الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

جامعة سعد دحلب البليدة-1-
Université Saad Dahlab Blida -1-

Mémoire de fin D'études

En vue de l'obtention du diplôme Master

Faculté de sciences
Département : Mathématiques

Spécialité : Modélisation stochastique et statistique

THEME

Allocation optimale des serveurs aux clients concurrents par les processus de décision semi-markovien

Présenté par :

ARAB Linda

ESSAHELI Ishak

Soutenu le : 23/ 09 / 2020

Devant le Jury :

Président : R. FRIHI	M.A.A	**Université de Blida 1**
Promotrice : Z. DAHMANE	M.A.A	**Université de Blida 1**
Examinatrice : H. OUKID	M.C.B	**Université de Blida 1**

Promotion : 2019/2020

Remerciements

Avant tout, nous remercions ALLAH, le créateur, l'omniscient, l'omniprésent, digne des meilleurs noms qu'il s'est attribué à lui-même.
C'est à lui que nous devons tout, le souffle de la vie et la lumière des yeux. Nous n'oublions pas son messager Mohamed paix de Dieu sur lui, envoyé en signe de miséricorde. C'est par la grâce d'ALLAH ce travail à été achevé, de nombreuses personnes y ont contribué et nous tenons à remercier :
Madame Dahmane notre promotrice, pour avoir dirigé ce travail, nous tenons à remercier pour son aide précieuse, son objectivité, sa disponibilité, sa rigueur scientifique et ses conseils qui ont fait progresser ce travail.
Les membres de jury composé de Madame Oukid et Monsieur Frihi qui ont aimablement accepté de faire partie de cette soutenance.
Merci à toutes les personnes ayant suivi de près ou de loin notre travail.

Dédicace

Je dédie ce modeste travail
A mon père
Mon plus haut exemple et mon modèle de persévérances pour aller toujours de l'avant et ne jamais baisser les bras et pour son enseignement.
A ma mère
Pour son affection, sa patience, sa compréhension, sa disponibilité, son écoute permanente et son soutien
Mes chers parents que dieu vous garde
A mon cher frère et mes adorables sœurs qu'aucun mot ne pourra décrire leur dévouement et leur soutien pour vous exprimer toute mon affection et ma tendresse
A mes collègues, à mon binôme, à ma famille et à ceux qui me sont chers
A mes chers amis que l'amitié sincère nous à liées, en témoignant les bons moments passés ensemble.

ESSAHELI ISHAK

Dédicace

Je dédie ce modeste travail

A Mes parents, qui ont œuvré pour ma réussite, de par leur amour, leurs soutiens, leurs sacrifices consentis et leurs précieuse conseils, pour toute leurs assistances et leurs présences dans notre vie, reçoivent à travers ce travail aussi modeste soit-il, l'expression de nos sentiments et de gratitude.

A mes adorables sœurs : Samar, Malak, et mon frère Walid

Pour vos encouragements continus, j'espère que j'étais un bon exemple pour vous et que vous aurez fait plus que moi.

A mon grand-père et ma grand-mère.

A mes oncles et tantes : Hamimi, Slimane, Mohamed, Youcef, Samia, Fatima.

A mon binôme Ishak.

A mes amies qui m'ont encouragé : Meriem, Sara, Youssra, Soumia, Asmaa, Fatiha, Nora, pour les sympathiques moments qu'on a passés ensemble.

A ma promotion et tous les étudiants de MSS.

ARAB LINDA

ملخص:

تندرج هذه الأطروحة تحت موضوع العمليات العشوائية، وبصورة أدق "عمليات اتخاذ القرار شبه الماركوفية".
تتمثل مشكلتنا في "التخصيص الأمثل للعملاء المتنافسين" في العثور على قاعدة تحكم تقلل من متوسط معدل الرفض أو ما يعادل ازدياد متوسط معدل نقل الرسائل المقبولة.
من بين الطرق المستخدمة للبحث عن السياسة المثلى، اخترنا "خوارزمية تكرار القيمة". تستخدم هذه الطريقة التي تتكون من حساب (تكراري) تسلسل القيم التي تقارب الحد الأدنى لمتوسط التكلفة (في حالتنا، الحد الأدنى لمتوسط معدل الرفض).
ميزة هذه الطريقة هي أنها سهلة التطبيق، علاوة على أنها تتجنب الكثير من العمليات الحسابية.

Résumé :

Notre problème « Allocation optimale des serveurs aux clients concurrents » consiste à trouver une règle de contrôle qui minimise le taux de rejet moyen ou de manière équivalente, maximise le débit moyen des messages acceptés.

Parmi les méthodes utilisées pour la recherche de politique optimale, nous avons choisi «value itération ». Cette méthode utilise l'approche récursive qui consiste à calculer (de manière récursive) une séquence de valeurs qui se rapproche du coût moyen minimal (dans notre cas, le taux de rejet moyen minimal).

L'avantage de cette méthode, c'est qu'elle est simple à appliquer, par ailleurs, elle permet également d'éviter beaucoup de calculs**.**

Abstract:

Our problem of "Optimal allocation of servers to concurrent clients" consists in finding a control rule that minimizes the average rejection rate or, equivalently, maximizes the average throughput of accepted messages.

Among the methods used for optimal policy research, we chose "value iteration". This method uses the recursive approach which consists in calculating (recursively) a sequence of values that approximates the minimum average cost (in our case, the minimum average rejection rate).

The advantage of this method is that it is simple to apply, and it also avoids a lot of calculations.

Notation et abréviation

MDP : Markov décision process.

SMDP : Semi -Markov décision 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.

$\boldsymbol{X_n}$: L'état du système au n^{iem} instant de décision.

$\boldsymbol{P_{ij}(R_i)}$: 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 ».

$\boldsymbol{g_i(R)}$: L'espérance du coût moyen (si l'état initial est i).

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

μ : Taux de service.

λ : Taux d'arrivé.

$\boldsymbol{g^*}$: Le taux de perte moyen.

Li : Le nombre de client de type non prioritaire où i c'est le nombre de client de type prioritaire dans le système.

$\boldsymbol{v_n(i1, i2, k)}$: L'espérance de coût total sur les n premiers instants de décision

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

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

.$\boldsymbol{\tau_i(a)}$**:** Le délai prévu jusqu'au prochain instant de décision.

.$\boldsymbol{C_i(a)}$: Les coûts attendus encourus jusqu'au prochain instant de décision.

Listes des figures

Sommaire

Table des matières

Introduction générale

Les processus décisionnels de Markov (MDP) constituent un cadre fréquemment utilisé pour formaliser les processus de décisions séquentielles impliquant un agent (fournisseur de service, gestionnaire du système, contrôleur) en interaction avec un environnement ou un système dynamique.

Le processus de décision semi Markovien est un modèle très étudié en file d'attente, il offre un formalisme pour modéliser et résoudre beaucoup de problèmes dans l'incertain.

Un processus de décision markovien (Markov décision processus), ou semi-markovien (semi-Markov décision process) est un modèle aléatoire où un agent prend des décisions où les résultats de ses actions sont aléatoires.

Les MDP sont une extension des chaînes de Markov avec plusieurs actions à choisir par état et où des récompenses sont gagnées (ou des coûts sont engendrés) par l'agent.

Les MDP sont utilisés pour étudier des problèmes d'optimisation à l'aide d'algorithmes de programmation dynamique ou d'apprentissage par renforcement dans de nombreuses disciplines, notamment la robotique, l'automatisation, l'économie et l'industrie manufacturière.

Ce mémoire comporte trois chapitres :

Le premier chapitre contient quelque définitions et éléments de base du modèle de décision markovien avec un petit résumé d'un problème de maintenance qui a fait l'objet d'un mémoire de master 2018 [3].

Dans le deuxième chapitre, nous présentons les notions de base du model de décision semi-markovien, la méthode de transformation des données et l'algorithmes de résolution" itération par valeur ".

Nous terminons ce chapitre par un exemple d'application en file d'attente « contrôle optimale d'un système de service stochastique » tiré d'un mémoire de master 2019 [2].

Le dernier chapitre est consacré à la description, la modélisation et la résolution numérique de notre problème : " Allocation optimale des serveurs aux clients concurrents".

Chapter 1

Processus de décision markovien

1.1 Introduction :

Les processus de décision markovien MDP ont fait l'objet d'un mémoire de master 2018 [3], institulé"gestion optimal d'une centrale électrique et problème de maintenance par les processus de décision markovien".

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

Dans ce chapitre, nous donnons un bref aperçu sur les MDP suivi d'un exemple de maintenance [3] afin de donner une idée sur le type de problème qui peuvent être resolus par cet outil.

1.2 Définition :

1.2.1 Processus stochastique [2] :

Un processus stochastique $X = (X_t)_{t\in T}$est une famille de variables aléatoires X_t indexée par un ensemble T.

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

1.2.2 Chaine de Markov [4] :

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

$$P(X_{n+1} = j/X_n = i, X_{n-1} = i_{n-1}, ..., X_1 = i_1, X_0 = i_0) = P(X_{n+1} = j/X_n = i)$$

$$\forall n \in N; \forall j, i_0, i_1, ..., i_{n-1}, i \in E$$

1.2.3 Processus de décision markovien [2] :

Un processus de décision Markovien est un processus de contrôle 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 Systéme dynamique [2] :

Un système dynamique est la donnée d'un système et d'une loi décrivant l'évolution de ce système, par exemple: 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 [2] :

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

1.2.6 Politique [3] :

Une politique ou stratégie (choix d'une action ou décision) pour chaque état, c'est un ensemble de règle ‹‹ if stat then action ››.

