Deep Q-learning pour l'ordonnancement de paquets sous contraintes strictes de latence et de taille de buffer

Sylvain NÉRONDAT[1,2] Xavier LETURC[1] Christophe J. LE MARTRET[1] Philippe CIBLAT[2]

[1]Thales, 92230 Gennevilliers , France

[2]LTCI, Telecom Paris, Institut Polytechnique de Paris, 91120 Palaiseau, France

Résumé – Nous proposons deux ordonnanceurs basés sur du deep reinforcement learning (DRL) dans le but de réduire la perte de paquets par expiration et par dépassement de capacité du buffer, l'un spécifique aux taux d'arrivée, l'autre, général, entraîné sur une pluralité de taux d'arrivée. Nous obtenons moins de pertes de paquets avec l'ordonnanceur spécifique par rapport aux heuristiques de l'état de l'art. L'ordonnanceur général obtient des performances soit équivalentes, soit meilleures par rapport aux heuristiques.

Abstract – We propose two schedulers based on DRL in order to reduce packet loss due to delay violation and buffer overflow. The first scheduler is arrival rate specific, the second generalizes regardless of arrival rates. We obtain less packet loss with the specific scheduler compared to the state-of-the-art heuristics. The general scheduler obtains either equivalent or better performance compared to the heuristics.

1 Introduction

Dans cet article, nous considérons un système de communications multi-utilisateurs dans lequel une unité centrale réalise l'ordonnancement de paquets. Les utilisateurs cherchent à transmettre des flux de paquets ayant différentes qualité de service (QoS). On associe à chaque utilisateur autant de files d'attente que de QoS différentes. L'ordonnancement consiste à sélectionner l'ordre dans lesquelles les files d'attente seront servies (c.-à-d. dont les paquets seront extraits pour être envoyés sur le canal de transmission). Nous supposons que chaque paquet possède un indicateur de durée de vie (TTL). Traditionnellement, les algorithmes d'ordonnancement de paquets sont conçus par le biais d'heuristiques, mais de nouvelles solutions basées sur le deep reinforcement learning (DRL) commencent à émerger.

Un des avantages du DRL est la capacité de ces algorithmes à prendre en compte une quantité plus importante d'information (par exemple l'ensemble des TTL des paquets des files d'attente) alors que les heuristiques sont des modèles paramétriques dépendant de quelques paramètres (au maximum sept pour celles considérées dans cet article). De plus, une heuristique vise un objectif particulier (délais, débit...) alors que le DRL permet de combiner plusieurs de ces objectifs dans la récompense. On peut donc s'attendre à ce que ce type d'approche soit plus performante que les approches traditionnelles. Notre contribution s'inscrit dans le cadre du DRL avec une architecture de type deep Q-network (DQN).

Quelle que soit l'approche (heuristique ou DRL), les algorithmes d'ordonnancement sont caractérisés par : 1) l'observation qu'ils ont du système, 2) l'objectif à optimiser. On peut noter une grande variété de ces deux caractéristiques dans les solutions de l'état de l'art, ce qui rend l'exercice d'énumération exhaustive des solutions publiées difficile. Dans ce qui suit, nous proposons une manière de classer les solutions pour en avoir une vue synthétique. Concernant l'observation du système, on peut distinguer trois catégories : 1) channel-aware (CA), où l'ordonnanceur connait des informations sur la qualité des liens à servir (de type Channel Quality Indicator (CQI)) ou leurs capacités, 2) buffer-aware (BA), où l'ordonnanceur connait le nombre de paquets dans les files d'attente, 3) delay-aware (DA), où l'ordonnanceur connait des informations sur le TTL des paquets dans les files d'attente. Dans cette dernière catégorie, on trouve en particulier : a) la connaissance du TTL le plus faible (paquet le plus urgent à transmettre) par file d'attente et que l'on notera de manière conventionnelle head-of-line (HOL), ou b) la connaissance de l'ensemble des TTL des paquets que l'on notera symboliquement $\{\text{TTL}\}$.

