.412  758
.824  818.235
244.118  303.529  362.941  422.353  481.765  541.176  600.588  660  719.412  778
.824  838.235
264.118  323.529  382.941  442.353  501.765  561.176  620.588  680  739.412  798
.824  858.235
284.118  343.529  402.941  462.353  521.765  581.176  640.588  700  759.412  818
.824  878.235
304.118  363.529  422.941  482.353  541.765  601.176  660.588  720  779.412  838
.824  898.235
324.118  383.529  442.941  502.353  561.765  621.176  680.588  740  799.412  858
.824  918.235
344.118  403.529  462.941  522.353  581.765  641.176  700.588  760  819.412  878
.824  938.235
364.118  423.529  482.941  542.353  601.765  661.176  720.588  780  839.412  898
.824  958.235
1042.25  1036.81  1031.37  1025.93  1020.49  1015.05  1009.61  1004.17  998.725
 993.284  987.843
3208.28  2991.53  2774.78  2558.03  2341.28  2124.53  1907.78  1691.03  1474.28
 1257.53  1040.78

( mn , Mn )=(4.11765 , 1409.95)
```

Figure 10 : l'affichage de v (i, t) pour n=2

Le critère d'arrêt $0 \leq M_n - m_n \leq \varepsilon * m_n$ n'est pas vérifiée car 1408,83>1.4099

Pour n=226

```
pour n=226
68051.9  67699.8  67430.9  67258.7  67221.2  67355.5  67748.9  68173.9  68598.9
 69023.9  69448.9
68857.2  68432.2  68090.5  67845.4  67735.1  67796.5  68117.1  68542.1  68967.1
 69392.1  69817.1
69670  69245  68820  68491.6  68298  68276.2  68513.6  68938.6  69363.6  69788.6
  70213.6
70489.3  70064.3  69639.3  69214.3  68924.1  68805.6  68946.5  69371.5  69796.5
 70221.5  70646.5
71312.9  70887.9  70462.9  70037.9  69633.4  69400.6  69427.3  69793.2  70218.2
 70643.2  71068.2
72155.3  71730.3  71305.3  70880.3  70455.3  70084.4  69972.9  70232.3  70657.3
 71082.3  71507.3
73017.3  72592.3  72167.3  71742.3  71317.3  70892.3  70609.4  70719.8  71115.9
 71540.9  71965.9
73889.5  73464.5  73039.5  72614.5  72189.5  71764.5  71377.2  71287.2  71585.6
 72010.6  72435.6
74783.3  74358.3  73933.3  73508.3  73083.3  72658.3  72233.3  71979.7  72107.8
 72496.6  72921.6
75690.8  75265.8  74840.8  74415.8  73990.8  73565.8  73140.8  72773.1  72732.7
 73007.3  73432.3
76618.1  76193.1  75768.1  75343.1  74918.1  74493.1  74068.1  73643.1  73450.6
 73594.8  73952.3
77540.3  77115.3  76690.3  76265.3  75840.3  75415.3  74990.3  74565.3  74250.5
 74257  74529.1
78490.3  78065.3  77640.3  77215.3  76790.3  76365.3  75940.3  75515.3  75119.7
 74990.6  75162.6
79444  79019  78594  78169  77744  77319  76894  76469  76044  75792  75852.9
80406.7  79981.7  79556.7  79131.7  78706.7  78281.7  77856.7  77431.7  77006.7
 76656  76600
81389  80964  80539  80114  79689  79264  78839  78414  77989  77576.3  77404
82369.4  81944.4  81519.4  81094.4  80669.4  80244.4  79819.4  79394.4  78969.4
 78544.4  78264.9
83374.8  82949.8  82524.8  82099.8  81674.8  81249.8  80824.8  80399.8  79974.8
 79549.8  79182.6
84403.7  83978.7  83553.7  83128.7  82703.7  82278.7  81853.7  81428.7  81003.7
 80578.7  80157.1
85438.5  85013.5  84588.5  84163.5  83738.5  83313.5  82888.5  82463.5  82038.5
 81613.5  81188.5
86526.7  86101.7  85676.7  85251.7  84826.7  84401.7  83976.7  83551.7  83126.7
 82701.7  82276.7

( mn , Mn )=(331.297 , 331.617)

dans cet exemple l'algorithme convergeapres226iteration avec le cout optimale il
 est entre  les deux borne mn et Mn331.297   331.617
```

Figure 11 : l'affichage de v (i, t) pour n=226 et le coût moyen optimal

Le critère d'arrêt est vérifiée après 226 itérations $0 \leq M_n - m_n \leq \varepsilon * m_n$ $(\varepsilon = 0.001)$ $(0.32<0.33)$.

Le coût moyen minimal est compris entre $(M_n ; m_n) = (331{,}297; 331{,}617)$.

La politique optimale : voir (Annexe 2).

4 éme cas $\lambda = 8$; $k = 25$

Pour n=1 : on aura v_1 (i, t)=