1.2.7 Politique Stationnaire [3] :

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

1.2.8 Politique Optimale [3] :

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

1.3 Le modèle :

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 systéme dynamique avec des instants équidistants du temps t =0,1,2,...

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

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 systéme :

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

-Au prochain instant de décision ,le systéme sera dans l'état j avec une probabilité $P_{ij}(a)$ tq :

$$\sum_{j \in I} P_{ij}(a) = 1, i \in I$$

On note que le paiment en une étape coûte $C_i(a)$, et les probabilités de transitions en une étape $P_{ij}(a)$ sont supposées être homogénes dans le temps.

Le modèle de décision markovien a de nombreuses applications pontentielles : contrôle des stocks, maintenance, fabrications, télécommunications.[1]

1.3.1 Problème de maintenance [2] :

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 la condition de travail i est meilleure que la condition de travail $i+1$.

L'équipement se 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'equipement sera dans la condition de fonctionnement j (l'état j) avec une probabilité q_{ij}.

on suppose que $q_{ij} = 0$ pour $j \prec i$ et $\sum_{j \prec i} q_{ij} = 1$.

La condition de travail i=N représente un dysfonctionnement nécessitant une réparation forcée prenant deux jours.

Pour les états intermédiaires i avec $1 \prec i \prec N$, 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 systéme réparé a la condition de fonctionnement $i = 1$, le coût d'une réparation forcée en cas de panne est C_f et le coût d'une réparation préventive en état de marche i est C_{pi}.

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

Pour plus des détails sur la modélisation et la resolution de ce problème voir TIJMS [1] et mémoire de master 2018 [3].

Chapter 2

Processus de décision semi-markovien

2.1 Introduction :

Dans le chapitre précédent, nous avons vu que dans un processus de décision markovien, les décision ne peuvent être prises qu'à des instant bien fixes t=0,1,... .

Cependant, il ya beaucaup de problémes de contrôle stochastique où les temps entre les instants de décision sont aléatoires. Un outil possible pour analyser de tels problèmes est le modèle de décision semi-markovien .

Dans la section 2.2, nous discutons les éléments de base de ce modèle. De plus, pour le critère d'optimalité du coût moyen à long terme par unité de temps, nous donnons une méthode de transformation des données par laquelle le modèle de décision semi-markovien peut être converti en un modèle de décision markovien à temps discret équivalent .

Dans la section 2.3, nous discutons l'algorithme d'itération par valeur pour un modèle de décision semi-markovien dans lequel les temps entre les instants de décision sont distribués de façon exponentielle.

Dans ce cas particulier, l'effort de calcul de l'algorithme d'itération par valeur peut être considérablement réduit en introduisant des instants de décision fictifs.

Cette astuce simple crée des matrices de transition conduisant à un algorithme d'itération par valeur beaucoup plus efficace.

2.2 Le modéle de décision Semi-markovien:

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

À ces instants, une décision doit être prise et des frais sont encourus en conséquence de la décision prise.

L'ensemble des états possibles est noté par I. Pour chaque état $i \in I$, un ensemble $A(i)$ d'actions possibles est disponible.

On suppose que l'espace d'état I et les ensembles 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 à un instant de décision l'action a est choisie dans l'état i, alors le temps jusqu'au prochain instant de décision ne dépendent que de l'état actuel i et l'action choisie a et sont donc indépendants de l'histoire passée du système.

De plus, les coûts engagés jusqu'au prochain instant de décision ne dépendent que de l'état actuel et de l'action choisie dans cet état.

Nous notons que dans des problèmes spécifiques, l'état se produisant à la transition suivante dépendra souvent du temps jusqu'à cette transition.

2.2.1 Le coût moyen à long terme par unité de temps :

Le coût moyen à long terme par unité de temps est considéré comme le critère d'optimalité. Pour ce critère, le modèle de décision semi-markovien est en fait déterminé par les caractéristiques suivantes :

$P_{ij}(a)$ = la probabilité qu'un prochain instant de décision le système soit dans l'état j si l'action a est choisie dans l'état actuel i.

$\tau_i(a)$ = le temps attendu 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 instatnt de décision si l'action a est choisie dans l'état actuel i.

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

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

Puisque l'espace d'état est 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 alors que dans une politique stationnaire R, le processus stochastique incluse $(X_n)_{n\geq 0}$ est une chaîne de Markov à temps discret avec des probabilités de transition en une étape $P_{ij}(R_i)$.

on fixe maintenant une politique stationnaire R, notez par :

$V_n\,(i1, i2, k)$ = l'espérance de coût total sur les n premiers instants de décision.

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

$g_i\,(R) = \lim_{n\to\infty} \frac{V_n(i1,i2,k)}{n} \quad i \in I$ cette limite existe d'aprés le théoreme suivant

Théorème 1 [1] :

Supposons que la chaîne de Markov intégrée $(X_n)_{n\geq 0}$ associée à la politique R n'a pas deux ensembles fermés disjoints.

Alors:

$lim_{t\to\infty} \frac{Z(t)}{t} = g(R)$ avec une probabilité 1.

pour chaque état initial $X_0 = i$, où la constante $g_i(R)$ est donnée par

$$g_i(R) = \sum_{j\in I} c_j(R_j)\pi_j(R) / \sum_{j\in I} \tau_j(R_j)\pi_j(R)$$

avec$\{\pi_j(R)\}$indiquant la distribution d'équilibre de la chaîne de Markov $(X_n)_{n\geq 0}$.

2.2.2 La méthode de transformation des données :

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

Considérons maintenant le modèle de décision de Markov en temps discret dont les éléments de base sont donnés par

$$\bar{I} = I \ , \ \bar{A}(i) = A(i) \qquad i \in \bar{I},$$

$$\bar{c}_i(a) = c_i(a)/\tau_i(a) \quad a \in \bar{A}(i) \ et \ i \in \bar{I},$$

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

Ce modèle de décision de Markov à temps discret a la même classe de politiques stationnaires que le modèle de décision semi-Markov original.

Pour chaque politique stationnaire R, soit $\bar{g}_i(R)$ le coût moyen à long terme par unité de temps dans le modèle à temps discret lorsque la politique R est utilisée et que l'état initial est i. Il en va de même pour chaque politique stationnaire R qui

$$g_i(R) = \bar{g}_i(R), i \in I. \qquad (*)$$

Ce résultat ne nécessite aucune hypothèse sur la structure de chaîne des chaînes de Markov associées aux politiques stationnaires. Cependant, nous prouvons le résultat $(*)$ uniquement pour le cas "unichain " (chaine unique voir définition Annexe01).

Fixer une politique stationnaire R et supposer que la chaîne de Markov intégrée $(X_n)_{n\geq 0}$ dans le modèle semi-Markov n'a pas deux ensembles fermés disjoints.

Notons $\bar{X}_n$ l'état au la nième instant de décision dans le modèle discret en temps transformé.

On voit directement que la chaîne de Markov $\left(\bar{X}_n\right)_{n\geq 0}$ est également "unichain "sous la politique R.

Les probabilités d'équilibre $\bar{\pi}_j(R)$ de la chaîne de Markov $\left(\bar{X}_n\right)_{n\geq 0}$ satisfont les équations d'équilibres :

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

Selon le théorème 1, $\bar{g}(R) = g(R)$. Ainsi, nous pouvons conclure qu'une politique optimale de coût moyen dans le modèle semi-markovien peut être obtenue en résolvant un modèle de décision de Markov en temps discret approprié.

Cette conclusion est particulièrement utile en ce qui concerne l'algorithme d'itération par valeur . En appliquant ce dernier au modèle transformé, il n'y a aucune restriction à supposer que pour chaque politique stationnaire, la chaîne de Markov associée $\left(\bar{X}_n\right)_{n\geq 0}$ est apériodique.

En choisissant la constante τ strictement inférieure à $min_{i,a}\ \tau_i(a)$, on a toujours $P_{ii}(a) > 0$ pour tout i, a et donc l'apériodicité requise.

2.3 Algorithmes pour une politique optimale :

La résolution d'un modéle de décision markovien et semi-markovien peut se faire moyennant deux algorithmes :

" itération par politique " (policy-iteration) et " iteration par valeur " (value-itération).

Le premier algorithme génère une séquence de politiques améliorées, comme son nom l'indique. Le second calcule de maniére recursive une sequence de valeurs (value function) qui se rapproche du coût moyen minimal.

Ces valeurs (the value functions) donnent les bornes inf et sup du coût moyen minimal.

L'avantage de ce dernier algorithme (iteration par valeur) , c'est qu'il est plus simple à appliquer. D'autre part , l'algorithme ‹‹ algorithme d'itération par politique ›› nécessite à chaque iteration, la résolution d'un système d'équation linéaire de même taille que l'espace des états. Ce qui n'est pas le cas pour le second algorithme "iteration par valeur ".

Dans ce qui suit , nous allons rappeler un seul algorithme "iteration par valeur ", qui sera utilisé pour la résolution de notre problème.

Une étude exhaustive de ces deux algorithme est disponible dans TI-JMS[1] , ainsi que dans les deux mémoires de master [2] et [3].

2.3.1 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 coûts minimaux attendus au cours des n premiers instants de décision ne prend pas en compte les temps de transition non identiques et donc ces coûts ne peuvent pas être liés au coût moyen minimal par unité de temps.

Cependant, par la méthode de transformation des données de la section 2.2.2, nous pouvons convertir le modèle de décision semi-Markov en un modèle de décision markovien à temps discret de sorte que les deux modèles ont le même coût moyen pour chaque politique stationnaire.