Pour décrire l'état de l'art des techniques DRL et positionner notre contribution, nous ajoutons deux caractéristiques supplémentaires qui précisent les conditions de gestion des paquets dans les files d'attente pour définir la récompense. Tout d'abord il y a le caractère fini (ou infini) de la taille des files d'attente. Lorsque les files d'attentes sont de taille finie, il peut arriver que le nombre de place restant dans une file d'attente soit inférieur au nombre de paquets arrivant dans cette file, de sorte que les paquets surnuméraires sont rejetés. Ce phénomène est conventionnellement connu sous le nom de buffer overflow (BO). Nous avons ensuite le fait de supprimer les paquets (retirés de la file d'attente) dont la valeur du TTL est égale à 0, ce qui est l'essence même de la notion de TTL. Mais il existe dans la littérature des solutions dans lesquelles les paquets ne sont pas effacés pour des TTL négatifs, ceci étant compensé par des pénalités plus importantes et proportionnelles au dépassement du TTL dans la récompense. Nous utiliserons l'acronyme TE (pour TTL enforcement) pour signifier que les paquets de TTL égal à 0 sont effacés.

TABLE 1 : Heuristiques de l'état de l'art.

Algorithme	CA	DA	BA	Objectif
RR	N	N	N	F
EDF	N	HOL	N	D
PF	R	N	N	F
MLWDF	R	HOL	N	OF
EXP-rule	SE	HOL	N	OF
LOG-rule	SE	HOL	N	OF

Nous avons synthétisé l'état de l'art sous forme de tableaux en distinguant les heuristiques (tableau 1) et les algorithmes DRL (tableau 2). Les acronymes utilisés dans les tableaux sont définis comme ci-après. O (le modèle est utilisé), N (la caractéristique ou le modèle n'est pas utilisé), R (débit), SE (efficacité spectrale), CQI (qualité du canal), H (réponse impulsionnelle du canal), C (capacité), F (équité), D (délais), L (perte de paquets), OF (dépassement de la taille du buffer). Par exemple, la référence DRL de la première ligne du tableau 2 s'interprète comme suit : l'algorithme connait la réponse impulsionnelle du canal, il utilise comme information sur le délais le TTL le plus petit de chaque file, et n'utilise pas d'information sur l'occupation des files d'attente. La récompense cherche à optimiser conjointement le débit, le délai et la perte de paquets. Il a été évalué dans un environnement dans lequel les paquets de TTL égal à 0 sont supprimés et la taille des files d'attente est infinie.

TABLE 2 : Algorithmes DRL de l'état de l'art.

Article	CA	DA	BA	Obj.	TE	BO
[1]	H	HOL	N	E, D, L	O	N
[2]	CQI	HOL	O	D	N	O
[3]	N	N	O	L	N	O
[4]	C	N	N	R	N	N
[5]	SE	HOL	O	D	N	N
Notre sol.	C	{TTL}	O	L	O	O

Pour les heuristiques on trouve les solutions classiques de la littérature (voir [6]) : le round Robin (RR), l'earliest deadline first (EDF), le proportionnal fair (PF), le modified largest weighted delay first (MLWDF), l'exponential rule (EXP-rule) et la log rule (LOG-rule).

Concernant le DRL, nous avons fait un recensement (non-exhaustif) de travaux sur l'ordonnancement utilisant une architecture DQN qui montre déjà la grande diversité des approches et modèles utilisés. Le tableau 2 permet aussi de situer notre approche par rapport à l'état de l'art ("Notre sol."). Cette approche est originale à deux titres et constitue le cœur des contributions de cet article. Premièrement, nous proposons une approche a priori nouvelle qui prend en compte l'ensemble des informations de TTL des paquets dans les files d'attente alors que dans la littérature les solutions utilisent au mieux la valeur minimale du TTL par file (HOL). Deuxièmement, nous considérons un modèle de file d'attente qu'on peut qualifier de *réaliste* puisqu'il considère à la fois la finitude des files d'attente (BO) et une suppression des paquets ayant un TTL égal à zéro (TE). De plus, nous comparons notre solution DQN avec les heuristiques recensées dans le tableau 1 dans le même environnement que pour la solution DQN (c.-à-d. BO + TE).

