Nouvelle approche pour le contrôle d'admission d'appels dans un réseau cellulaire

Sidi-Mohammed Senouci — Guy Pujolle

Laboratoire Informatique de Paris 6 (LIP6)
Université de Paris VI
8, rue du Capitaine Scott
75015 Paris – France
{Sidi-Mohammed.Senouci, Guy.Pujolle}@lip6.fr

RÉSUMÉ. *Nous considérons, dans ce papier, le problème du contrôle d'admission d'appels (CAC) dans un réseau cellulaire supportant plusieurs classes de trafic. Ce problème peut être formulé comme un processus de décision semi-markovien (SMDP). Mais, vu la complexité à trouver une solution exacte pour résoudre ce problème, nous avons utilisé une approche basée sur la programmation neuro-dynamique (NDP). Cette approche, appelé aussi apprentissage par renforcement (RL), permet de construire une politique d'admission d'appels robuste et assez dynamique. Nous démontrons que les politiques obtenues en utilisant les deux implémentations de l'algorithme par renforcement, TRL-CAC et NRL-CAC, fournissent de meilleurs résultats par rapport aux solutions classiques. Nous démontrons aussi, par un ensemble de simulations, la robustesse de ces politiques qui améliorent considérablement la qualité de service et réduisent les probabilités de blocage des appels de handoff et ceci malgré les variations des conditions de trafic.*

ABSTRACT. *In this paper, the call admission control (CAC) problem in a multimedia cellular network that handles several classes of traffic with different resource requirements is addressed. The problem is formulated as a Semi-Markov Decision Process (SMDP) problem. It is too complex to allow for an exact solution for this problem, so, we use a real-time neuro-dynamic programming (NDP) [Reinforcement Learning (RL)] algorithm to construct a dynamic call admission control policy. It is shown that the policies obtained using our algorithms TQ-CAC and NQ-CAC, which are two implementations of the RL algorithm, provide a good solution and are able to earn significantly higher revenues than classical solutions. A broad set of experiments shows the robustness of our policies which improves the Quality of Service (QoS) and reduces call blocking probabilities for handoff calls in spite of variations in the traffic conditions.*

MOTS-CLÉS : *Réseaux cellulaires, CAC, Apprentissage par renforcement.*

KEYWORDS: *Cellular Networks, CAC, Reinforcement Learning.*

1. Introduction

Les demandes en transmissions sans fils fournissant des communications fiables de voix et de données ont crû massivement ces dernières années. Contrairement aux réseaux filaires, plusieurs problèmes tels que le contrôle d'admission d'appels, l'allocation des ressources, la gestion de la localisation et le routage sont plus difficiles à résoudre et doivent leur complexité aux imperfections du support sans fil.

La zone géographique desservie par un réseau mobile est divisée en petites surfaces appelées cellules. Chacune d'elles est couverte par un émetteur appelé "station de base" dont lui est assigné un ensemble de canaux [1]. Un canal est alloué à chaque usager, mais quand celui ci passe d'une cellule à une autre il doit demander un nouveau canal dans la cellule destination. Cet événement, appelé transfert inter cellulaire ou handoff, doit être transparent à l'utilisateur. Si la cellule destination n'a aucun canal disponible, l'appel est coupé. L'un des soucis majeurs lors de la conception des réseaux mobiles est la réduction de cette probabilité de coupure. En effet, du point de vue de l'usager, elle est beaucoup plus désagréable qu'un échec de connexion. Ceci est d'autant plus important qu'afin de répondre à la croissance des réseaux mobiles, la taille des cellules est de plus en plus réduite, ce qui augmente considérablement le nombre de transferts inter-cellulaires. Dans (Ramjee et al.,1996 ; Yoon et al., 1993), les auteurs démontrent que les techniques de réservation de canaux (souvent appelées Guard Channel Schemes) permettent de réduire la probabilité de coupure de communication en réservant simplement, dans chaque cellule, des canaux à l'usage exclusif des handoffs. Si ces techniques sont simples à dimensionner dans un cadre de trafic mono-classe, elles deviennent beaucoup plus compliquées à optimiser dans un contexte multi-classes. Dans un tel contexte, il est parfois préférable de bloquer un appel d'une classe moins prioritaire et de recevoir un autre appel d'une classe plus prioritaire. En plus, ces techniques ignorent complètement l'expérience et la connaissance qui pourraient être acquises pendant l'exécution en temps réel du système. Dans ce nouveau contexte, l'utilisation de techniques d'apprentissage (Barto et al., 1995 ; Mitchell, 1997) permet d'aboutir à de très bons résultats dans des périodes assez raisonnables.

Ce papier propose une nouvelle approche pour résoudre le problème de contrôle d'admission des appels (CAC) dans les réseaux cellulaires supportant plusieurs classes de trafic. La politique (ou stratégie) optimale de CAC est obtenue en utilisant un algorithme d'apprentissage par renforcement connu sous le nom de Q-learning (Watkins et al., 1992). Le double objectif de ce type d'apprentissage est de conduire de manière optimale un système au cours du temps et à apprendre cette conduite optimale à travers un ensemble d'expériences. Cette politique réduit les probabilités de blocage des appels de handoff et est capable de produire de plus grands profits.