Dans le modèle à temps discret, il n'y a aucune restriction à supposer que tous les $\bar{c}_i(a) = c_i(a)/\tau_i(a)$ sont positifs; sinon, ajoutez une constante positive suffisamment grande à chaque $\bar{c}_i(a)$.

Algorithme d'itération par valeur :

Étape 0 : Choisissez $V_0(i)$ tel que $0 \leq V_0(i) \leq min_a\{c_i(a)/\tau_i(a)\}$ pour tout i . Choisissez un nombre τ avec $0 < \tau \leq min_{i,a}\tau_i(a)$. Soit $n := 1$.

Étape 1 : Calculez la fonction $V_n(i)$, $i \in I$, à partir de

$$V_n(i) = min_{a \in A(i)} \left[\frac{c_i(a)}{\tau_i(a)} + \frac{\tau}{\tau_i(a)} \sum_{j \in I} P_{ij}(a) V_{n-1}(j) + \left(1 - \frac{\tau}{\tau_i(a)}\right) V_{n-1}(i) \right] \qquad (**)$$

.

Soit $R(n)$ une politique stationnaire dont les actions minimisent le côté droit de $(**)$.

Étape 2 : Calculez 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
$0 \leq (M_n - m_n) \leq \varepsilon m_n$, où ε est un nombre de précision prédéfini. Sinon, passez à l'étape 3.

Étape 3 : $n := n + 1$ et passez à l'étape 1.

Supposons que l'hypothèse unichain faible de la section 6.5 [1] soit satisfaite pour les chaînes de Markov intégrées $\{Xn\}$ associées aux politiques stationnaires . Il n'y a aucune restriction à supposer que les chaînes de Markov $\{\bar{X}n\}$ dans le modèle transformé sont apériodiques . Ensuite, l'algorithme s'arrête après un nombre fini d'itérations avec une politique $R(n)$ dont la fonction de coût moyen $g_i(R(n))$ est satisfaisante

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

où g^* représente le coût moyen minimal par unité de temps.

Concernant le choix de τ dans l'algorithme, il est recommandé de prendre $\tau = min_{i,a}\tau_i(a)$ lorsque les chaînes de Markov intégrées {Xn} dans le modèle semi-Markov sont apériodiques; sinon, $\tau = \frac{1}{2} min_{i,a}\tau_i(a)$ est un choix raisonnable.

2.4 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.

Dans ce qui suit, 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.

Une étude détaillée de ce système a fait l'object d'un mémoire de master 2019 [2].

2.4.1 Contrôle optimal d'un systéme de service stochastique [2] :

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 $S\mu$.

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

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

minimal.

Pour plus des détails sur la modélisation et la resolution de ce problème voir mémoire de master 2019 [2].

Chapter 3

Application et programmation

3.1 Itération par la valeur et décisions fictives :

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

Dans chaque itération de la méthode, les bornes inférieure et supérieure indiquent à quel point le coût moyen de la politique actuelle s'écarte du coût moyen minimal.

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

De par la nature même de l'algorithme d'itération par valeurs, il est fastidieux sur le plan des calculs d'avoir de nombreux $P_{ij}(a)$ non nuls.

Dans les applications avec des temps répartis exponentiellement entre les époques de décision, l'effort de calcul de l'algorithme d'itération par valeur peut souvent être considérablement réduit en incluant des instants de décision dites fictifs.

L'état du système reste inchangé aux instants de décision fictifs.

L'inclusion d'instants de décision fictifs ne change pas la nature markovienne du processus de décision, car les temps entre les transitions d'états sont distribués de façon exponentielle et ont donc la propriété sans mémoire.

L'astuce des instants de décision fictifs réduit non seulement l'effort de calcul, mais simplifie également la formulation de l'algorithme d'itération par valeur.

3.2 Allocation optimale des serveurs aux clients concurrents :

Dans les réseaux de communication, un problème important est l'allocation des serveurs aux classes de clients concurrents.

Supposons que les messages de types 1 et 2 arrivent à un système de communication selon des processus de Poisson indépendants avec des débits respectifs $\lambda 1$ et $\lambda 2$.

Le système de communication a c canaux de transmission identiques pour traiter les messages, chaque canal ne pouvant traiter qu'un seul message à la fois.

Le système n'a pas de tampon pour stocker temporairement des messages qui trouvent tous les canaux occupés à l'arrivée.

De tels messages doivent être rejetés de toute façon.

Cependant, un nouveau message peut également être rejeté lorsqu'il existe un canal libre.

L'objectif est de trouver une règle de contrôle qui minimise le taux de rejet moyen, ou de manière équivalente, maximise le débit moyen des messages acceptés.

La théorie de la décision de Markov nous permet d'appliquer une approche optimale globale (overall optimal policy).

Pour ce faire, nous supposons que les temps de transmission des messages sont distribués de façon exponentielle avec une moyenne de $1/\mu 1$ pour les messages de type 1 et avec une moyenne de $1/\mu 2$ pour les messages de type 2.

3.2.1 Modélisation [1] :

Formulation avec des instants de décision fictives :

Une formulation simple du problème comme un problème de décision semi-markovien utilise les instants d'arrivée comme les seules instants de décision.

Dans une telle formulation, les vecteurs $(P_{ij}(a), j \in I)$ des probabilités de transition en une étape ont de nombreuses entrées non nulles.

Dans notre problème spécifique, cette difficulté peut être contournée en incluant les instants d'achèvement de service comme des instants de décision fictifs en plus des instants de décision réelles, étant les instants d'arrivée des messages.

Ce faisant, une transition de n'importe quel état se fait toujours vers l'un des quatre etats voisins au plus.

Dans l'approche aux instants de décision fictives, nous prenons comme espace d'états

$I = \{(i_1, i_2, k) \ | i_1, i_2 = 0, 1, ..., c \ ; i_1 + i_2 \leq c \ ; k = 0, 1, 2\}$

.

L'état (i_1, i_2, k) avec $k = 1$ ou 2 correspond à la situation dans laquelle un message de type k arrive et trouve i_1 messages de type 1 et i_2 messages de type 2 en cours de transmission.

L'état auxiliaire $(i_1, i_2, 0)$ correspond à la situation dans laquelle la transmission d'un message vient de se terminer et les messages i_1 de type 1 et i_2 de type 2 en transmission.

Pour les états (i_1, i_2, k) avec $k = 1$ ou 2, les actions possibles sont notées par

$$a = \begin{cases} 0 \ , & \text{rejeter le message ,} \\ 1 \ , & \text{accepter le message ,} \end{cases}$$

avec la condition que $a = 0$ est la seule décision possible lorsque $i_1 + i_2 = c$ et également pour l'état $s = (i_1, i_2, 0)$.

Grâce aux instants de décision fictifs, chaque transition d'un état donné se fait vers l'un au plus des quatre états voisins.

En d'autres termes, la plupart des probabilités de transition en une étape sont nulles.

De plus, les probabilités de transition sont extrêmement faciles à spécifier, on a le min (X_1, X_2) est distribué de façon exponentielle avec la moyenne $1/(\alpha_1 + \alpha_2)$ et $P\{X_1 < X_2\} = \alpha_1/(\alpha_1 + \alpha_2)$ lorsque X_1 et X_2 sont des variables aléatoires indépendantes ayant des distributions exponentielles avec les moyens respectifs $1/\alpha_1$ et $1/\alpha_2$.

Notons par :

$v(i_1, i_2) = \lambda_1 + \lambda_2 + i_1\mu_1 + i_2\mu$.

Si le systéme à l'état $s = (i_1, i_2, k)$ avec $k = 0, 1, 2$ et l'action $a = 0$ est choisie, alors

$$p_{sv}(0) = \begin{cases} \lambda_1/v(i_1, i_2), & v = (i_1, i_2, 1), \\ \lambda_2/v(i_1, i_2), & v = (i_1, i_2, 2), \\ i_1\mu_1/v(i_1, i_2), & v = (i_1 - 1, i_2, 0), \\ i_2\mu_2/v(i_1, i_2), & v = (i_1, i_2 - 1, 0). \end{cases}$$

et $\tau_s(0) = 1/v(i_1, i_2)$

Si le systéme à l'état $s = (i_1, i_2, 1)$ et l'action $a = 1$ est choisie, alors

$$p_{sv}(1) = \begin{cases} \lambda_1/v(i_1+1,i_2), & v=(i_1+1,i_2,1), \\ \lambda_2/v(i_1+1,i_2), & v=(i_1+1,i_2,2), \\ (i_1+1)\mu_1/v(i_1+1,i_2), & v=(i_1,i_2,0), \\ i_2\mu_2/v(i_1+1,i_2) & v=(i_1+1,i_2-1,0). \end{cases}$$

et $\tau_s(1) = 1/v(i_1+1,i_2)$, avec $v(i_1+1,i_2) = \lambda_1+\lambda_2+(i_1+1)\mu_1+i_2\mu_2$

Si le systéme à l'état $s=(i_1,i_2,2)$ et l'action $a=1$ est choisie, alors

$$p_{sv}(1) = \begin{cases} \lambda_1/v(i_1,i_2+1), & v=(i_1,i_2+1,1), \\ \lambda_2/v(i_1,i_2+1), & v=(i_1,i_2+1,2), \\ i_1\mu_1/v(i_1,i_2+1), & v=(i_1-1,i_2+1,0), \\ (i_2+1)\mu_2/v(i_1,i_2+1) & v=(i_1,i_2,0). \end{cases}$$

et $\tau_s(1) = 1/v(i_1,i_2+1)$, avec $v(i_1,i_2+1) = \lambda_1+\lambda_2+i_1\mu_1+(i_2+1)\mu_2$

Enfin, les coûts attendus en une étape $c_s(a)$ sont simplement donnés par

$$c_s(a) = \begin{cases} 1 \ , & s=(i_1,i_2,1) \ et \ a=0 \ , \\ 1 \ , & s=(i_1,i_2,2) \ et \ a=0 \ , \\ 0 \ , & sinon \ . \end{cases}$$

3.2.2 Algorithme d'itération par valeur :

Maintenant, après avoir spécifié les éléments de base du modèle de décision semi-markovien, nous sommes en mesure de formuler l'algorithme d'itération par valeur pour le calcul d'une règle d'acceptation (presque) optimale.