```
pour n=1
0  30  60  90  120  150  180  210  240  270  300
10  40  70  100  130  160  190  220  250  280  310
20  50  80  110  140  170  200  230  260  290  320
30  60  90  120  150  180  210  240  270  300  330
40  70  100  130  160  190  220  250  280  310  340
50  80  110  140  170  200  230  260  290  320  350
60  90  120  150  180  210  240  270  300  330  360
70  100  130  160  190  220  250  280  310  340  370
80  110  140  170  200  230  260  290  320  350  380
90  120  150  180  210  240  270  300  330  360  390
100  130  160  190  220  250  280  310  340  370  400
110  140  170  200  230  260  290  320  350  380  410
120  150  180  210  240  270  300  330  360  390  420
130  160  190  220  250  280  310  340  370  400  430
140  170  200  230  260  290  320  350  380  410  440
150  180  210  240  270  300  330  360  390  420  450
160  190  220  250  280  310  340  370  400  430  460
170  200  230  260  290  320  350  380  410  440  470
180  210  240  270  300  330  360  390  420  450  480
190  220  250  280  310  340  370  400  430  460  490
1440  1350  1260  1170  1080  990  900  810  720  630  540

( mn , Mn )=(0 , 1440)
```

Figure 12 : l'affichage de v (i, t) pour n=1

Le critère d'arrêt $0 \leq M_n - m_n \leq \varepsilon * m_n$ n'est pas v vérifiée car $(1440 \neq 0)$ donc on passe a l'étape suivante

Pour n=2 :

```
pour n=2
4.44444  64.4444  124.444  184.444  244.444  304.444  364.444  424.444  484.444
 544.444  604.444
24.4444  83.8889  143.889  203.889  263.889  323.889  383.889  443.889  503.889
 563.889  623.889
44.4444  103.889  163.333  223.333  283.333  343.333  403.333  463.333  523.333
 583.333  643.333
64.4444  123.889  183.333  242.778  302.778  362.778  422.778  482.778  542.778
 602.778  662.778
84.4444  143.889  203.333  262.778  322.222  382.222  442.222  502.222  562.222
 622.222  682.222
104.444  163.889  223.333  282.778  342.222  401.667  461.667  521.667  581.667
 641.667  701.667
124.444  183.889  243.333  302.778  362.222  421.667  481.111  541.111  601.111
 661.111  721.111
144.444  203.889  263.333  322.778  382.222  441.667  501.111  560.556  620.556
 680.556  740.556
164.444  223.889  283.333  342.778  402.222  461.667  521.111  580.556  640  700
  760
184.444  243.889  303.333  362.778  422.222  481.667  541.111  600.556  660  719
.444  779.444
204.444  263.889  323.333  382.778  442.222  501.667  561.111  620.556  680  739
.444  798.889
224.444  283.889  343.333  402.778  462.222  521.667  581.111  640.556  700  759
.444  818.889
244.444  303.889  363.333  422.778  482.222  541.667  601.111  660.556  720  779
.444  838.889
264.444  323.889  383.333  442.778  502.222  561.667  621.111  680.556  740  799
.444  858.889
284.444  343.889  403.333  462.778  522.222  581.667  641.111  700.556  760  819
.444  878.889
304.444  363.889  423.333  482.778  542.222  601.667  661.111  720.556  780  839
.444  898.889
324.444  383.889  443.333  502.778  562.222  621.667  681.111  740.556  800  859
.444  918.889
344.444  403.889  463.333  522.778  582.222  641.667  701.111  760.556  820  879
.444  938.889
364.444  423.889  483.333  542.778  602.222  661.667  721.111  780.556  840  899
.444  958.889
935.556  941.667  947.778  953.889  960  966.111  972.222  978.333  984.444  990
.556  996.667
2694.44  2532.44  2370.44  2208.44  2046.44  1884.44  1722.44  1560.44  1398.44
 1236.44  1074.44

( mn , Mn )=(4.44444 , 1254.44)
```

Figure 13 : l'affichage de v (i, t) pour n= 2

Le critère d'arrêt $0 \leq M_n - m_n \leq \varepsilon * m_n$ n'est pas vérifiée car $1249.996 > 0.00444$

Pour n=250

```
pour n=250
86403.2  86020.7  85714.2  85494.1  85382.3  85410.9  85612.9  86062.9  86512.9
 86962.9  87412.9
87253.9  86803.9  86429.8  86142.2  85962.9  85924  86058.5  86458.8  86908.8  8
7358.8  87808.8
88112.4  87662.4  87212.4  86848.9  86593.6  86478.8  86537.4  86863.9  87313.9
 87763.9  88213.9
88978.7  88528.7  88078.7  87628.7  87287  87085.7  87057.7  87298.3  87748.3  8
8198.3  88648.3
89851.8  89401.8  88951.8  88501.8  88060.2  87758.9  87631  87771.9  88211.6  8
8661.6  89111.6
90737.4  90287.4  89837.4  89387.4  88937.4  88518.7  88273.4  88296.8  88663.9
 89113.9  89563.9
91644.2  91194.2  90744.2  90294.2  89844.2  89394.2  89008  88890.6  89144.7  8
9594.7  90044.7
92569.7  92119.7  91669.7  91219.7  90769.7  90319.7  89869.7  89579.1  89681  9
0081.6  90531.6
93504.1  93054.1  92604.1  92154.1  91704.1  91254.1  90804.1  90402.2  90303.4
 90593.2  91043.2
94470.7  94020.7  93570.7  93120.7  92670.7  92220.7  91770.7  91320.7  91056.5
 91168.1  91561.7
95450.2  95000.2  94550.2  94100.2  93650.2  93200.2  92750.2  92300.2  91917.8
 91855.5  92118.1
96460.4  96010.4  95560.4  95110.4  94660.4  94210.4  93760.4  93310.4  92864.9
 92647  92764.4
97475.2  97025.2  96575.2  96125.2  95675.2  95225.2  94775.2  94325.2  93875.2
 93533.2  93500.4
98526.4  98076.4  97626.4  97176.4  96726.4  96276.4  95826.4  95376.4  94926.4
 94503.5  94326.1
99595.9  99145.9  98695.9  98245.9  97795.9  97345.9  96895.9  96445.9  95995.9
 95545.9  95241.6
100697  100247  99797  99347  98897  98447  97997  97547  97097  96647  96246.8