---

1. Les canaux peuvent être des fréquences radio, des intervalles de temps (time slots) ou des codes, ceci dépend de la technique d'accès utilisée.

Nous considérons un système avec deux classes de trafic. Un revenu, représentant le coût du service fournit aux utilisateurs, est associé à chaque classe de trafic. Ces revenus dépendent également du fait que c'est un nouvel appel ou bien un appel de handoff. Les appels de handoff sont plus prioritaires et donc les revenus associés sont plus grands. L'objectif est donc, de recevoir des clients et d'en rejeter d'autres dans le but de maximiser le cumul des revenus reçus au cours du temps. Ce problème peut être formulé comme un processus de décision semi-markovien SMDP dont l'apprentissage est la solution idéale.

Le reste du papier est organisé comme suit. Après une courte description de l'algorithme du Q-Learning ainsi qu'une formulation du problème de CAC comme un SMDP (section 2), nous détaillons les deux différentes implémentations de l'algorithme du Q-Learning (TRL-CAC et NRL-CAC) capables de résoudre ce SMDP dans la section 3. L'évaluation des performances et les résultats de simulations sont présentés dans la section 4. Finalement, la section 5 récapitule les principales contributions de ce travail et conclut le papier.

2. Description du Problème

Nous proposons une nouvelle approche pour résoudre le problème de contrôle d'admission des appels CAC dans un réseau cellulaire. L'approche est basée sur le fait que le CAC peut être vue comme un SMDP, et que l'apprentissage est une des voies la plus intéressante pour résoudre ce genre de problèmes. Nous avons utilisé, plus spécialement, l'apprentissage par renforcement (RL), appelé aussi programmation neuro-dynamique (NDP). Dans l'apprentissage par renforcement, et comme le montre la figure 1, un agent apprend une politique d'admission en interagissant directement avec son environnement au travers d'expériences itérées, ceci, dans le but d'améliorer ses performances. Il existe une variété d'algorithmes d'apprentissage par renforcement. Un algorithme particulier, appelé Q-learning (Watkins et al., 1992), apparaît comme étant le plus approprié pour le problème du contrôle d'admission CAC. Dans ce qui suit, nous allons décrire brièvement cet algorithme, puis détailler la manière de l'utiliser pour résoudre le problème CAC.

2.1. L'algorithme du Q-Learning

Supposons que l'"*apprenti*" agent existe dans un environnement représenté par un ensemble d'états fini $S = \{s_1, s_2, ..., s_n\}$ et qu'il peut exécuter une des actions contenue dans l'ensemble fini $A = \{a_1, a_2, ..., a_n\}$. L'interaction entre l'agent et l'environnement consiste, à chaque instant, en les séquences suivantes :

– L'agent observe l'état actuel de l'environnement $s_t \in S$;

– En se basant sur l'état s_t, l'agent prend une décision en exécutant une action $a_t \in A$;

– L'environnement fait alors une transition vers un nouvel état $s_{t+1} = s' \in S$ suivant la probabilité $P_{ss'}(a_t)$;

– L'agent reçoit instantanément un certain revenu $r_t = r(s_t, a_t)$ indiquant la conséquence de cette décision.

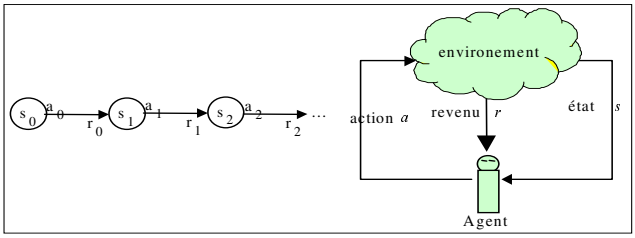


Figure 1. *L'interaction Agent-environnement*

L'objectif de l'agent est d'apprendre une politique $\pi : S \to A$, permettant de choisir la prochaine action $a_t = \pi(s_t)$ à effectuer et ceci en fonction de l'état actuel s_t.

Pour chaque politique π, on associe une valeur $Q^{\pi}(s,a)$, qu'on appellera sa Q-valeur et qui représente la moyenne des gains prévus si l'action *a* est exécutée lorsque l'état courant du système est *s* et qu'ensuite π est adoptée comme politique d'admission. La politique optimale $\pi^*(s)$, est la politique qui maximise le cumul des revenus $r_t = r(s_t, a_t)$ reçus après un temps infini. Le but de l'algorithme du Q-learning est de rechercher une approximation pour $Q^*(s,a) = Q^{\pi^*}(s,a)$ de manière récursive avec seulement (s_t, a_t, s_t', r_t) comme information disponible. Cette information comprend l'état à l'instant t, s_t , l'état à l'instant t+1, $s_t'=s_{t+1}$, l'action prise quand le système est à l'état s_t, a_t, et le revenu, r_t, reçu à l'instant *t* suite à l'exécution de cette action.