Dans la transformation des données, nous prenons

$$\tau = \frac{1}{\lambda_1+\lambda_2+c_1\mu_1+c_2\mu_2}.$$

En utilisant les spécifications ci-dessus, le schéma d'itération par valeur devient assez simple pour le problème d'allocation.

Notez que les expressions pour les temps de transition en une étape $\tau_s(a)$ et les probabilités de transition en une étape $p_{st}(a)$ ont un dénominateur commun et donc le rapport $p_{st}(a)/\tau_s(a)$ a une forme très simple.

En spécifiant le schéma d'itération des valeurs $(**)$, nous distinguons les états auxiliaires $(i_1,i_2,0)$ et les autres états.

Dans les états $(i_1,i_2,0)$, la seule décision possible est de laisser le système tel qu'il est (autrement dit, pas d'arrivée, pas de décision).

Donc

$V_n(i_1,i_2,0) = \tau\lambda_1 V_{n-1}(i_1,i_2,1)+\tau\lambda_2 V_{n-1}(i_1,i_2,2)+\tau i_1\mu_1 V_{n-1}(i_1-1,i_2,0)+$
$\tau i_2\mu_2 V_{n-1}(i_1,i_2-1,0) + \{1-\tau v(i_1,i_2)\}V_{n-1}(i_1,i_2,0)$,
où $V_{n-1}(i_1,i_2,1) = 0$ lorsque $i_1 < 0$ ou $i_2 < 0$.

Pour les états $s = (i_1,i_2,1)$,

$$\begin{aligned} V_n(i_1,i_2,1) = min\,[&v(i_1,i_2) + \tau\lambda_1 V_{n-1}(i_1,i_2,1) + \tau\lambda_2 V_{n-1}(i_1,i_2,2) \\ &+ \tau i_1\mu_1 V_{n-1}(i_1-1,i_2,0) + \tau i_2\mu_2 V_{n-1}(i_1,i_2-1,0) \\ &+\{1-\tau v(i_1,i_2)\}V_{n-1}(i_1,i_2,1), \\ &\tau\lambda_1 V_{n-1}(i_1+1,i_2,1) + \tau\lambda_2 V_{n-1}(i_1+1,i_2,2) \\ &+ \tau(i_1+1)\mu_1 V_{n-1}(i_1,i_2,0) + \tau i_2\mu_2 V_{n-1}(i_1+1,i_2-1,0) \\ &+\{1-\tau v(i_1+1,i_2)\}V_{n-1}(i_1,i_2,1)]\,, \end{aligned}$$

Pour les états $s = (i_1,i_2,2)$,

$$\begin{aligned} V_n(i_1,i_2,2) = min\,[&v(i_1,i_2) + \tau\lambda_1 V_{n-1}(i_1,i_2,1) + \tau\lambda_2 V_{n-1}(i_1,i_2,2) \\ &+ \tau i_1\mu_1 V_{n-1}(i_1-1,i_2,0) + \tau i_2\mu_2 V_{n-1}(i_1,i_2-1,0) \\ &+\{1-\tau v(i_1,i_2)\}V_{n-1}(i_1,i_2,2), \\ &\tau\lambda_1 V_{n-1}(i_1,i_2+1,1) + \tau\lambda_2 V_{n-1}(i_1,i_2+1,2) \\ &+ \tau i_1\mu_1 V_{n-1}(i_1-1,i_2+1,0) + \tau(i_2+1)\mu_2 V_{n-1}(i_1,i_2,0) \\ &+\{1-\tau v(i_1,i_2+1)\}V_{n-1}(i_1,i_2,2)]\,, \end{aligned}$$

à condition de mettre $V_{n-1}(i_1,i_2,1) = V_{n-1}(i_1,i_2,2) = \infty$ quand $i_1+i_2 = c+1$ pour évite la décision irréalisable $a = 1$ dans les états $(i_1,i_2,1)$ et $(i_1,i_2,2)$ avec $i_1+i_2 = c$.

L'algorithme d'itération par valeur pour la formulation de décision semi-markovienne avec des instants de décision fictives nécessite les états supplémentaires $(i_1,i_2,0)$.

3.3 Résolution avec l'algorithme d'itération par valeur :

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

Etape 0: Choisissez $V_0(i_1,i_2,k)$ tel que $0 \le V_0(i_1,i_2,k) \le min_a\{c_i(a)/\tau_i(a)\}$
$\forall i \in I$, $i_1+i_2 \le c$, $k = 0,1,2$.

Choisissez un nombre τ avec $0 < \tau \le min_{i,a}\tau_i(a)$.

Dans la transformation des données, nous prenons $\tau = \frac{1}{\lambda_1+\lambda_2+c_1\mu_1+c_2\mu_2}$
donc on prend $c_1 = 10$ et $c_2 = 0$ alors $\tau = 0.008547009$

Soit $n := 1$.

Etape 1: Calculez la fonction $V_n(i_1, i_2, k)$ $\forall i \in I$, $i_1 + i_2 \leq c$, $k = 0, 1, 2$, à partir de

$V_n(i_1, i_2, 0) = \tau\lambda_1 V_{n-1}(i_1, i_2, 1) + \tau\lambda_2 V_{n-1}(i_1, i_2, 2) + \tau i_1\mu_1 V_{n-1}(i_1 - 1, i_2, 0) + \tau i_2\mu_2 V_{n-1}(i_1, i_2 - 1, 0) + \{1 - \tau v(i_1, i_2)\}V_{n-1}(i_1, i_2, 0)$,

où $V_{n-1}(i_1, i_2, 1) = 0$ lorsque $i_1 < 0$ ou $i_2 < 0$.

pour les états $(i_1, i_2, 0)$ avec $i_1 + i_2 \leq c$, $k = 0, 1, 2$

$$\begin{aligned} V_n(i_1, i_2, 1) = min\,[&v(i_1, i_2) + \tau\lambda_1 V_{n-1}(i_1, i_2, 1) + \tau\lambda_2 V_{n-1}(i_1, i_2, 2) \\ &+ \tau i_1\mu_1 V_{n-1}(i_1 - 1, i_2, 0) + \tau i_2\mu_2 V_{n-1}(i_1, i_2 - 1, 0) \\ &+ \{1 - \tau v(i_1, i_2)\}V_{n-1}(i_1, i_2, 1), \\ &\tau\lambda_1 V_{n-1}(i_1 + 1, i_2, 1) + \tau\lambda_2 V_{n-1}(i_1 + 1, i_2, 2) \\ &+ \tau(i_1 + 1)\mu_1 V_{n-1}(i_1, i_2, 0) + \tau i_2\mu_2 V_{n-1}(i_1 + 1, i_2 - 1, 0) \\ &+ \{1 - \tau v(i_1 + 1, i_2)\}V_{n-1}(i_1, i_2, 1)] \ , \end{aligned}$$

Pour les états $s = (i_1, i_2, 1)$ avec $i_1 + i_2 \leq c$, $k = 0, 1, 2$

$$\begin{aligned} V_n(i_1, i_2, 2) = min\,[&v(i_1, i_2) + \tau\lambda_1 V_{n-1}(i_1, i_2, 1) + \tau\lambda_2 V_{n-1}(i_1, i_2, 2) \\ &+ \tau i_1\mu_1 V_{n-1}(i_1 - 1, i_2, 0) + \tau i_2\mu_2 V_{n-1}(i_1, i_2 - 1, 0) \\ &+ \{1 - \tau v(i_1, i_2)\}V_{n-1}(i_1, i_2, 2), \\ &\tau\lambda_1 V_{n-1}(i_1, i_2 + 1, 1) + \tau\lambda_2 V_{n-1}(i_1, i_2 + 1, 2) \\ &+ \tau i_1\mu_1 V_{n-1}(i_1 - 1, i_2 + 1, 0) + \tau(i_2 + 1)\mu_2 V_{n-1}(i_1, i_2, 0) \\ &+ \{1 - \tau v(i_1, i_2 + 1)\}V_{n-1}(i_1, i_2, 2)] \ , \end{aligned}$$

Pour les états $s = (i_1, i_2, 2)$ avec $i_1 + i_2 \leq c$, $k = 0, 1, 2$,

on va mettre $V_{n-1}(i_1, i_2, 1) = V_{n-1}(i_1, i_2, 2) = \infty$ quand $i_1 + i_2 = c + 1$ pour évite la décision irréalisable $a = 1$ dans les états $(i_1, i_2, 1)$ et $(i_1, i_2, 2)$ avec $i_1 + i_2 = c$.

Etape 2: Calculez les limites

$m_n = min_{i\in I}\{V_n(i_1, i_2, k) - V_{n-1}(i_1, i_2, k)\}$, $M_n = max_{i\in I}\{V_n(i_1, i_2, k) - V_{n-1}(i_1, i_2, k)\}$.

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

Sinon, passez à l'étape 3.

Étape 3. $n := n + 1$ et passez à l'étape 1.

3.3.1 Résultat numérique :

On considèreles données numériques suivantes :

C=10, λ_1=10, λ_2=7, μ_1=10, μ_2=1.