101842  101392  100942  100492  100042  99591.7  99141.7  98691.7  98241.7  9779
1.7  97341.7
103026  102576  102126  101676  101226  100776  100326  99876.4  99426.4  98976.
4  98526.4
104301  103851  103401  102951  102501  102051  101601  101151  100701  100251
99800.9
105665  105215  104765  104315  103865  103415  102965  102515  102065  101615
101165
107119  106669  106219  105769  105319  104869  104419  103969  103519  103069
102619

( mn , Mn )=(378.055 , 378.43)

dans cet exemple l'algorithme convergeapres250iteration avec le cout optimale il
 est entre  les deux borne mn et Mn378.055   378.43
```

Figure 14: l'affichage de v (i, t) pour n=250 et le coût moyen optimal

Le critère d'arrêt est vérifié après 250 itérations $0 \leq M_n - m_n \leq \varepsilon * m_n$ $(\varepsilon = 0.001)$ $(0.375 \leq 0.378)$.

Le coût moyen minimal est compris entre $(M_n ; m_n) = (378{,}055; 378.43)$

La politique optimale : voir (Annexe 2).

b. Conclusion

L'algorithme d'itération par valeur nécessite 174, 311, 226 et 250 itérations. et les limites respectives (m_n,M_n)=(319,34.319,652) ,(367,062.367,422) ,(331,297.331,617) et (378,055.378,44) , sur le cout moyen minimal.

3.2.2 Résolution avec l'algorithme d'itération par politique [1]

Pour résoudre le problème avec cet algorithme il faut suivre les étapes suivantes :

- Étape 00 : (initialisation) choisissez une politique stationnaire R
- Étape 01 : (étape de détermination de la valeur) pour la politique actuelle R, calculez le coût moyen g(R) et les valeurs relatives $v_{i,t}$ (R) tq $0 \leq i \leq M \quad et \quad 0 \leq t \leq s$ comme solution unique aux équations linéaire suivants :

$v_{i,t}= C_{it}(R_i) - g\tau_{i,t}(R_i) + \sum_{j\in I} P_{(i,t)(j,t)}\, v_{j,t}$ i∈ I , $0\leq t \leq s$.

$v_{s,t}= 0$.

Ou s est un état choisi arbitrairement.

- étape 02 : (étape d'amélioration de la politique) pour chaque état i∈ I déterminez une action a_i donnant le minimum :

$$\min_{0\leq a\leq s}\{c_{i,t}(a) - g\tau_{i,t}(a) + \frac{\lambda}{\lambda+\mu\min(i,a)}\, v_{i+1,a} + \frac{\min(i,a)\mu}{\lambda+\min(i,a)\mu}\, v_{i-1,a}\}$$

Par convention $v_{-1,t}=0 \ \forall\ 0 \leq t \leq s$.

Avec :

$\tau_{(i,t)}(a)=\frac{1}{\lambda+\min(i;a)\mu}$ $0\leqslant i\leqslant M-1$,$0\leqslant a\leqslant s$

$C_{(i,t)}(a)=k(t,a)+\frac{hi+ra}{\lambda+\min(i,a)\mu}$ $0\leqslant a\leqslant s$

Et

$$\tau_{(m,t)}(s)=\frac{1}{\lambda+s\mu}+\lambda/\lambda+s\mu(\frac{1}{s\mu-\lambda})=\frac{s\mu}{(\lambda+s\mu)(s\mu-)\lambda}$$

$$C_{(m,t)}(s)=k(t,s)+\frac{hM+rs}{\lambda+s\mu}+\frac{1}{\lambda+s\mu}\{\frac{h(M+1)+rs}{s\mu-\lambda}+\frac{h\lambda}{(s\mu-\lambda)^2}\}$$

La nouvelle politique stationnaire $\bar{R}$ s'obtient en choisissant $\bar{R}_i=a_i$ pour tout i∈ I avec la convention que $\bar{R}_i$ est choisi égal à l'ancienne action R_i lorsque cette action minimise la quantité d'amélioration de la politique.

- Étape 03 : (teste de convergence)

Si la nouvelle politique $\bar{R}$ =R alors l'algorithme est arrêté avec la politique R sinon passez à l'étape 01 avec R remplacer par$\bar{R}$.

a. Résultat numérique

On considère les mêmes données de l'exemple étudié :

S=10, μ =1, r=30, et h=1

Le taux d'arrivée λest 7 et 8.

Le coût de commutation k prend les deux valeurs 10 et 25, dans chaque exemple nous prenons la borne M=20.

L'algorithme d'itération par politique est initialisé avec la politique :

R= { (0,0) (1,1) (2,1) (3,2) (4,2) (5,3) (6,3) (7,4) (8,5) (9,5) (10,5) (11,6) (12,7) (13,8) (14,8) (15,9) (16,9) (17,9) (18,10) (19,10) (20,10) }

Itération 1

Etape 1 : (détermination de la valeur)

$v_{i,t}= C_{it}(R_i) -g\tau_{i,t}(R_i) +\sum_{j\in I} P_{(i,t)(j,t)}\, v_{j,t}$ i∈ I , $0\leq t\leq s$.

Le coût moyen est les valeurs relatives de la politique R sont calculées en résolvant les

équations linéaire

Pour l'état (0,0) on aura le sous système suivant :

$V_{0,0}(0)=0-g\frac{1}{7}+V_{1,0}+0$

$V_{0,1}(0)=10-g\frac{1}{7}+V_{1,0}+0$

$V_{0,2}(0)=20-g\frac{1}{7}+V_{1,0}+0$

$V_{0,3}(0)=30-g\frac{1}{7}+V_{1,0}+0$

$V_{0,4}(0)=40-g\frac{1}{7}+V_{1,0}+0$

$V_{0,5}(0)=50-g\frac{1}{7}+V_{1,0}+0$

$V_{0,6}(0)=60-g\frac{1}{7}+V_{1,0}+0$

$V_{0,7}(0)=70-g\frac{1}{7}+V_{1,0}+0$

$V_{0,8}(0)=80-g\frac{1}{7}+V_{1,0}+0$

$V_{0,9}(0)=90-g\frac{1}{7}+V_{1,0}+0$

$V_{0,10}(0)=100-g\frac{1}{7}+V_{1,0}+0$

Pour l'état (1,1) on aura le sous système suivant :

$V_{1,0}(1)=10+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

$V_{1,1}(1)=0+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

$V_{1,2}(1)=10+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

$V_{1,3}(1)=20+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

$V_{1,4}(1)=30+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

$V_{1,5}(1)=40+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

$V_{1,6}(1)=50+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

$V_{1,7}(1)=60+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

$V_{1,8}(1)=70+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

$V_{1,9}(1)=80+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

$V_{1,10}(1)90+\frac{40}{8}-g\frac{1}{8}+\frac{7}{8}V_{2,1}+\frac{1}{8}V_{0,1}$

Pour l'état (2,1) on aura le sous système suivant :

$V_{2,0}(1)=10+\frac{50}{8}-g\frac{1}{8}+\frac{7}{8}V_{3,1}+\frac{1}{8}V_{1,1}$

$V_{2,1}(1)=0+\frac{50}{8}-g\frac{1}{8}+\frac{7}{8}V_{3,1}+\frac{1}{8}V_{1,1}$

$V_{2,2}(1)=10+\frac{50}{8}-g\frac{1}{8}+\frac{7}{8}V_{3,1}+\frac{1}{8}V_{1,1}$

$V_{2,3}(1)=20+\frac{50}{8}-g\frac{1}{8}+\frac{7}{8}V_{3,1}+\frac{1}{8}V_{1,1}$

.

.

.

.

.

.

$V_{2,10}(1)=90+\frac{50}{8}-g\frac{1}{8}+\frac{7}{8}V_{3,1}+\frac{1}{8}V_{1,1}$