L'article est organisé comme suit : la section 2 introduit les notations et le modèle étudié. La section 3 décrit les deux approches, heuristiques et deep Q-learning (DQL), pour résoudre le problème de d'ordonnancement. La section 4 expose et commente les résultats de simulations. La section 5 est dédiée à une conclusion.

2 Formulation du problème

Nous considérons n_f files d'attente où chaque file i correspond à une QoS pour un utilisateur donné. On suppose que chaque file peut contenir au maximum B paquets. Les paquets arrivants et ne pouvant être stockés dans le buffer s'il est plein sont supprimés (on dira qu'il y a un BO). Nous définissons le slot comme l'unité de temps. À chaque slot, nous pouvons extraire des paquets d'une seule file d'attente afin de les envoyer sur un canal. Le canal associé à la file d'attente i peut envoyer $n_{c,i}$ paquets sans erreur à chaque slot. Dans cet article, on considère les hypothèses simplificatrices suivantes : i) le canal est constant au cours du temps, ii) on connait $n_{c,i}$. Ces hypothèses seront relâchées dans des travaux ultérieurs, en prenant en compte, p. ex., des métriques telles que le CQI.

Chaque paquet arrive dans une file i avec un TTL de D, qui est ensuite décrémenté de 1 à chaque slot passé dans i. Une file i peut se représenter sous la forme d'un vecteur $\mathbf{s}_i = [d_{0,i}, \cdots, d_{B-1,i}]$ tel que $d_{j,i} \in \{-1, 0, \cdots, D\}$ est la TTL du $j$ème emplacement de la $i$ème file, et où -1 désigne un emplacement vide. On peut noter que $d_{0,i}$ est le HOL de i.

On note par q_i le nombre de paquets dans la file d'attente i à un slot donné, c.-à-d. le nombre d'emplacement $d_{j,i}$ dont la valeur est supérieure à -1. Au début de chaque slot, l'algorithme d'ordonnancement de paquet sélectionnera une file d'attente. Ensuite les n_i paquets les plus anciens de cette file seront extraits et envoyés sur le canal, avec $n_i = \min(n_{c,i}, q_i)$. Après l'extraction des paquets, ceux restant dans les files d'attente avec un TTL égal à 0 sont supprimés à cause de violation de la contrainte de latence. Nous notons $n_{d,i}$ le nombre de paquets supprimés à cause de cette contrainte. Nous définissons par $n_{e,i} := n_i + n_{d,i}$ le nombre total de paquets ne faisant plus partie de la file d'attente. Les paquets arrivent dans les files d'attente à la fin de chaque slot. Les paquets dépassant la place restante $\kappa_i = B - q_i + n_{e,i}$ sont supprimés en raison d'un BO. Et ce processus recommence au slot suivant.

2.1 Espace d'état

On note l'état global $\mathbf{s}_k := [\mathbf{s}_{1,k}, \cdots, \mathbf{s}_{n_f,k}]$ avec $\mathbf{s}_{i,k}$ l'état de la file d'attente i au slot k, défini par le TTL de chaque paquet présent dans cette file d'attente. L'espace d'état $\mathcal{S}_i$ de la file d'attente i est l'ensemble des vecteurs possibles $\mathbf{s}_{i,k}$.

L'espace d'état global $\mathcal{S}$ est l'ensemble des espaces d'état de chaque file d'attente, c.-à-d. $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_{n_f}$. On peut montrer que la cardinalité de $\mathcal{S}_i$ (c.-à-d. le nombre total d'états possibles) est de $|\mathcal{S}_i| = \binom{B+D+1}{D+1}$ et donc que la cardinalité de l'espace d'état global vaut $|\mathcal{S}| = \prod_{i=1}^{n_f} |\mathcal{S}_i| = \binom{B+D+1}{D+1}^{n_f}$.