Les taux d' arrivée : λ_1=10 et λ_2=7.

Les taux de transmission des messages : μ_1=10 et μ_2=1.

L' algorithme d'itération par valeur est lancé avec $V_0(i_1,i_2,k)= 0$ pour tous les états avec $i_1+i_2 \leq c$, k=0,1,2 et $V_0(i_1,i_2,k)=\infty$ pour $i_1+i_2 = c+1$.

On choisit le nombre de tolérance $\varepsilon = 0.001$ pour le critère d'arrêt.

Une fois les valeurs calculées, elles seront stockées dans 3 matrices triangulaire de dimension (c*c).

Dans les matrices les Lignes i_1 et les colonnes i_2.

Etat $(i_1, i_2, 0)$:

$$V_n(i_1,i_2,0) = \begin{pmatrix} V_n(0,0,0) & . & . & . & . & V_n(0,c,0) \\ . & . & . & . & V_n(1,c-1,0) & \\ . & . & . & . & & \\ . & . & . & . & & \\ . & . & . & & & \\ . & V_n(c-1,1,0) & & & & \\ V_n(c,0,0) & & & & & \end{pmatrix}$$

Etat $(i_1, i_2, 1)$:

Un message de type 1 arrive et les messages i_1 de type 1 et i_2 de type 2 sont en cours de transmission.

$$V_n(i_1,i_2,1) = \begin{pmatrix} V_n(0,0,1) & . & . & . & . & V_n(0,c,1) \\ . & . & . & . & V_n(1,c-1,1) & \\ . & . & . & . & & \\ . & . & . & . & & \\ . & . & . & & & \\ . & V_n(c-1,1,1) & & & & \\ V_n(c,0,1) & & & & & \end{pmatrix}$$

Etat $(i_1, i_2, 2)$:

Un message de type 2 arrive et les messages i_1 de type 1 et i_2 de type 2 sont en cours de transmission.

$$V_n(i_1,i_2,2) = \begin{pmatrix} V_n(0,0,2) & . & . & . & . & V_n(0,c,2) \\ . & . & . & . & V_n(1,c-1,2) & \\ . & . & . & . & & \\ . & . & . & . & & \\ . & . & . & & & \\ . & V_n(c-1,1,2) & & & & \\ V_n(c,0,2) & & & & & \end{pmatrix}$$

Etape 0 : On choisit $V_0(i_1, i_2, k) = 0$ pour tous (i_1, i_2, k) :

```
l'affichage de V(i1,i2,0)
0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0
0    0    0    0    0    0    0
0    0    0    0    0    0
0    0    0    0    0
0    0    0    0
0    0    0
0    0
0
l'affichage de V(i1,i2,1)
0    0    0    0    0    0    0    0    0    0    0    100000
0    0    0    0    0    0    0    0    0    0    100000
0    0    0    0    0    0    0    0    0    100000
0    0    0    0    0    0    0    0    100000
0    0    0    0    0    0    0    100000
0    0    0    0    0    0    100000
0    0    0    0    0    100000
0    0    0    0    100000
0    0    0    100000
0    0    100000
0    100000
100000
l'affichage de V(i1,i2,2)
0    0    0    0    0    0    0    0    0    0    0    100000
0    0    0    0    0    0    0    0    0    0    100000
0    0    0    0    0    0    0    0    0    100000
0    0    0    0    0    0    0    0    100000
0    0    0    0    0    0    0    100000
0    0    0    0    0    0    100000
0    0    0    0    0    100000
0    0    0    0    100000
0    0    0    100000
0    0    100000
0    100000
100000
```

Figure 0 : l'affichage de $V(i_1, i_2, k)$ pour n=0

1ercas voici les paramètres choisis :
C=10,λ_1=10,λ_2=7,μ_1=10,μ_2=1.

Soit n=1 :
Etape 1 : $V(i_1, i_2, k)$=

```
l'affichage de V(i1,i2,0)
0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0    0
0    0    0    0    0    0    0    0
0    0    0    0    0    0    0
0    0    0    0    0    0
0    0    0    0    0
0    0    0    0
0    0    0
0    0
0
l'affichage de V(i1,i2,1)
0    0    0    0    0    0    0    0    0    0    27    100000
0    0    0    0    0    0    0    0    0    36    100000
0    0    0    0    0    0    0    0    45    100000
0    0    0    0    0    0    0    54    100000
0    0    0    0    0    0    63    100000
0    0    0    0    0    72    100000
0    0    0    0    81    100000
0    0    0    90    100000
0    0    99    100000
0    108    100000
117    100000
100000
l'affichage de V(i1,i2,2)
0    0    0    0    0    0    0    0    0    0    27    100000
0    0    0    0    0    0    0    0    0    36    100000
0    0    0    0    0    0    0    0    45    100000
0    0    0    0    0    0    0    54    100000
0    0    0    0    0    0    63    100000
0    0    0    0    0    72    100000
0    0    0    0    81    100000
0    0    0    90    100000
0    0    99    100000
0    108    100000
117    100000
100000
mn0=0    mn1=0    mn2=0    le min mn=0
Mn0=0    Mn1=117    Mn2=117    le max Mn=117
```

Figure 1 : l'affichage de $V(i_1, i_2, k)$ pour n=1

Etape 2 : Le critère d'arrêt $0 \leq (M_n - m_n) \leq \varepsilon m_n$ n'est pas vérifié car $(117 \not\leq 0)$ donc on passe l'étape suivante.

Etape 3 : n=2 (n:=n+1).
Pour n=2 :

```
l'affichage de V(i1,i2,0)
0    0    0    0    0    0    0    0    0    0    3.92308
0    0    0    0    0    0    0    0    0    5.23077
0    0    0    0    0    0    0    0    6.53846
0    0    0    0    0    0    0    7.84615
0    0    0    0    0    0    9.15385
0    0    0    0    0    10.4615
0    0    0    0    11.7692
0    0    0    13.0769
0    0    14.3846
0    15.6923
17
l'affichage de V(i1,i2,1)
0    0    0    0    0    0    0    0    0    5.23077    51.6923    100000
0    0    0    0    0    0    0    0    6.53846    66.1538    100000
0    0    0    0    0    0    0    7.84615    79.2308    100000
0    0    0    0    0    0    9.15385    90.9231    100000
0    0    0    0    0    10.4615    101.231    100000
0    0    0    0    11.7692    110.154    100000
0    0    0    13.0769    117.692    100000
0    0    14.3846    123.846    100000
0    15.6923    128.615    100000
17    132    100000
134    100000
100000
l'affichage de V(i1,i2,2)
0    0    0    0    0    0    0    0    0    3.92308    51.6923    100000
0    0    0    0    0    0    0    0    5.23077    66.1538    100000
0    0    0    0    0    0    0    6.53846    79.2308    100000
0    0    0    0    0    0    7.84615    90.9231    100000
0    0    0    0    0    9.15385    101.231    100000
0    0    0    0    10.4615    110.154    100000
0    0    0    11.7692    117.692    100000
0    0    13.0769    123.846    100000
0    14.3846    128.615    100000
15.6923    132    100000
134    100000
100000
mn0=0    mn1=0    mn2=0    le min mn=0
Mn0=17    Mn1=38.2308    Mn2=38.2308    le max Mn=38.2308
```

Figure 2 : l'affichage de $V(i_1, i_2, k)$ pour n=2

Le critère d'arrêt $0 \leq (M_n - m_n) \leq \varepsilon m_n$ n'est pas vérifié car $(38.2308 \not\leq 0)$ donc on passe l'étape suivante .

Pour n=3 (on fait le même travail pour les autres itérations jusqu'a obtenir une politique optimale R(n)).

L'optimum est atteint Pour n=599,

voici les valeurs :

```
l'affichage de V(i1,i2,0)
754.263 793.733 827.355 866.233 911.843 956.127 1031.54 1111 1207.66 1326.07 1504.28
754.298 793.812 827.535 866.643 912.766 958.156 1035.85 1119.64 1223.78 1371.41
754.337 793.905 827.756 867.174 914.04 971.175 1042.8 1134.72 1261.98
754.383 794.019 828.047 867.929 915.016 976.324 1056 1167.99
754.44 794.172 828.477 869.172 919.642 985.827 1085.72
754.518 794.412 829.246 871.699 927.98 1013.88
754.651 794.885 831.01 878.346 952.725
754.946 796.132 836.356 901.035
755.843 800.485 857.216
759.439 819.729
787.263
l'affichage de V(i1,i2,1)
754.298 793.812 827.535 866.643 912.766 958.156 1035.85 1119.64 1223.78 1371.41 1521.
754.337 793.905 827.756 867.174 914.04 971.175 1042.8 1134.72 1261.98 1488.41 100000
754.383 794.019 828.047 867.929 915.016 976.324 1056 1167.99 1378.98 100000
754.44 794.172 828.477 869.172 919.642 985.827 1085.72 1284.99 100000
754.518 794.412 829.246 871.699 927.98 1013.88 1202.72 100000
754.651 794.885 831.01 878.346 952.725 1130.88 100000
754.946 796.132 836.356 901.035 1059.72 100000
755.843 800.485 857.216 1018.03 100000
759.439 819.729 974.216 100000
787.263 936.729 100000
904.263 100000
100000
l'affichage  de V(i1,i2,2)
793.733 827.355 866.233 911.843 965.127 1031.54 1111 1207.66 1324.66 1443.07 1621.28
793.812 827.535 866.643 912.766 968.156 1035.85 1119.64 1223.78 1340.78 1488.41 10000
793.905 827.756 867.174 914.04 971.175 1042.8 1134.72 1251.72 1378.98 100000
794.019 828.047 867.929 916.016 975.324 1056 1167.99 1284.99 100000
794.172 828.477 869.172 919.642 985.827 1085.72 1202.72 100000
794.412 829.246 871.699 927.98 1013.88 1130.88 100000
794.885 831.01 878.346 952.725 1069.72 100000
796.132 836.356 901.035 1018.03 100000
800.485 857.216 974.216 100000
819.729 936.729 100000
904.263 100000
100000
mn0=1.76611    mn1=1.76611    mn2=1.76611    le min mn=1.76611
Mn0=1.76782    Mn1=1.76782    Mn2=1.76782    le max Mn=1.76782
```

Figure 3 : l'affichage de $V(i_1, i_2, k)$ pour n=599

Le critère d'arrêt $0 \leq (M_n - m_n) \leq \varepsilon m_n$ est vérifiée après 599 itérations car (0.00170898≤0.00176611) avec (ε=0.001).