Les Q-valeurs sont mises à jours, de façon récursive à chaque transition, en utilisant la formule suivante (Mitchell, 1997 ; Watkins et al., 1992 ; Nie et al., 1999):

$$Q_{t+1}(s,a) = \begin{cases} Q_t(s,a) + \alpha_t \Delta Q_t(s,a), & si\ s = s_t\ et\ a = a_t \\ Q_t(s,a), & \text{sinon} \end{cases} \quad (1)$$

Où $$\Delta Q_t(s,a) = \left\{ r_t + \gamma \max_b \left[Q_t(s'_t, b) \right] \right\} - Q_t(s,a) \quad (2)$$

Le facteur $\gamma \in [0,1]$ est une constante de propagation temporelle et $\alpha_t = \frac{1}{1 + visit_t(s_t, a_t)}$ est le taux d'apprentissage, où $visit_t(s_t, a_t)$ est le nombre de fois où l'action a_t a été choisie lorsque le système est dans l'état s_t.

Dans (Watkins et al., 1992), l'auteur démontre que si chaque paire *(s,a)* est infiniment visitée, et que le pas d'apprentissage α tend vers zéro, $Q_t(x,a)$ converge sûrement (avec une probabilité égale à 1) vers $Q^*(x, a)$, lorsque $t \to \infty$. La politique optimale sera, alors, celle avec la plus grande Q-valeur : $\pi^*(s) = \arg\max_{a \in A(s)} Q^*(s,a)$.

2.2. Formulation SMDP

Dans cette section, nous allons formuler le problème de CAC dans un réseau cellulaire sous forme d'un processus de décision semi markovien (SMDP). Nous considérons un système FCA (Fixed Channel Allocation) avec un nombre de canaux disponibles, N fixe, dans chaque cellule. Mais ceci peut facilement être étendu à un système DCA (Dynamic Channel Allocation) et fera l'objet de travaux futurs.

Concentrons-nous sur une cellule donnée et considérons deux [2] classes de trafic C1 et C2. Les appels de la première classe demandent seulement un canal, contrairement aux appels de la deuxième classe qui quant à eux demandent deux canaux. Nous prenons en considération, les nouveaux appels qui peuvent surgire dans la cellule ainsi que les appels de handoff venant des cellules voisines.

Ce système cellulaire peut être considéré comme un système à événement discret. Les principales événements pouvant se produire dans une cellule incluent les arrivées et les départs des appels. Ces événements sont modélisés par des variables stochastiques avec des distributions appropriées. En particulier, les arrivées de nouveaux appels dans la cellule suivent une loi de Poisson. Nous supposons également, que les arrivées des appels de handoff sont Poissonniennes et que le temps de séjour de chaque appel est exponentiellement distribué. Les appels arrivent, donc, dans la cellule puis partent soit en effectuant un handoff vers une autre cellule soit en se terminant normalement. Le réseau devra, alors, choisir d'accepter ou de rejeter ces demandes de connexion. En retour, il collecte l'ensemble des revenus reçus de la part des clients acceptés (les gains) ainsi que l'ensemble des revenus reçus de la part des clients rejetés (les pertes). L'objectif de l'opérateur du réseau est de trouver une politique de CAC qui maximise l'ensemble des gains à long terme et qui réduit les probabilités de blocage des appels de handoff ainsi que l'ensemble des pertes. Nous avons utilisé, pour notre simulation, les paramètres expérimentaux données par les tableaux 1 et 3.

L'ensemble des états "s", des actions "a" et des revenus "r" a été défini comme suit :

1) **Etats :** A l'instant t, le système est dans une configuration particulière, x, défini par le nombre d'appels en cours pour chaque type de trafic. A un instant aléatoire, un événement e peut se produire indiquant ou bien l'arrivée d'un appel (nouveau ou handoff) ou bien le départ d'un appel. L'évènement de départ est dû à une terminaison normale de l'appel ou bien à un transfert inter-cellulaire de l'appel vers une cellule voisine. La configuration x et l'événement e déterminent, ensemble, l'état du système $s=(x, e)$:

– $x=(x_1, x_2)$, avec x_1 et x_2 représentant le nombre d'appels de chacune des classes de trafic ($C1$ et $C2$ respectivement) dans la cellule.

– $e=\{1,2,3,4\}$ où

2. L'idée peut facilement être étendue à plusieurs classes de trafic.

$$e = \begin{cases} 1 & \text{arrivée d'un nouvel appel de type } C1 \\ 2 & \text{arrivée d'un nouvel appel de type } C2 \\ 3 & \text{arrivée d'un appel de handoff de type } C1 \\ 4 & \text{arrivée d'un appel de handoff de type } C2 \end{cases}$$

Nous ne tenons pas compte des états associés à un départ d'un appel. La raison de cette simplification est que le départ d'un appel n'est pas un point de décision dans le contrôle d'admission, et donc aucune mesure ne doit être prise.