Pour l'état (3,2) on aura le sous système suivant :

$V_{3,0}(2)=20+10-g\frac{1}{9}+\frac{7}{9}V_{4,2}+\frac{2}{9}V_{2,2}$

$V_{3,1}(2)=10+10-g\frac{1}{9}+\frac{7}{9}V_{4,2}+\frac{2}{9}V_{2,2}$

$V_{3,2}(2)=0+10-g\frac{1}{9}+\frac{7}{9}V_{4,2}+\frac{2}{9}V_{2,2}$

$V_{3,3}(2)=10+10-g\frac{1}{9}+\frac{7}{9}V_{4,2}+\frac{2}{9}V_{2,2}$

.

.

.

.

.

.

$V_{3,10}(2)=80+10-g\frac{1}{9}+\frac{7}{9}V_{4,2}+\frac{2}{9}V_{2,2}$

Et ainsi de suite avec tous les états de la politique initiale.

Pour l'état (20,10) on aura le sous système suivant :

$V_{20,0}(10)=100+\frac{500}{17}-g\frac{1}{17}+\frac{7}{17}V_{21,10}+\frac{10}{17}V_{19,10}$

$V_{20,1}(10)=90+\frac{500}{17}-g\frac{1}{17}+\frac{7}{17}V_{21,10}+\frac{10}{17}V_{19,10}$

$V_{20,2}(10)=80+\frac{500}{17}-g\frac{1}{17}+\frac{7}{17}V_{21,10}+\frac{10}{17}V_{19,10}$

$V_{20,3}(10)=70+\frac{500}{17}-g\frac{1}{17}+\frac{7}{17}V_{21,10}+\frac{10}{17}V_{19,10}$

.

.

.

.

.

.

$V_{20,10}(10)=0+\frac{500}{17}-g\frac{1}{17}+\frac{7}{17}V_{21,10}+\frac{10}{17}V_{19,10}$

Remarque :

Dans ce cas on ne peut pas résoudre ce problème, car le nombre d'inconnues est supérieur au nombre d'équations dans le système.

Conclusion générale

Le modèle de décision semi markovien est la meilleure façon de résoudre un large éventail de problème d'optimisation.

Parmi les méthodes utilisées pour la recherche d'une politique optimale, nous avons cité les deux algorithmes : « Policy-iteration algorithm » et « value- iteration algorithm ».

A travers le problème que nous avons présenté « contrôle optimal d'un système de service stochastique à s serveurs », nous avons essayé de répondre à la question suivante :

Comment activer et désactiver les s serveurs ? Et sous quel critère ?

Les s serveurs peut être par exemple, des caisses dans un super marché, des guichets dans un bureau de poste ou des machines de production dans une usine.

L'utilisation de ses serveurs induit un coût, sous le critère de minimiser le coût de leur exploitation, nous avons vu qu'il est possible de mettre ce problème sous forme d'un modèle de décision semi markovien SMDP [1].

A partir du modèle établi par [1], nous avons réussi à proposer une solution par « value iteration algorithm » (en pratique cet algorithme nécessite un grand nombre d'itération pour converger). Par contre nous avons trouvé des difficultés dans l'application de « policy iteration algorithm » (cet algorithme nécessite moins d'itérations que le précédent).

Une autre alternative pour contourner cette dernière difficulté, consiste a réduire encore plus l'espace d'états et proposer un autre sous ensemble E inclus dans l'espace d'états I initialement proposer, par exemple :

E= {(0,0), (i_t, t) t=1,2,…. s ; $i_t \leq i_{t+1}$ }

L'état (i_t, t) veut dire que t serveurs sont activés si le nombre de clients dans le système i atteint le seuil i_t , ($i \geq i_t$).

Pour t=10, nous avons

E= {(0,0), (i_1,1), (i_2,2),…… (i_{10},10), $i_1 \leq i_2 \leq \cdots \leq i_{10}$ }.

Il s'agit donc, de chercher les seuils $i_1, i_2, \dots, i_{10}$ à partir des quels s serveurs seront activés, t=1,2,……,10.

Cette approche a été proposée par R.D.Nobel dans son article [7].

De nombreuses applications des processus de décision Markovien impliquent un espace d'état infini. Habituellement, cela obstrue l'utilisation de l'algorithme d'itération par politiques, puisque chaque étape de cet algorithme nécessite la solution d'un système d'équations linéaires dont la taille est égale à la dimension de l'espace d'États. Dans les applications structurées, il est parfois possible de contourner cette difficulté en se limitant à une sous-classe de politiques structurées et en utilisant une technique d'immersion. De cette façon, il peut être possible de développer un algorithme d'itération par politique sur mesure qui fonctionne sur une sous-classe de politiques structurées, où un seul système relativement petit d'équations linéaires doit être résolu à chaque étape [7].

Heureusement pour nous, l'objectif de recherche de la politique optimale de contrôle optimal d'un système de service stochastique a été atteint en un nombre fini d'itérations par l'algorithme d'itération par valeur « value iteration ».

Finalement, on peut conclure que l'algorithme d'itération par valeur (value iteration) est applicable pour n'importe qu'elle problème de décision semi markovien, ce qui n'est pas le cas pour la méthode d'itération par politique (policy iteration).

Annexe01 :

Unichain assumption

Un unichain est une chaine de Markov à l'état fini qui contient au plus une seule classe récurrente, et peut-être, certains états transitoires.

Un unichain est la généralisation naturelle d'une chaine récurrente permettre certains comportements transitoires initiaux sans perturber le comportement asymptotique à long terme de la chaine récurrente.

Théorème 7.1.1 :

Supposons que la chaine de Markov intégrée $\{x_n\}$ associée à la politique R n'a pas deux ensembles disjoints.

On a $\lim_{t\to\infty} \frac{z(t)}{t}$ =g(R) avec une probabilité =1, ou z(t) représente les coûts totaux induits jusqu'au temps t avec t>0.

Pour chaque état initial x_o=i, g(R) est donnée par :

G (R) =$\sum_{j\in I} C_j(R_j)\ \pi_j(R)\ /\ \sum_{j\in I}\ \tau_j(R_j)\ \pi_j(R)$ [1]

(La preuve de ce théorème ce trouve a la page 281 de [1])

Les deux formules (2.6.2) et (2.6.3) sont :

t_n = n E (τ_1) /1−λ E (τ_1) …… (2.6.2)

$h_n=\frac{h}{1-\lambda\ E\ (\tau_1)}\ [\frac{1}{2}$ n (n-1) E (τ_1) $+\frac{\lambda n E\ (E\ (\tau_1)^{\ 2})}{2[1-\lambda E\ (\tau_1)]}]$…… (2.6.3)

Annexe 02 : affichage des politiques stationnaires (optimales)