Dans cet exemple l'algorithme converge apres 599 itérations avec un taux de perte moyen minimum compris entre (mn,Mn) = (1.76611 , 1.76782).

L'algorithme d'itération par valeur est arrêté avec la politique stationnaire R(599).

voici la politique optomale :

```
l'affichage de la politique optimal pour k=1
1    1    1    1    1    1    1    1    1    1    0
1    1    1    1    1    1    1    1    1    0
1    1    1    1    1    1    1    1    0
1    1    1    1    1    1    1    0
1    1    1    1    1    1    0
1    1    1    1    1    0
1    1    1    1    0
1    1    1    0
1    1    0
1    0
0
l'affichage de la politique optimal pour k=2
1    1    1    1    1    1    1    1    0    0    0
1    1    1    1    1    1    1    1    0    0
1    1    1    1    1    1    1    0    0
1    1    1    1    1    1    1    0
1    1    1    1    1    1    0
1    1    1    1    1    0
1    1    1    1    0
1    1    1    0
1    1    0
1    0
0
```

Figure 4 : l'affichage affichage de la politique optimale

Cette politique optimale représente la décision qu'on doit prendre chaque fois qu'un client arrive. Autrement dit, pour tous les états (i_1, i_2, k) avec

$i_1+i_2 \prec c$, k=1,2 quel est le bon choix, accepter ou refuser les arrivées de type 1 ou 2.

Interprétation des résultats :

Des études numériques indiquent que la règle de contrôle optimale globale a une structure intuitivement attrayante .elle est caractérisée par des entiers $L0, L1, ..., Lc-1$, avec $L0 \geq L1 \geq \cdots \geq Lc-1$.

Un message de type 1 (appelez le type prioritaire) est toujours accepté tant que il y'a au moins un canal libre.

Un message arrivant de type 2 (non prioritaire) qui trouve i messages de type 1 (prioritaire) présents à l'arrivée n'est accepté que lorsque moins de Li messages de type 2 (non prioritaire) sont présents dans le systéme.

les valeurs Li optimales sont :

$L0 = L1 = 8$, $L2 = L3 = 7$, $L4 = 6$, $L5 = 5$, $L6 = 4$, $L7 = 3$, $L8 = 2$, $L9 = 1$.

Par exemple:

$L0 = L1 = 8$, veut dire:

Si un message de type 2 arrive et trouve 0 ou 1 messages de type 1 dans le systéme, n'est accepté que si moins de 8 messages de type 2 sont presents dans le système.

$L4 = 6$, veut dire:

Si un message de type 2 arrive et trouve 4 messages de type 1 dans le systéme, n'est accepté que si moins de 6 messages de type 2 sont presents dans le systéme.

$L8 = 2$, veut dire:

Si un message de type 2 arrive et trouve 8 messages de type 1 dans le systéme, n'est accepté que si moins de 2 messages de type 2 sont presents dans le systéme.

2 éme cas voici les paramètres choisis :

C=15,λ_1=10,λ_2=7,μ_1=10,μ_2=1.

L'optimum est atteint Pour n=1083,

Le critère d'arrêt $0 \leq (M_n - m_n) \leq \varepsilon m_n$ est vérifié après 1083 itérations car (0.000137329$\leq$0.000152359) avec (ε=0.001).

Dans cet exemple l'algorithme converge apres 1083 itérations avec un taux de perte moyen minimum compris entre (mn,Mn)= (0.152359,0.152496).

remarque:

Quand on augmente le nombre de canaux le nombre d'itération augmante et le taux de perte moyen minimum diminue.

L'algorithme d'itération par valeur est arrêté avec la politique stationnaire R(1083).

voici la politique optomale :

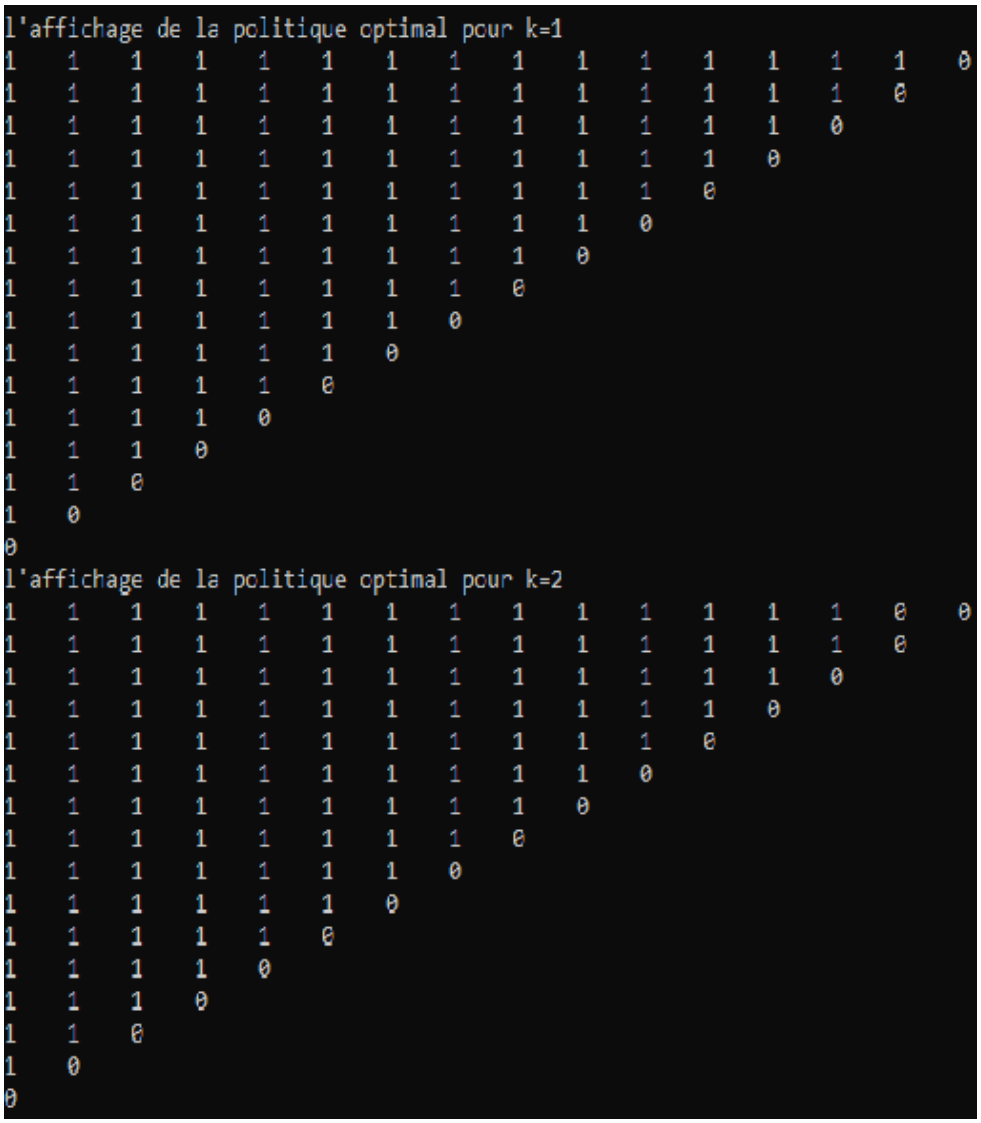

Figure 5 : l'affichage affichage de la politique optimale

Interprétation des résultats :

les valeurs Li optimales sont :
$L0 = L1 = 14$, $L2 = 13$, $L3 = 12$, $L4 = 11$, $L5 = 10$, $L6 = 9$, $L7 = 8$, $L8 = 7$, $L9 = 6$, $L10 = 5$, $L11 = 4$, $L12 = 3$, $L13 = 2$, $L14 = 1$.

3 éme cas voici les paramètres choisis :

C=10,λ_1=10,λ_2=10,μ_1=1,μ_2=1.

L'optimum est atteint Pour n=201,

Le critère d'arrêt $0 \leq (M_n - m_n) \leq \varepsilon m_n$ est vérifié après 201 itérations car (0.0102539≤0.010749) avec (ε=0.001).

Dans cet exemple l'algorithme converge apres 201 itérations avec un taux de perte moyen minimum compris entre (mn,Mn) = (10.749 , 10.7593).