2) **Actions :** Appliquer une action c'est accepter ou bien rejeter l'appel. Ainsi, l'ensemble des actions possibles est A={1,0} où

$$a_i = \begin{cases} 1 & \text{accépter} \\ 0 & \text{rejeter} \end{cases}$$

3) **Revenus :** Le revenu *r(s,a)* représente le gain instantané encouru par l'exécution de l'action *a* quand le système est dans l'état *s*. Quand un appel de classe *i* arrive ($e = e_i \in \{1,2,3,4\}$), le revenu associé sera nulle si l'appel est rejeté (*a=0*) et égale à η_i si l'appel est accepté (*a=1*).

$$r(s,a) = \begin{cases} \eta_i & si\ a = 1\ et\ e = e_i \\ 0 & sinon \end{cases}$$

Les valeurs de η_i sont données par le tableau 1. Vu que l'objectif principal de l'opérateur du réseau est de diminuer les probabilités de blocage des appels de handoff, des valeurs de revenus assez grandes leurs ont été affectées. Nous supposons aussi que les appels de la classe de trafic C1 sont plus prioritaires que ceux de la classe C2.

η_1	η_2	η_3	η_4
5	1	50	10

Tableau 1. *Revenus Instantanés.*

En résumé, nous avons choisi la description de l'ensemble des états comme étant $s = ((x_1, x_2), e)$, où x_i est le nombre d'appels en cours de la classe C_i, et $e \in \{1,2,3,4\}$ représente l'arrivée d'un nouvel appel ou d'un appel de handoff dans la cellule pour les deux classes de trafic considérées *C1* et *C2*. Quand un événement se produit, l'agent doit choisir une des actions possibles *A(s)*={0=*rejeter*, 1=*accepter*}. Lors de la terminaison d'un appel, aucune mesure ne doit être prise. L'agent doit donc déterminer une politique d'acceptation des appels permettant de maximiser le cumul des gains reçus et ceci connaissant uniquement l'état du réseau *s*. Ce système constitue un SMDP avec comme espace d'états un ensemble fini $S = \{(x, e)\}$ et un espace d'actions possibles, l'ensemble fini $A = \{0,1\}$.

3. Implémentation des algorithmes

Après avoir formulé le problème CAC sous forme d'un SMDP, nous allons décrire deux implémentations de l'algorithme Q-learning capables de résoudre ce

problème. Nous les avons nommé TRL-CAC (Table Reinforcement Learning CAC) et NRL-CAC (Neural network Reinforcement Learning CAC). Le premier algorithme TRL-CAC utilise une simple table pour représenter la fonction *Q* (l'ensemble de tous les Q-valeurs). En revanche, le second algorithme, NRL-CAC, utilise un réseau de neurones multicouches (Mitchell, 1997).

3.1 Représentation des Q-valeurs

Pour représenter et sauvegarder les Q-valeurs, nous utilisons deux différentes approches : une simple table et un réseau de neurone. Les différences entre ces deux approches sont exprimées en terme de complexité de calcul et de besoins de stockage en mémoires. Ces différences sont résumées dans le tableau 2 ci-dessous.

	Nombre d'opérations de calcul nécessaires	**Unités mémoire nécessaires**
Table	1	$676 \times 5 = 3380$
Réseau de neurones	$(4+4)\times 10+(1+1)\times 10=100$	$4 \times 10 + 10 = 50$

Tableau 2. *Table vs. Réseaux de neurones.*

1.1.1 Table

L'approche utilisant une table (voir figure 2 (a)), est la méthode la plus simple et la plus efficace. Elle a l'avantage que le calcul y est exact et qu'elle est complètement conforme aux hypothèses de structures faites afin de prouver la convergence de l'algorithme du Q-learning (Watkins et al., 1992).

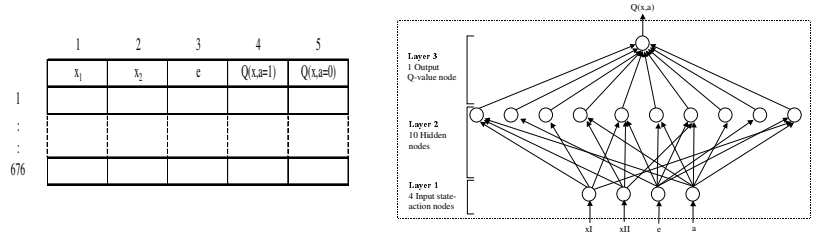
Dans le cas de deux classes de trafic seulement, et comme nous pouvons le constater dans le tableau 2, une seule opération de comparaison entre les deux Q-valeurs d'acceptation et de rejet est nécessaire. Mais pour obtenir ces deux valeurs, c'est une affaire d'indexation qui peut être fastidieuse. En effet, la taille de la table utilisée pour stocker les Q-valeurs est très grande ($676 \times 5 = 3380$ d'unités mémoires sont nécessaires dans notre cas).