```
pour i= 0   0  1  2  3  4  4  4  4  4  4  4
pour i= 1   1  1  2  3  4  4  4  4  4  4  4
pour i= 2   2  2  2  3  4  5  5  5  5  5  5
pour i= 3   3  3  3  3  4  5  5  5  5  5  5
pour i= 4   3  3  3  3  4  5  6  6  6  6  6
pour i= 5   4  4  4  4  4  5  6  7  7  7  7
pour i= 6   5  5  5  5  5  5  6  7  8  8  8
pour i= 7   5  5  5  5  5  5  6  7  8  8  8
pour i= 8   6  6  6  6  6  6  6  7  8  9  9
pour i= 9   7  7  7  7  7  7  7  7  8  9  10
pour i= 10   7  7  7  7  7  7  7  7  8  9  10
pour i= 11   8  8  8  8  8  8  8  8  8  9  10
pour i= 12   9  9  9  9  9  9  9  9  9  9  10
pour i= 13   9  9  9  9  9  9  9  9  9  9  10
pour i= 14   10  10  10  10  10  10  10  10  10  10  10
pour i= 15   10  10  10  10  10  10  10  10  10  10  10
pour i= 16   10  10  10  10  10  10  10  10  10  10  10
pour i= 17   10  10  10  10  10  10  10  10  10  10  10
pour i= 18   10  10  10  10  10  10  10  10  10  10  10
pour i= 19   10  10  10  10  10  10  10  10  10  10  10
< mn , Mn >=<367.062 , 367.422>
```

Figure 15 : affichage de la politique optimale pour $\lambda = 8$; $k = 10$