La transition d'un état $\mathbf{s}_{i,k}$ à l'état suivant $\mathbf{s}_{i,k+1}$ dépend à la fois du nombre de paquets extraits et du nombre de paquets arrivant dans la file d'attente i. Pour cela, on supposera que l'arrivée des paquets suit une distribution de Poisson de paramètre $\lambda_i \in [0, \lambda_{\max}]$ avec $i \in \{1, \cdots, n_f\}$ et on note $n_{r,i}$ le nombre de paquets arrivant dans la file d'attente i selon cette distribution. On a $\mathbf{s}_{i,k+1} = t(\mathbf{s}_{i,k}, a, n_{r,i})$ où t est la fonction de transition et où a est l'action, qui est décrite dans la section suivante. Pour simplifier les notations, nous écrivons parfois l'état global actuel $\mathbf{s} = \mathbf{s}_k$ et l'état global suivant $\mathbf{s}' = \mathbf{s}_{k+1}$ lorsqu'il n'y a pas d'ambiguïté.

2.2 Espace d'action

Nous notons $\mathcal{A} = \{1, \cdots, n_f\}$ l'espace d'actions, c.-à-d. l'ensemble des actions disponibles pour l'ordonnanceur. On note $a \in \mathcal{A}$ une action que l'ordonnanceur peut entreprendre. L'action $a = i$ correspond à la sélection de la file d'attente i et

l'envoi de $n_{a,i}$ paquets sur le canal. Ainsi, le nombre de paquets extraits pour chaque file i est $n_{a,i} = n_i \delta_{a,i}$, où $\delta_{a,i}$ est le symbole de Kronecker ($\delta_{a,i} = 1$ si $a = i$, 0 sinon).

2.3 Fonction de coût

En fonction de l'ordonnanceur implémenté, le nombre de paquets perdus en raison d'une violation de latence ou d'un BO peut varier. Cette perte de paquets représente le coût associé à l'ordonnanceur. Notre objectif est de trouver l'ordonnanceur minimisant la perte de paquets à long terme.

La QoS considérée dans cet article est la latence maximale D. Nous notons c_d le coût dû à la violation de QoS qui correspond au nombre de paquets avec un TTL nul pour le slot courant qui ne sont pas envoyés et donc supprimés. Selon le modèle du système, le coût $c_d(\mathbf{s}, a, \mathbf{s}') = \sum_{i=1}^{n_f} n_{d,i}$.

Nous notons c_o le coût dû au BO. Il s'agit du nombre de paquets reçus dépassant la place restante dans chaque file i. Selon le modèle du système, le coût $c_o(\mathbf{s}, a, \mathbf{s}') = \sum_{i=1}^{n_f} \max(0, n_{r,i} - \kappa_i)$.

Le coût c du système est l'addition des deux coûts précédents. Nous avons ainsi $c(\mathbf{s}, a, \mathbf{s}') = c_d(\mathbf{s}, a, \mathbf{s}') + c_o(\mathbf{s}, a, \mathbf{s}')$.

Nous cherchons la politique optimale μ^*, minimisant le coût à long terme pondéré (opposé de la récompense) :

$$\mu^* = \arg\min_{\mu} \bar{J}(\mu) \tag{1}$$

avec $\bar{J}(\mu) = \lim_{N\to\infty} \mathbb{E}\left[\sum_{n=0}^{N} \gamma^n c(\mathbf{s}_n, a_n, \mathbf{s}'_n)\right]$, où $\gamma \in [0, 1)$ est le facteur d'oubli.

3 Résolution du problème

Notre objectif est de résoudre le problème d'optimisation (1). Ce problème est standard dans le cadre des Processus Décisionnel de Markov. Il pourrait être résolu par des algorithmes itératifs comme la *Value Iteration*. Néanmoins cette approche ne convient pas en raison du nombre d'états dans le problème considéré. En effet, notre modèle contient $n_f = 4$ files d'attente, une contrainte de latence de $D = 7$ slots et une taille de buffer de $B = 10$, nous avons un nombre d'état supérieur à 10^{18}. Avec ce nombre d'état, il n'est pas possible de calculer les matrices de transitions nécessaires à *Value Iteration*. C'est pourquoi, nous allons le résoudre via une approche qui soutient le passage à l'échelle. L'approche retenue sera le DQL que nous présentons en Section 3.2. Cette approche sera comparée aux heuristiques les plus courantes de l'état de l'art que nous décrivons à la Section 3.1.