L'algorithme d'itération par valeur est arrêté avec la politique stationnaire R(201).

voici la politique optomale :

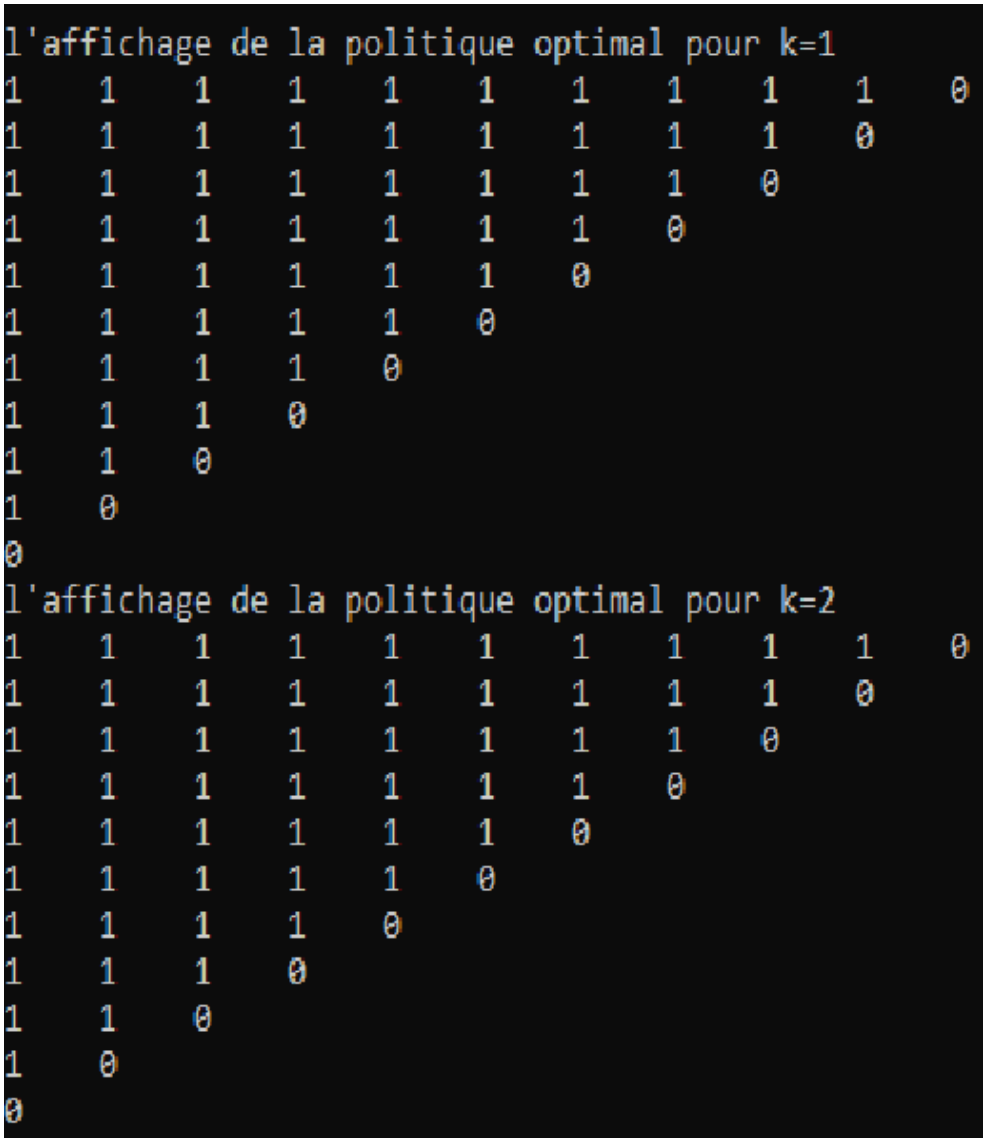

Figure 6 : l'affichage affichage de la politique optimale

Interprétation des résultats :

Dans ce cas, nous avons un seul type de message (λ_1=λ_2=**10,**μ_1=μ_2=**1).**

Donc pas de message prioritaire, un message qui arrive est toujours accepté tant qu'il y'a au moins un canal libre.

4 éme cas voici les paramètres choisis :
C=10,λ_1=9,λ_2=8,μ_1=5,μ_2=1.
L'optimum est atteint Pour n=487,

Le critère d'arrêt $0 \leq (M_n - m_n) \leq \varepsilon m_n$ est vérifié après 487 itérations car (0.00305176≤0.0031084).

Dans cet exemple l'algorithme converge apres 487 itérations avec un taux de perte moyen minimum compris entre (mn,Mn) = (3.1084 , 3.11145).

L'algorithme d'itération par valeur est arrêté avec la politique stationnaire R(487).

voici la politique optomale :

```
l'affichage de la politique optimal pour k=1
1    1    1    1    1    1    1    1    1    1    0
1    1    1    1    1    1    1    1    1    0
1    1    1    1    1    1    1    1    0
1    1    1    1    1    1    1    0
1    1    1    1    1    1    0
1    1    1    1    1    0
1    1    1    1    0
1    1    1    0
1    1    0
1    0
0
l'affichage de la politique optimal pour k=2
1    1    1    1    1    1    1    1    0    0    0
1    1    1    1    1    1    1    1    0    0
1    1    1    1    1    1    1    0    0
1    1    1    1    1    1    0    0
1    1    1    1    1    0    0
1    1    1    1    1    0
1    1    1    1    0
1    1    1    0
1    1    0
1    0
0
```

Figure 7 : l'affichage affichage de la politique optimale

Interprétation des résultats :

$L0 = L1 = 8$, $L2 = 7$, $L3 = 6$, $L4 = L5 = 5$, $L6 = 4$, $L7 = 3$, $L8 = 2$, $L9 = 1$.

L'algorithme d'itération par valeur converge apres 599, 1083, 201 et 487 itérations avec limites respectives (mn,Mn)=(1.76611 , 1.76782) , (0.152359 , 0.152496) ,(10.749 , 10.7593) et (3.1084 , 3.11145) et un taux de perte moyen minimum compris entre (mn,Mn)=(1.76611 , 1.76782) , (0.152359 , 0.152496) ,(10.749 , 10.7593) et (3.1084 , 3.11145).

Conclusion générale :

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

Il y'a beaucoup de méthodes utilisées pour la recherche d'une politique optimale « value- iteration algorithm, Policy-iteration algorithm ... », Parmi ces méthodes nous avons choisi l'algorithme d'itération par valeur « value- iteration algorithm ».

A travers le problème que nous avons présenté « Allocation optimale des serveurs aux clients concurrents », nous avons répondu à la question suivante :

Comment trouver une règle de contrôle qui minimise le taux de perte moyen ou de manière équivalente maximise le débit moyen des messages acceptés ?

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 établit par [1], nous avons réussi à proposer une solution par « value iteration algorithm », en pratique cet algorithme nécessite un grand nombre d'itérations pour converger.

Nous avons traité quatre exemples et nous avons remarqué :

Si on augmente le nombre de canaux (c) le nombre d'itérations augmente et le taux de perte moyen diminue.

Si on augmente les taux d'arrivées (λ_i) le nombre d'itérations diminue et le taux de perte moyen augmente.

Si on augmente les taux de service (μ_i) le nombre d'itérations augmente et le taux de perte moyen diminue.

Le cas ou λ_i et μ_i sont égaux l'algorithme ne s'arrête pas car il n'éxiste pas un taux de perte, le messege qui arrive se traite directement.

Le cas ou $\lambda_1=\lambda_2$ et $\mu_1=\mu_2$ comme si nous avons un seul type de message.

Heureusement pour nous, l'objectif de recherche de politique optimale d'allocation optimale des serveurs aux clients concurrents a été atteint après un nombre fini d'itérations par l'algorithme d'itération par valeur « value iteration algorithm ».

Annexe

Annexe01 :

Hypothèse de la chaine unique (unichain assumption)

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

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

Annexe02:

programmation

```
#include <iostream>

using namespace std;

int main()

{

int i1,i2,s,l1,l2,u1,u2,n;

 float
v00[100][100],v01[100][100],v02[100][100],v10[100][100],v11[100][100],v12[100][100],
vd0 [100][100],vd2 [100][100]  ;

float vp1[100][100],vp2[100][100],vd1[100][100],vi1i2[100][100];

float a,i,b,c,d,t,mn,Mn,m,mn0,mn1,mn2,Mn0,Mn1,Mn2,e=0.001;

cout<<"donner le nombre de canaux c"<<endl;

cin>>s;
```

```
cout<<"donner le taux d'arrive l1 , l2 et donner le taux de servise u1 et u2 "<<endl;
cin>>l1>>l2>>u1>>u2;
t=0.008547009;
// n3mro les vi11 et tout
for(i1=0;i1<=s;i1++){
for(i2=0;i2<=s;i2++)
if(i1+i2<=s+1){
vi1i2[i1][i2]=l1+l2+i1*u1+i2*u2;}
//affichage les vi11
}
cout<<endl;
cout<<endl;
cout<<"la matrice vi1i2 "<<endl<<endl;
cout<<endl;
cout<<endl;
for(i1=0;i1<=s;i1++){
for(i2=0;i2<=s;i2++)
if(i1+i2<=s){
cout<<vi1i2[i1][i2]<<" ";}
cout<<endl;}
cout<<endl;
cout<<"le nombre d'etration n=0";
//calcule les v00
for(i1=0;i1<=s+1;i1++){
for(i2=0;i2<=s;i2++){
if(i1+i2<=s){
```

```
v00[i1][i2]=0;}
}
for(i2=0;i2<=s+1;i2++){
if(i1+i2<=s){
v01[i1][i2]=0;}
else {if((i1+i2)==(s+1)){
v01[i1][i2]=100000;}}
}
for(i2=0;i2<=s+1;i2++){
if(i1+i2<=s){
v02[i1][i2]=0;}
else {if((i1+i2)==(s+1)){
v02[i1][i2]=100000;}}
}
}
//affichage les v00
cout<<endl;
cout<<endl;
cout<<"l'affichage de V(i1,i2,0)"<<endl;
for(i1=0;i1<=s;i1++){
for(i2=0;i2<=s;i2++){
if(i1+i2<=s){
cout<<v00[i1][i2]<<" ";
}}cout<<endl;}
cout<<"l'affichage de V(i1,i2,1)"<<endl;
for(i1=0;i1<=s+1;i1++){
```

```
for(i2=0;i2<=s+1;i2++){
if(i1+i2<=s+1){
cout<<v01[i1][i2]<<" ";
}}cout<<endl;}
cout<<"l'affichage de V(i1,i2,2)"<<endl;
for(i1=0;i1<=s+1;i1++){
for(i2=0;i2<=s+1;i2++){
if(i1+i2<=s+1){
cout<<v02[i1][i2]<<" ";
}}cout<<endl;}
cout<<endl;
cout<<endl;

n=0;
m=1;
i=0;
while(i<m)
{
n=n+1;
cout<<endl;
cout<<endl;
cout<<"l'iteration numero n="<<n<<" ";
// calcule les matrice v
for(i1=0;i1<=s+1;i1++){
// pour aucune arrive
for(i2=0;i2<=s;i2++){
```

```
if(i1+i2<=s){
v10[i1][i2]=t*l1*v01[i1][i2]+t*l2*v02[i1][i2]+t*i1*u1*v00[i1-1][i2]+t*i2*u2*v00[i1][i2-1]+(1-t*vi1i2[i1][i2])*v00[i1][i2];}}
// pour arrive de type 1
for(i2=0;i2<=s+1;i2++){
if(i1+i2<=s){
a=vi1i2[i1][i2]+t*l1*v01[i1][i2]+t*l2*v02[i1][i2]+t*i1*u1*v00[i1-1][i2]+t*i2*u2*v00[i1][i2-1]+(1-t*vi1i2[i1][i2])*v01[i1][i2];
b=t*l1*v01[i1+1][i2]+t*l2*v02[i1+1][i2]+t*(i1+1)*u1*v00[i1][i2]+t*i2*u2*v00[i1+1][i2-1]+(1-t*vi1i2[i1+1][i2])*v01[i1][i2];
{
if(a>b){
v11[i1][i2]=b;
vp1[i1][i2]=1;}
else{
v11[i1][i2]=a;
vp1[i1][i2]=0;}}}
else {if(i1+i2<=s+1){
v11[i1][i2]=100000;}}}
// pour arrive de type 2
for(i2=0;i2<=s+1;i2++){
if(i1+i2<=s){
c=vi1i2[i1][i2]+t*l1*v01[i1][i2]+t*l2*v02[i1][i2]+t*i1*u1*v00[i1-1][i2]+t*i2*u2*v00[i1][i2-1]+(1-t*vi1i2[i1][i2])*v02[i1][i2];
d=t*l1*v01[i1][i2+1]+t*l2*v02[i1][i2+1]+t*i1*u1*v00[i1-1][i2+1]+t*(i2+1)*u2*v00[i1][i2]+(1-t*vi1i2[i1][i2+1])*v02[i1][i2];
{
if(c>d){
```

```
v12[i1][i2]=d;
vp2[i1][i2]=1;}
else{
v12[i1][i2]=c;
vp2[i1][i2]=0;}}}
else {if(i1+i2<=s+1){
v12[i1][i2]=100000;}}} }
//affichage les v11
cout<<endl;
cout<<endl;
cout<<"la matrice v"<<n<<endl<<endl;
cout<<"l'affichage de V(i1,i2,0)"<<endl;
for(i1=0;i1<=s;i1++){
for(i2=0;i2<=s;i2++){
if(i1+i2<=s){
cout<<v10[i1][i2]<<" ";
}}cout<<endl;}
cout<<"l'affichage de V(i1,i2,1)"<<endl;
for(i1=0;i1<=s+1;i1++){
for(i2=0;i2<=s+1;i2++){
if(i1+i2<=s+1){
cout<<v11[i1][i2]<<" ";
}}cout<<endl;}
cout<<"l'affichage de V(i1,i2,2)"<<endl;
for(i1=0;i1<=s+1;i1++){
for(i2=0;i2<=s+1;i2++){
```

```
if(i1+i2<=s+1){
cout<<v12[i1][i2]<<" ";
}}cout<<endl;}
cout<<endl;
cout<<endl;
//affichage les vp1
cout<<"la matrice de politique optimal vp1"<<endl<<endl;
for(i1=0;i1<=s;i1++){
for(i2=0;i2<=s;i2++){
if(i1+i2<=s){
cout<<vp1[i1][i2]<<" ";
}}cout<<endl;}
cout<<endl;
cout<<endl;
//affichage les vp2
cout<<"la matrice de politique optimal vp2"<<endl<<endl;
for(i1=0;i1<=s;i1++){
for(i2=0;i2<=s;i2++){
if(i1+i2<=s){
cout<<vp2[i1][i2]<<" ";
}}cout<<endl;}
cout<<endl;
cout<<endl;
//calcule les vd
for(i1=0;i1<=s;i1++){
for(i2=0;i2<=s;i2++){
```

```
if(i1+i2<=s){
vd0[i1][i2]=v10[i1][i2]-v00[i1][i2];}
}
for(i2=0;i2<=s;i2++){
if(i1+i2<=s){
vd1[i1][i2]=v11[i1][i2]-v01[i1][i2];}
}
for(i2=0;i2<=s;i2++){
if(i1+i2<=s){
vd2[i1][i2]=v12[i1][i2]-v02[i1][i2];}}
}
//affichage les vd
cout<<"la matrice vd"<<n<<endl<<endl;
for(i1=0;i1<=s;i1++){
for(i2=0;i2<=s;i2++){
if(i1+i2<=s){
cout<<vd0[i1][i2]<<" ";
}}cout<<endl;}
cout<<endl;
cout<<endl;
for(i1=0;i1<=s;i1++){
for(i2=0;i2<=s;i2++){
if(i1+i2<=s){
cout<<vd1[i1][i2]<<" ";
}}cout<<endl;}
cout<<endl;
```

```
   cout<<endl;
   for(i1=0;i1<=s;i1++){
   for(i2=0;i2<=s;i2++){
   if(i1+i2<=s){
   cout<<vd2[i1][i2]<<" ";
   }}cout<<endl;}
   cout<<endl;
   cout<<endl;
   //nhsbo el min w el max
   Mn0=vd0[0][0];
   mn0=vd0[0][0];
   Mn1=vd1[0][0];
   mn1=vd1[0][0];
   Mn2=vd2[0][0];
   mn2=vd2[0][0];
   //cout<<"mn0="<<mn0<<" "<<"mn1="<<mn1<<" "<<"mn2="<<mn2<<" "<<"le min
mn="<<mn;
   //cout<<endl;
   //cout<<"Mn0="<<Mn0<<" "<<"Mn1="<<Mn1<<" "<<"Mn2="<<Mn2<<" "<<"le max
Mn="<<Mn;
   for(i1=0;i1<=s;i1++){
   for(i2=0;i2<=s;i2++){
   if(i1+i2<=s){
   if(vd0[i1][i2]<mn0)
   {mn0=vd0[i1][i2];}
   if(Mn0<vd0[i1][i2])
   {Mn0=vd0[i1][i2];}}}
```

```
   for(i2=0;i2<=s;i2++){
   if(i1+i2<=s){
   if(vd1[i1][i2]<mn1)



   {mn1=vd1[i1][i2];}
   if(Mn1<vd1[i1][i2])
   {Mn1=vd1[i1][i2];}}}
   for(i2=0;i2<=s;i2++){
   if(i1+i2<=s){
   if(vd2[i1][i2]<mn2)
   {mn2=vd2[i1][i2];}
   if(Mn2<vd2[i1][i2])
   {Mn2=vd2[i1][i2];}}}}
   if((mn0<=mn1)&&(mn0<=mn2))
   {mn=mn0;}
   else{if((mn1<=mn0)&&(mn1<=mn2))
   {mn=mn1;}
   else{if((mn2<=mn0)&&(mn2<=mn1))
   {mn=mn2;}}}
   cout<<endl;
   cout<<"mn0="<<mn0<<" "<<"mn1="<<mn1<<" "<<"mn2="<<mn2<<" "<<"le min
mn="<<mn;
   if((Mn2<=Mn0)&&(Mn1<=Mn0))
   {Mn=Mn0;}
   else{if((Mn2<=Mn1)&&(Mn0<=Mn1))
```

```
{Mn=Mn1;}
else{if((Mn0<=Mn2)&&(Mn1<=Mn2))
{Mn=Mn2;}}}
cout<<endl;
cout<<"Mn0="<<Mn0<<" "<<"Mn1="<<Mn1<<" "<<"Mn2="<<Mn2<<" "<<"le max Mn="<<Mn;
cout<<endl;
m=Mn-mn;
i=e*mn;
cout<<endl;
cout<<"m="<<m<<" "<<"i="<<i;
//nredj3o les v1 f el v0
for(i1=0;i1<=s;i1++){
for(i2=0;i2<=s;i2++){
if(i1+i2<=s){
v00[i1][i2]=v10[i1][i2];}
}
for(i2=0;i2<=s;i2++){
if(i1+i2<=s+1){
v01[i1][i2]=v11[i1][i2];}
}
for(i2=0;i2<=s;i2++){
if(i1+i2<=s+1){
v02[i1][i2]=v12[i1][i2];}}}
}}
```

Bibliographies

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

[2] Boumerzoug Loubna et Serir Imen ; contrôle optimal d'un système de service stochastique à s serveurs par les processus de décision semi markovien ,université blida 1 (2018-2019) .

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

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

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

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

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