Ainsi, lorsque l'ensemble des paires état-action, *(s,a)*, est large ou lorsque les variables d'entrée, *(x,e)*, constituant l'état *s* sont des variables continues, l'utilisation d'une simple table devient prohibitive à cause des immenses besoins de stockage. Dans ce cas, quelques fonctions d'approximation, telles que les réseaux de neurones ou bien les arbres de régression (Mitchell, 1997), peuvent être utilisées de manière efficace.

1.1.2 Réseau de Neurones

Le réseau de neurone multi-couches (figure 2 (b)), constitué de 4 entrées, 10 unités cachés et une unité de sortie, a été utilisé pour représenter les Q-valeurs.

La complexité de calcul dépend de la taille du réseau de neurone. Dans notre cas, et comme le montre le tableau 2, $8 \times 10 + 2 \times 10 = 100$ opérations (multiplications et additions) sont nécessaires pour calculer les deux Q-valeurs d'acceptation et de rejet. Mais en terme de besoins de stockage, l'approche nécessite moins de mémoires de stockage (50 unités seulement) pour sauvegarder les poids du réseau de neurones.



(a) (b)

Figure 2. *Représentation des Q-valeurs (a) Table, (b) Réseau de Neurones.*

3.2. Implémentation

L'organigramme de la figure 3 récapitule les différentes phases des algorithmes TRL-CAC et NRL-CAC. Nous avons initialisé les Q-valeurs à zéro pour TRL-CAC et à des valeurs aléatoires pour NRL-CAC. Nous rappelons que les états les plus intéressants sont ceux associés aux arrivées des appels et donc, nous ne tenons pas compte des états associés aux départs des appels. Ceci réduit considérablement les capacités de calcul et de stockage des Q-valeurs.

Lorsqu'un appel arrive (nouveau ou handoff), l'algorithme détermine si la qualité de service n'est pas violée en acceptant cet appel. Si c'est le cas, l'appel est rejeté ; sinon l'action est choisie selon la formule :

$$a = \arg\max_{a \in A(s)} Q^*(s,a) \qquad (3)$$

où $A(s)$={1=*accepter*, 0=*rejeter*}.

En particulier, (3) implique les procédures suivantes. Quand un appel arrive, la Q-valeur d'acceptation ainsi que la Q-valeur de rejet de l'appel sont déterminées soit à partir de la table (TRL-CAC) soit à partir du réseau de neurones (NRL-CAC). Si le rejet a une plus grande valeur, l'appel est alors rejeté. Dans le cas contraire, si l'acceptation a la plus grande valeur, l'appel est accepté.

Dans ces deux cas, et pour apprendre les Q-valeurs optimales $Q^*(s,a)$, la fonction est mise à jour à chaque transition du système d'un état s vers un état s'. Pour les deux algorithmes, ceci se fait de la façon suivante :

1. TRL-CAC: C'est la formule *(1)* qui est utilisée pour mettre à jour la Q-valeur appropriée dans la table ;

2. NRL-CAC: Quand un réseau de neurone est utilisé pour stocker la fonction Q, une seconde procédure d'apprentissage est nécessaire pour apprendre les poids du

réseau de neurones. Cette procédure utilise l'algorithme de *rétro-propagation* (BP – Back Propagation) (Mitchell, 1997). Dans ce cas, ΔQ définie par la formule *(2)* est employée comme signal d'erreur qui est rétro-propagé dans les différentes couches du réseau de neurones.

Nous comparons nos politiques à la politique que nous avons appelé la politique *greedy* (Tong et al., 2000) (politique qui accepte un nouvel appel ou un appel de handoff si la contrainte de capacité n'est pas violée en acceptant cet appel).