```
pour i= 0   0  1  2  3  4  5  6  6  6  6  6
pour i= 1   1  1  2  3  4  5  6  6  6  6  6
pour i= 2   2  2  2  3  4  5  6  6  6  6  6
pour i= 3   3  3  3  3  4  5  6  6  6  6  6
pour i= 4   3  3  3  3  4  5  6  7  7  7  7
pour i= 5   4  4  4  4  4  5  6  7  7  7  7
pour i= 6   5  5  5  5  5  5  6  7  8  8  8
pour i= 7   5  5  5  5  5  5  6  7  8  8  8
pour i= 8   6  6  6  6  6  6  6  7  8  9  9
pour i= 9   6  6  6  6  6  6  6  7  8  9  9
pour i= 10   7  7  7  7  7  7  7  7  8  9  10
pour i= 11   7  7  7  7  7  7  7  7  8  9  10
pour i= 12   7  7  7  7  7  7  7  7  8  9  10
pour i= 13   8  8  8  8  8  8  8  8  8  9  10
pour i= 14   8  8  8  8  8  8  8  8  8  9  10
pour i= 15   8  8  8  8  8  8  8  8  8  9  10
pour i= 16   9  9  9  9  9  9  9  9  9  9  10
pour i= 17   9  9  9  9  9  9  9  9  9  9  10
pour i= 18   9  9  9  9  9  9  9  9  9  9  10
pour i= 19   10  10  10  10  10  10  10  10  10  10  10
< mn , Mn >=<331.297 , 331.617>
```

Figure 16 : affichage de la politique optimale pour $\lambda = 7$; $k = 25$

```
pour i= 0   0  1  2  3  4  5  6  6  6  6  6
pour i= 1   1  1  2  3  4  5  6  7  7  7  7
pour i= 2   2  2  2  3  4  5  6  7  7  7  7
pour i= 3   3  3  3  3  4  5  6  7  7  7  7
pour i= 4   3  3  3  3  4  5  6  7  8  8  8
pour i= 5   4  4  4  4  4  5  6  7  8  8  8
pour i= 6   5  5  5  5  5  5  6  7  8  8  8
pour i= 7   6  6  6  6  6  6  6  7  8  9  9
pour i= 8   6  6  6  6  6  6  6  7  8  9  9
pour i= 9   7  7  7  7  7  7  7  7  8  9  10
pour i= 10   7  7  7  7  7  7  7  7  8  9  10
pour i= 11   7  7  7  7  7  7  7  7  8  9  10
pour i= 12   8  8  8  8  8  8  8  8  8  9  10
pour i= 13   8  8  8  8  8  8  8  8  8  9  10
pour i= 14   9  9  9  9  9  9  9  9  9  9  10
pour i= 15   9  9  9  9  9  9  9  9  9  9  10
pour i= 16   10  10  10  10  10  10  10  10  10  10  10
pour i= 17   10  10  10  10  10  10  10  10  10  10  10
pour i= 18   10  10  10  10  10  10  10  10  10  10  10
pour i= 19   10  10  10  10  10  10  10  10  10  10  10
< mn , Mn >=<378.055 , 378.43>
```

Figure 17 : affichage de la politique optimale pour $\lambda = 8$; $k = 25$

Annexe03 : programmation.

Vous trouvez ci –dessous le programme d'itération par valeur élaboré en utilisant le langage C++ pour le problème de contrôle optimal d'un système de service stochastique