3.1 Approche par heuristiques existantes

Toutes les heuristiques considérées ici peuvent se mettre sous la forme $i^* = \arg\max_i f(\mathbf{x}_i)$, où f est la fonction associée à l'heuristique et $\mathbf{x}_i$ des paramètres liés à la file utilisée par la fonction f. Dans le tableau 3, nous décrivons les différentes fonctions f considérées, avec T_i le nombre de slots durant lesquels la file i n'a pas été sélectionnée pour envoyer des paquets (T_i est remis à 0 quand la file i est sélectionnée par l'ordonnanceur), $\bar{r}_i$ le nombre de paquets moyen par slot réellement émis à partir de la file i. Dans notre implémentation, le moyennage se fait depuis le début de la mise en route du système. Un moyennage sur une fenêtre glissante aurait aussi pû être mis en place. Enfin, α_i, β_i et η_i sont des constantes arbitraires positives et $\overline{\eta} = \frac{1}{n_f} \sum_{i=1}^{n_f} \eta_i (D - d_{0,i})$.

TABLE 3 : Fonctions associées aux heuristiques étudiées.

Nom	fonction f
RR	T_i
EDF	$-d_{0,i}$
PF	$\frac{n_{c,i}}{\bar{r}_i}$
MLWDF	$\alpha_i n_{c,i} (D - d_{0,i})$
Exp-rule	$\alpha_i \exp\left(\frac{\eta_i (D - d_{0,i})}{\beta_i + \sqrt{\overline{\eta}}}\right) n_{c,i}$
Log-rule	$\alpha_i \log\left(\beta_i + \eta_i (D - d_{0,i})\right) n_{c,i}$

3.2 Approche par Deep Q-learning

Dans le cadre des processus décisionnels de Markov, la fonction $Q(\mathbf{s}, a)$ correspond à l'espérance de la somme pondérée des récompenses futures en étant dans l'état $\mathbf{s}$ et en effectuant l'action a. Si le processus admet un nombre fini d'états (comme c'est le cas dans notre contexte), la politique optimale est déterministe et caractérisée par l'action maximisant la fonction Q pour un état donné. Par conséquent, l'approche dite Q-learning consiste à apprendre les valeurs de la fonction Q [7]. Lorsque le nombre d'états est trop grand, il convient d'approximer la fonction Q par un réseau de neurones profond dit DQN. On parle alors de DQL.

Le DQN considéré dans ce travail est un réseau de neurones complètement connecté prenant en entrée l'état courant $\mathbf{s}$ auquel on applique en amont une normalisation. En effet, il a été constaté que les techniques d'apprentissage fonctionnaient mieux quand les entrées du réseau avaient des valeurs pré-encadrées typiquement entre 0 et 1. Notre technique de normalisation est la suivante : l'état de chaque file ne sera pas représenté par la liste de ces TTL comme mentionné à la Section 2 mais de manière équivalente par l'histogramme normalisé de ces TTL. Ainsi le vecteur est de taille la valeur maximum du TTL plus deux, et à l'emplacement j de ce vecteur, on donne la proportion de paquets ayant $j - 1$ pour valeur de TTL. Par exemple, une file de taille 4 et de TTL maximum 3 ayant un état égal à $[0; 2; 2; -1]$ dans la représentation de la Section 2 aura un état normalisé égal à $[0,25; 0,25; 0; 0,5; 0]$. Le DQN considéré dans ce travail aura pour sortie la valeur de la fonction Q pour toutes les actions.