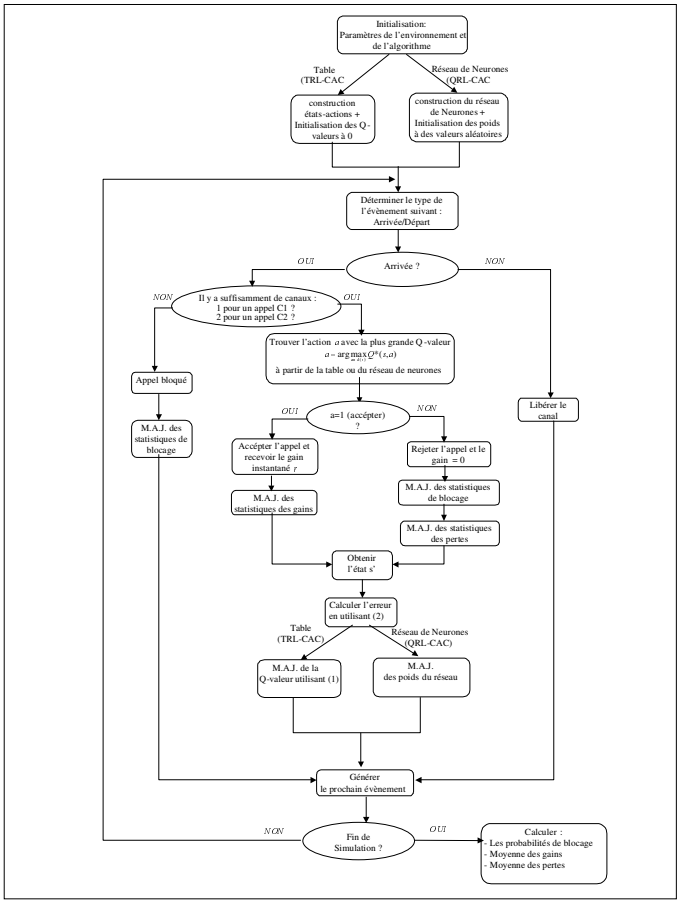


Figure 3. *Les algorithmes TRL-CAC et NRL-CAC.*

3.3. Exploration

Pour une exécution correcte et efficace de l'algorithme du Q-learning, toutes les paires potentiellement importantes des état-action *(s,a)* doivent être explorées. A la base, la convergence du Q-learning exige que toutes ces paires soient infiniment

visitées. Pour cela, pendant une période d'apprentissage assez longue, l'action est choisie non selon (3), mais selon la formule (4) suivante avec une probabilité d'exploration ε :

$$a = \arg \min_{a \in A(s)} visites\,(s, a) \qquad (4)$$

Où $visites(s,a)$ désigne le nombre de fois où la configuration *(s,a)* a été visitée. Cette heuristique, appelée $\varepsilon - directed$, accélère significativement la convergence des Q-valeurs. Ces valeurs sont, dans un premier temps, calculées en utilisant cette heuristique avec les paramètres indiqués dans le tableau 3 pendant une période de temps suffisamment longue. Dans un second temps, ces valeurs seront employées pour initialiser les Q-valeurs dans nos deux algorithmes de CAC.

4. Simulation

Afin d'évaluer les avantages de nos algorithmes, nous avons simulé un réseau cellulaire en utilisant une simulation à événement discret. Comme indiqué auparavant, nous avons considéré un système à 24 canaux fixes par cellule (FCA). Les performances des algorithmes ont été évaluées en terme de revenus reçus de la part des clients acceptés (les gains), de revenus reçus de la part des clients rejetés (les pertes), ainsi que de probabilités de blocage des appels de handoff. La constante de propagation temporelle a été choisie égale à $\gamma = 0.5$ et la probabilité d'exploration $\varepsilon = 1$. Les principales procédures impliquées dans la simulation sont récapitulées dans figure 3.

Un ensemble de simulations a été effectué, incluant le cas d'une charge de trafic constante pour toutes les classes de trafic, le cas de charges de trafic variables, et le cas d'une charge de trafic variable dans le temps. Les résultats expérimentaux sont présentés dans les figures 4 à 8. Ces résultats démontrent bien que l'apprentissage par renforcement est une très bonne solution pour le problème de contrôle d'admission CAC. En effet, dans tous les cas étudiés les probabilités de blocage des appels de handoff sont considérablement réduites et l'ensemble des gains reçus dû à l'acceptation de clients est également amélioré.

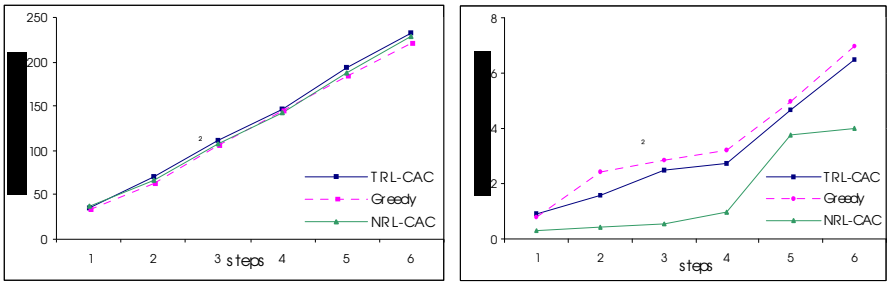
4.1. Charge de trafic constante

La première expérience considère une charge de trafic constante pour les deux classes *C1* et *C2*. Les paramètres de simulation utilisés sont les mêmes utilisées pendant la période d'apprentissage et sont récapitulées dans les tableaux 1 et 3. Nous supposons que la durée d'un appel est exponentiellement distribuée de paramètre μ_C ($1/\mu_C$ = 120s). La période de séjour d'un mobile dans une cellule est également exponentiellement distribuée de paramètre μ_H ($1/\mu_H$ = 60s). En conséquence, le temps de séjour d'un appel est exponentiel et de paramètre $\mu=\mu_C + \mu_H$ ($1/\mu$ = 40s).