```
#include<iostream>

Using namespace std;

int main( )

{

Int  i,j,t,b,n,a,M,k,h,r,abs,abs1,s, u ;

float e=0.001,mn,Mn,min1,min2,min3,minn,p,p1,p2,p3,p4,l,q;
```

```
float A [50],T[50];

float v1[50][50],v2[50][50],v[50][50],v0[50][50];

cout<<"donner le nombre de serveur s"<<endl;

cin>>s;

cout<<"donner le taux d'arrive lambda et le taux de service u avec la condition que soit lambda soit supérieur à su"<<endl;

cin>>l>>u;

cout<<"donner le cout d'exploitation r ,le cout d'attente h et le cout de commutation k"<<endl;

cin>>r>>h>>k;

cout<<"donner M"<<endl;

cin>>M;

for(i=0;i<=M;i++){

      for(j=0;j<=s;j++)

      v0[i][j]=0;

   }

b=0;

n=1;

cout<<"pour n="<<n<<endl;

for(i=0;i<=M-1;i++){

        for(a=0;a<=s;a++){
```

```
          if(i<=a)
minn=i;
else
minn=a;
                //cout<<minn<<"   "<<i<<"   "<<a<<endl;
                    p=l/(l+s*u);
                    p1=(minn*u)/(l+s*u);
  if(i==0)
          T[a]=(l+minn*u)*k*a+h*i+r*a+p*v0[i+1][a]+(1-(p+p1))*v0[i][0];
else
      T[a]=(l+minn*u)*k*a+h*i+r*a+p*v0[i+1][a]+p1*v0[i-1][a]+(1-(p+p1))*v0[i][0];
      //cout<<"v1  "<<i<<"  "<<0<<"="<<v1[i][0]<<endl;
      }
     //for(j=0;j<=s;j++)
      //   cout<<T[j]<<"   ";
      //cout<<endl;
    min3=T[0] ;
       for(j=1;j<=s;j++)
         {if(T[j]<min3)
         min3=T[j];}
```

```
        v1[i][0]=min3;
for(t=1;t<=s;t++){
         for(a=0;a<=s;a++) {
         if(i<=a)
           min1=i;
else min1=a;
         if ((t-a)>0)
           abs=t-a;
else abs=a-t;
        // cout<<"abs"<<t<<"-"<<a<<"="<<abs<<endl;
p2=(min1*u)/(l+s*u);
         if(i==0)
         A[a]=(l+min1*u)*k*abs+h*i+r*a+p*v0[i+1][a]+(1-((l+min1*u)/(l+s*u)))*v0[i][t];
else
           A[a]=(l+min1*u)*k*abs+h*i+r*a+p*v0[i+1][a]+ p2*v0[i-1][a]+(1-(p+p2))*v0[i][t];}
       //cout<<endl;
        //for(j=1;j<=s;j++){
        // cout<< A[j]<<"   ";}
         //cout<<endl;
         min2=A[0];
```

```
        for(j=1;j<=s;j++)
        {if(A[j]<min2)
        min2=A[j];
           }
           v1[i][t]=min2;}
           }
           for(t=0;t<=s;t++)
           {if ((t-s)>0)
          abs1=t-s;
else abs1=s-t;
        p3=l+s*u;
        p4=s*u-l;
            v1[M][t]=(p3*p4*k*abs1)/(s*u)+((h*l)/p4)+h*M+r*s+(p4/p3)*v0[M-
1][s]+(l*p4/(s*u)*p3)*v0[M][s]+(1-(p4/(s*u)))*v0[M][t];
            }
                        for(i=0;i<=M;i++)
           {for (j=0;j<=s;j++)
            cout<<v1[i][j]<<"  ";
            cout<<endl;}
            cout<<endl;
              for(i=0;i<=M;i++)
```

```
                {
                   for(j=0;j<=s;j++)
                v[i][j]=v1[i][j]-v0[i][j];}
                                // for(i=0;i<=M;i++)
               // {
                //   for(j=0;j<=s;j++)
                  //    cout<<v[i][j]<<"  ";
            //  cout<<endl;}
             //   cout<<endl;
                mn=v[0][0];
for(i=0;i<=M;i++)
              { for (j=0;j<=s;j++)
              {
                 if (v[i][j]<mn)
                  mn=v[i][j];}}
                  Mn=v[0][0];
                  for(i=0;i<=M;i++)
              { for (j=0;j<=s;j++) {
                  if (v[i][j]>Mn)
                  Mn=v[i][j];
```

```
            }}
            cout<<"( mn , Mn )=("<<mn<<" , "<<Mn<<")"<<endl;
            cout<<endl;
while (b==0)
  {
  for(i=0;i<=M-1;i++){
      for(a=0;a<=s;a++){
        if(i<=a)
minn=i;
else
minn=a;
              //cout<<minn<<"   "<<i<<"   "<<a<<endl;
                  p=l/(l+s*u);
                  p1=(minn*u)/(l+s*u);
              if(i==0)
        T[a]=(l+minn*u)*k*a+h*i+r*a+p*v1[i+1][a]+(1-(p+p1))*v1[i][0];
else
      T[a]=(l+minn*u)*k*a+h*i+r*a+p*v1[i+1][a]+p1*v1[i-1][a]+(1-(p+p1))*v1[i][0];
      //cout<<"v1  "<<i<<"  "<<0<<"="<<v1[i][0]<<endl;
      }
```