Lors de phase d'entraînement, nous utiliserons une approche *ϵ-greedy* où à chaque étape d'un épisode, on choisit l'action maximisant la valeur courante de la fonction Q avec une probabilité $1 - \epsilon$ et une action aléatoire avec une probabilité ϵ. Le paramètre ϵ est appelé facteur d'exploration. Au début de chaque épisode, on ré-initialise l'état courant tout en conservant la valeur de fonction Q obtenue à l'épisode précédent.

Lors de la phase d'évaluation, on applique la fonction Q, donc la politique associée obtenue à la suite de la phase d'entrainement, et on prend l'action la maximisant.

Durant la phase d'entrainement, nous avons retenu deux manières pour gérer le paramètre vectoriel $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_{n_f}]$ contenant le taux d'arrivée de paquet de chaque file : *i)* pour la première manière, nous avons entrainé un DQN par valeur de $\boldsymbol{\lambda}$. Par conséquent, nous avons obtenu un ensemble de DQN. Chaque DQN sera dit spécifique et dans la suite, nous appliquerons toujours le DQN spécifique associé à la valeur

de $\boldsymbol{\lambda}$ pour lequel il a été entraîné. *ii)* Pour la seconde manière, nous avons entraîné un unique DQN qui ne dépend pas de la valeur spécifique de $\boldsymbol{\lambda}$. Ce DQN sera dit général. Pour cela, durant la phase d'entrainement, la valeur de $\boldsymbol{\lambda}$ change aléatoirement à chaque épisode.

4 Simulations et résultats

Le DQN général a été entraîné avec des taux d'arrivée λ_i tirés aléatoirement entre 0 et 3 pour toutes les files. Pour le DQN spécifique, nous considérons que chaque file d'attente admet le même taux d'arrivée $\lambda_i = \lambda$, pour tout i. Nous avons entraîné en tout 15 DQNs spécifiques, un pour chaque taux d'arrivée de l'ensemble suivant : $\{\lambda_{\min} : p : \lambda_{\max}\}$ avec le pas $p = 0,2$, $\lambda_{\min} = 0,2$ et $\lambda_{\max} = 3$. Enfin, le vecteur du canal est le suivant $\mathbf{n}_c = [1; 2; 3; 1]$ ce qui signifie, par exemple, que la file d'attente 4 est associée à un canal acceptant la transmission d'un paquet par slot. On fixe $\gamma = 0,95$.

Chaque DQN possède deux couches cachées de 256 neurones chacune et a été entraîné sur 10.000 épisodes de 2.000 étapes. La fonction d'activation entre chaque couche cachée est une ReLU. Il n'y a pas de fonction d'activation pour la couche de sortie. Le facteur d'exploration ϵ est initialisé à 1 au début de l'entraînement et décroît ensuite d'un facteur de $0,99$ à chaque épisode jusqu'à le figer lorsqu'il atteint une valeur de $0,01$. Toutes les 50 étapes, nous actualisons le DQN en rejouant 256 expériences $(\mathbf{s}, a, \mathbf{s}', r)$ et en appliquant l'algorithme Adam où le taux d'apprentissage égal à 5.10^{-4}.

Pour l'évaluation, nous effectuons 100 épisodes indépendants de 5.000 étapes chacun par valeur de λ, correspondant au taux d'arrivée d'entraînement des DQNs spécifiques. Nous comparons les DQN général et spécifique aux heuristiques suivantes : RR, EDF, PF, LOG-rule, EXP-rule et MLWDF. Pour les trois dernières heuristiques, nous avons implémenté deux versions : la première version est celle de la littérature où les α_i dépendent du nombre moyen de paquets émis $\bar{r}_i$. Ainsi $\alpha_i = \eta_i / \bar{r}_i$ avec η_i une constante positive. Dans la seconde version (notée avec les indices 2 dans la figure 1), α_i est une constante positive égale à η_i. Pour LOG-rule, nous prenons $\beta_i = 1.1$ et $\eta_i = 1$ [8]. Pour EXP-rule et MLWDF, nous prenons $\eta_i = -\ln(10^{-2})/D$ et $\beta_i = 1$ [9]. La première version sera plus équitable que la seconde par la présence du terme $\bar{r}_i$ mais la qualité des canaux est moins mise en avant et donc elle sera moins efficace en terme de pertes de paquet.