Paramètres	Type de l'appel	
	C1	C2
Nombre de canaux	1	2
Temps de séjour	40 s	40 s
Débit des arrivées	λ_1 = 180 *appels / heure*	$\lambda_2 = \lambda_1 / 2$ = 90 *appels / heure*

Tableau 3. *Paramètres Expérimentaux.*

La figure 4(a) représente les gains reçus en suivant les politiques générées par les deux algorithmes TRL-CAC et NRL-CAC. Ils sont calculés toutes les dix minutes durant une heure de simulation. Ces gains, dues à l'acceptation de nouveaux appels ou d'appels de handoff des deux classes de trafic (*C1* ou *C2*) dans la cellule, ne sont pas très importants comparés à ceux de la politique *greedy*. La figure 4(b), quant à elle, démontre que les revenus dus au rejet de nouveaux appels ou à l'échec d'un transfert inter-cellulaire (les pertes) ont été amplement réduits.



(a) (b)

Figure 4. *Total par heure* (a) des g*ains* (b) *des pertes.*

4.2. Charges de trafic variables

Dans ce deuxième cas, nous avons utilisé les mêmes politiques que le cas précédent (charge de trafic constante) mais avec six différentes charges de trafic (pour les deux classes C1 et C2). Les différentes charges sont récapitulées dans le tableau 4 ci-dessous.

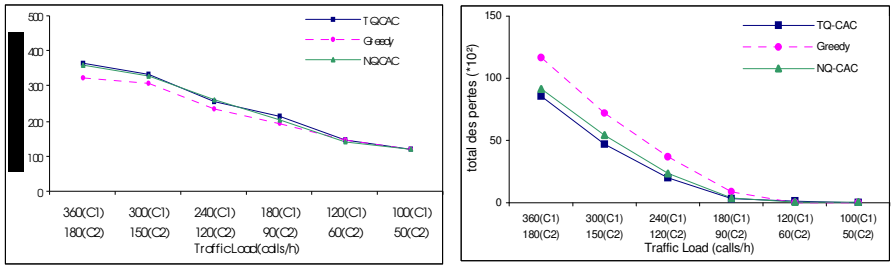
	Type de l'appel	
	C1	C2
Charges de trafic (appels/heure)	360	180
	300	150
	240	120
	180	90
	120	60
	90	45

Tableau 4. *Paramètres Expérimentaux.*

Nous pouvons également remarquer sur les figures 5 et 6 que les algorithmes proposés donnent de meilleurs résultats par rapport à d'autres heuristiques et ceci

pour toutes les charges de trafic considérées dans le tableau 4 et plus particulièrement quand la charge de trafic est grande. Nous pouvons noter que quand la charge de trafic est petite, les avantages de TRL-CAC et NRL-CAC deviennent négligeables. Ceci peut facilement être expliqué par le fait que lorsque la charge de trafic est petite, il y a assez de canaux pour tous les appels et ainsi tous les appels sont acceptés. Ceci explique pourquoi, dans la figure 6, la probabilité de blocage est presque égale à 0 quand la charge de trafic est fixée à *120 appels/h* pour C1 et à 60 *appels/h* pour C2.

Nous remarquons également que les résultats du premier algorithme TRL-CAC sont plus performants que ceux de NRL-CAC. Ceci ci s'explique par le fait que NRL-CAC emploie une fonction d'approximation (un réseau de neurones) pour représenter les Q-valeurs et donc a besoin de plus de temps pour converger. Les différences techniques entre ces deux algorithmes, exprimées en terme de complexité de calcul et de besoins de stockage mémoires, sont discutées plus en détails dans la section 3.1 ci-dessus.



(a) (b)

Figure 5. *Total par heure* (a) des g*ains* (b) *des pertes.*

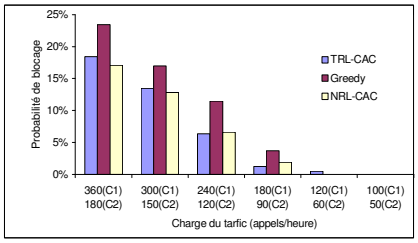


Figure 6. *Probabilités de blocage des appels de handoff pour six différentes charges de trafic.*

Tout ceci illustre, clairement, que nos deux algorithmes TRL-CAC et NRL-CAC offrent des possibilités intéressantes pour améliorer significativement la tâche de contrôle d'amission sur un large intervalle de charges de trafic. Il est intéressant d'observer que ni la table ni le réseau de neurones n'ont été réappris pour ces nouvelles charges, indiquant que le système possède des facultés de généralisation et d'adaptation.

4.3. Charge de trafic variable dans le temps

La charge de trafic dans un système cellulaire est typiquement variable dans le temps. Dans ce dernier cas, nous utilisons toujours la même politique que le premier cas (charge de trafic constante) et nous utilisons, comme dans (Nie et al., 1999), le modèle de trafic donnée dans la figure 7. Ce modèle décrit la variation des taux d'arrivées pendant une journée ouvrable. Dans une journée ouvrable, les heures de pointe se produisent à 11:00 heure du matin et à 4:00 heure de l'après midi. La figure 8 donne les résultats de simulation avec l'hypothèse que les deux classes de trafic suivent le même modèle de trafic donné dans la figure 7. La charge de trafic est fixée à un maximum de 180 appels/h pour la classe C1 et 90 appels/h pour la classe C2. Les probabilités de blocage ont été calculées sur une base de chaque heure. Les améliorations des algorithmes d'apprentissage proposés sont bien apparentes par rapport à la politique *greedy* et particulièrement pendant les pics de trafic (à 11:00 heure du matin et à 4:00 heure de l'après midi).