```
 for(j=0;j<=s;j++)
    cout<<T[j]<<"   ";
  cout<<endl;
 min3=T[0] ;
    for(j=1;j<=s;j++)
     {if(T[j]<min3)
     min3=T[j];}
     v2[i][0]=min3;
for(t=1;t<=s;t++){
      for(a=0;a<=s;a++) {
      if(i<=a)
        min1=i;
else min1=a;
      if ((t-a)>0)
        abs=t-a;
else abs=a-t;
      // cout<<"abs"<<t<<"-"<<a<<"="<<abs<<endl;
p2=(min1*u)/(l+s*u);
q=p*v1[i+1][a]+(1-((l+min1*u)/(l+s*u)))*v1[i][t];
cout<<"q= "<<q<<"v i+1 a   v i t  "<<v1[i+1][a]<<"   "<<v1[i][t]<<"  "<< 7.0/17.0<<endl;
```

```
if(i==0)
A[a]=(l+min1*u)*k*abs+h*i+r*a+p*v1[i+1][a]+(1-((l+min1*u)/(l+s*u)))*v1[i][t];
else
A[a]=(l+min1*u)*k*abs+h*i+r*a+p*v1[i+1][a]+ p2*v1[i-1][a]+(1-(p+p2))*v1[i][t];}
cout<<endl;
for(j=0;j<=s;j++){
cout<< A[j]<<"   ";}
cout<<endl;
min2=A[0];
for(j=0;j<=s;j++)
{if(A[j]<min2)
min2=A[j];
}
v2[i][t]=min2;}
}
for(t=0;t<=s;t++)
{if ((t-s)>0)
abs1=t-s;
else abs1=s-t;
p3=l+s*u;
```

```
p4=s*u-l;
v2[M][t]=(p3*p4*k*abs1)/(s*u)+((h*l)/p4)+h*M+r*s+(p4/p3)*v1[M-1][s]+(l*p4/(s*u)*p3)*v1[M][s]+(1-(p4/(s*u)))*v1[M][t];
}
for(i=0;i<=M;i++)
{for (j=0;j<=s;j++)
cout<<v2[i][j]<<"  ";
cout<<endl;}
cout<<endl;
for(i=0;i<=M;i++)
{
for(j=0;j<=s;j++)
v[i][j]=v2[i][j]-v1[i][j];}
// for(i=0;i<=M;i++)
// {
// for(j=0;j<=s;j++)
// cout<<v[i][j]<<"  ";
// cout<<endl;}
// cout<<endl;
mn=v[0][0];
for(i=0;i<=M;i++)
```

```
{ for (j=0;j<=s;j++)
{
   if (v[i][j]<mn)
    mn=v[i][j];}}
    Mn=v[0][0];
    for(i=0;i<=M;i++)
{ for (j=0;j<=s;j++) {
    if (v[i][j]>Mn)
    Mn=v[i][j];
}}
cout<<"( mn , Mn )=("<<mn<<" , "<<Mn<<")"<<endl;
cout<<endl;
//cout<<"r  h  k  l  u  "<<r<<h<<k<<l<<u<<endl;
if(n==1)
 b=1;
else
 {n=n+1;
cout<<"pour n="<<n<<endl;
for(i=0;i<=M;i++)
{for(j=0;j<=s;j++)
```

```
            v1[i][j]=v2[i][j];
            }
            //for(i=0;i<=M;i++)
             //   {
               //     for(j=0;j<=s;j++)
                 //     cout<<v1[i][j]<<"  ";
           //  cout<<endl;}
            // cout<<endl;
            }
}
```

cout<<"dans cet exemple l'algorithme convergeapres"<<n<<"iteration avec les deux borne mn et Mn"<<mn<<"   "<<Mn<<endl;}

Bibliographie

[1] Henk C. Tijms, john Wiley ,A First Course in Stochastic Models, Vrije Universiteit, Amsterdam, The Netherlands (2003).

[2] ZeKRI YASSINE –Lekhal Sami, gestion optimale d'une centrale électrique & problème de maintenance par les processus de décision Markovien université blida1 (2016 -2017).

[3] Sébastien Loust, chaine de Markov et processus markoviens de sauts (application aux files d'attente) à l'école Centrale de Marseille (2008-2009).

[4] Adriana TAPUS, utilisation de processus de décision markoviens pour la planification et l'exécution d'actions par un robot mobile (20 juin 2002).

[5] PaulWeng, Processus décisionnels de Markov des récompenses ordinales au multicritère, Formalismes et modèles en PDA, pages 505 à 524

[6] Thomas Huber, Arnaud Maret, équations fonctionnelles, (2017).

[7] Rein D. Nobel, Henk C. Tijms, Optimal control for an $M^X/G/1$ queue with two service modes, European Journal of Operational Research 16 March 1999, Pages 610-619