La figure 1 représente la différence de nombre de paquets perdus entre la méthode DQN spécifique et les méthodes DQN général, RR, EDF, PF, MLWDF-1&2, EXP-1&2 et LOG-1&2 en fonction du taux d'arrivée λ. On remarque que les secondes versions de MLWDF, EXP et LOG ont de bonnes performances, proches du DQN général. Le DQN spécifique admet les meilleurs performances et l'écart est renforcé quand les taux d'arrivée sont élevés, c.-à-d. quand le système est mis sous tension. Ainsi la construction de politique par le DRL permet d'améliorer les heuristiques classiques et donc de concevoir des politiques adaptées au problème posé.

5 Conclusion

Deux ordonnanceurs s'appuyant sur le DQL pour minimiser la perte de paquets sous contraintes strictes de latence et de taille de buffer ont été mis au point. Les performances de ces deux ordonnanceurs ont été comparées à des heuristiques de l'état de l'art. Des gains importants ont été observés surtout pour des taux d'arrivée élevés. Par conséquent, ces résultats préliminaires sont prometteurs et des travaux futurs portant sur des systèmes plus complexes et menant une analyse de l'apport de la connaissance du TTL de chaque paquet sur les algorithmes proposés doivent être conduits.

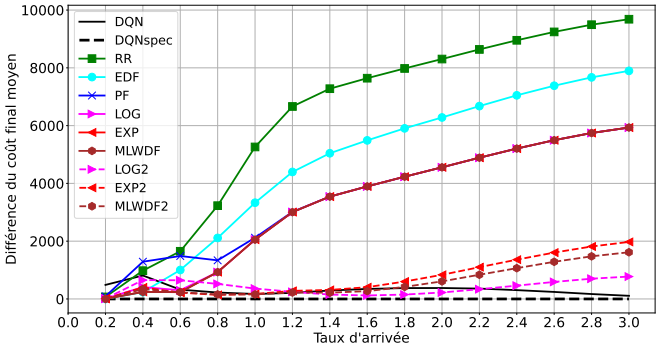

FIGURE 1 : Différence entre le coût moyen d'un algorithme et celui du DQN spécifique en fonction du taux d'arrivée.

Références

[1] X. Chen, C. Wu, T. Chen, H. Zhang, Z. Liu, Y. Zhang, and M. Bennis, "Age of information aware radio resource management in vehicular networks : A proactive deep reinforcement learning perspective," *IEEE Transactions on Wireless Communications*, 2020.

[2] T. Zhang, S. Shen, S. Mao, and G.-K. Chang, "Delay-aware cellular traffic scheduling with deep reinforcement learning," in *IEEE Global Communications Conference (GLOBECOM)*, 2020.

[3] Q. Wang, T. Nguyen, and B. Bose, "Towards adaptive packet scheduler with deep-q reinforcement learning," in *International Conference on Computing, Networking and Communications (ICNC)*, 2020.

[4] W. AlQwider, T. F. Rahman, and V. Marojevic, "Deep Q-network for 5G NR downlink scheduling," in *IEEE International Conference on Communications Workshops (ICC Workshops)*, 2022.

[5] S. Shen, T. Zhang, S. Mao, and G.-K. Chang, "DRL-based channel and latency aware radio resource allocation for 5G service-oriented RoF-MmWave RAN," *Journal of Lightwave Technology*, 2021.

[6] F. Capozzi, G. Piro, L. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in LTE cellular networks : Key design issues and a survey," *IEEE Communications Surveys & Tutorials*, 2013.

[7] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, 1992.

[8] B. Sadiq, R. Madan, and A. Sampath, "Downlink scheduling for multiclass traffic in LTE," *EURASIP Journal on Wireless Communications and Networking*, 2009.

[9] S. Shakkottai and A. L. Stolyar, "Scheduling algorithms for a mixture of real-time and non-real-time data in HDR," in *Proc. 17th International Teletraffic Congress (ITC-17)*, 2001.