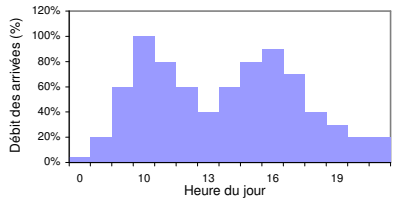


Figure 7. *Modèle de trafic d'une une journée ouvrable.*

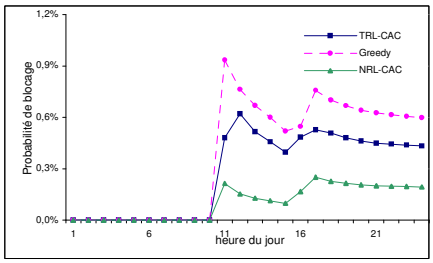


Figure 8. *Probabilité de blocage des appels de handoff avec une charge de trafic variable dans le temps.*

5. Conclusion

Dans cet article, nous avons présenté une nouvelle approche pour résoudre le problème de contrôle d'admission d'appel CAC dans un réseau cellulaire. Nous avons formulé le problème comme un processus de décision semi-markovien (SMDP), avec malheuresement un espace d'états assez grand. Les solutions traditionnelles de SMDP sont complètement inadéquates pour de tels problèmes à grande échelle. Ainsi, les solutions optimales sont obtenues en utilisant un système autonome basé sur deux différentes implémentations de l'algorithme d'apprentissage par renforcement 'Q-Learning', et que nous avons appelé TRL-CAC et NRL-CAC.

Les avantages, acquis en utilisant une telle approche, peuvent être récapitulés comme suit. Tout d'abord, l'apprentissage fournit une voie simple pour obtenir une solution optimale au processus de décision semi-markovien et dont il était difficile de trouver une solution exacte en utilisant les méthodes traditionnelles. En second lieu, les deux algorithmes proposés, TRL-CAC et NRL-CAC, sont exécutés en temps réel, et donc il est possible d'effectuer en ligne la tâche d'apprentissage tout en exécutant le contrôle d'admission. Comparé à d'autres approches, le système offre une certaine faculté de généralisation. Ainsi, n'importe quel événement imprévu, dû aux variations des conditions de trafic, peut être considéré comme une nouvelle expérience pour améliorer l'adaptation et l'apprentissage du système. Troisièmement, la politique d'acceptation peut être déterminée rapidement et avec peu d'efforts. Nous démontrons, également, que les algorithmes de CAC proposés donnent de meilleurs résultats comparés à d'autres heuristiques.

6. Bibliographie

G. Barto, S. J. Bradtke, and S. P. Songh, "Learning to act using real-time dynamic programming", *Artificial Intelligence*, vol. 72, pp. 81-138, 1995.

P. Marbach, O. Mihatsch and J. N. Tsitsikils, "Call admission control and routing in integrated services networks using neuro-dynamic programming", *IEEE Journal on Selected Areas in Communications (JSAC'2000)*, vol. 18, N°. 2, pp. 197 –208, Février 2000.

T. M. Mitchell, "*Machine Learning*", McGraw-Hill companies, Inc., 1997.

J. Nie and S. Haykin, "A Q-Learning based dynamic channel assignment technique for mobile communication systems", *IEEE Transactions on Vehicular Technology*, vol. 48, N°. 5, Séptembre 1999.

R. Ramjee, R. Nagarajan and D. Towsley, "On Optimal Call Admission Control in Cellular Networks", *IEEE INFOCOM*, pp. 43-50, San Francisco, CA, Mars 1996.

S. Senouci, A. L. Beylot, Guy Pujolle, "A dynamic Q-learning-based call admission control for multimedia cellular networks", IEEE International Conference on Mobile and Wireless Communications Networks (MWCN'2001), pp. 37-43, Recife, Brazil, Août 2001.

H. Tong and T. X. Brown, "Adaptive Call Admission Control under Quality of Service Constraint: a Reinforcement Learning Solution", *IEEE Journal on Selected Areas in Communications (JSAC'2000)*, vol. 18**,** N°. 2, pp. 209-221, Février 2000.

C. J. C. H. Watkins and P. Dayan, "Q-learning", *Machine Learning*, vol. 8, pp. 279-292, 1992.

C.H. Yoon, C.K. Un, Performance of personal portable radio telephone systems with and without guard channels, *IEEE Journal on Selected Areas in Communications (JSAC'1993)*, vol. 11, pp. 911-917, Août 1993.