Optimisation et implémentation embarquée du Q-Learning sur FPGA pour la navigation autonome de robots mobiles logistiques en environnement déterministe à forte densité d'obstacles

Marouane BEN-AKKA[1] Camel TANOUGAST[2] Nidhal REZG[1,2]

[1]Université de Lorraine, LGIPM, Equipe SyLEE, 1 rue Augustin Fresnel, BP 45112, 57073 Metz, France

[2]Institut en Innovation Logistique, LGIPM, Equipe SyLEE, 2 rue Valentin Bousch, 57070 Metz, France

Résumé – La planification de trajectoire constitue une tâche fondamentale en robotique mobile, guidant un agent d'une position de départ jusqu'à une cible en évitant les obstacles. L'intégration de méthodes de planification efficaces pour les robots mobiles logistiques représente un enjeu crucial afin d'assurer des transports autonomes fiables et performants en entrepôt, où la navigation en environnement dense, la réactivité temps réel et la faible consommation énergétique sont essentielles. Cet article présente une implémentation sur FPGA d'une stratégie epsilon-greedy à décroissance progressive, adaptée à l'algorithme de Q-Learning, dans le but d'optimiser la planification locale de trajectoire pour des robots mobiles. Des simulations et une implémentation matérielle sont proposées afin d'évaluer la performance de générateurs de politiques basés sur le Q-Learning dans un environnement déterministe caractérisé par une forte densité d'obstacles (30 %). Comparativement aux travaux similaires, les résultats d'implémentation sur FPGA montrent que les générateurs de politiques proposés, reposant sur un registre à décalage à rétroaction linéaire (LFSR) 16 bits, offrent un bon compromis entre performance du Q-Learning, consommation de ressources logiques et efficacité énergétique, les rendant adaptés à une implémentation embarquée d'un planificateur temps réel de trajectoire pour robots mobiles.

Abstract – Path planning is a fundamental task in mobile robotics, guiding an agent from a starting position to a target while avoiding obstacles. The integration of efficient path planning methods for mobile logistics robots is a key challenge to ensure reliable and autonomous transportation in warehouse environments, where dense obstacles, real- time responsiveness, and low energy consumption are critical requirements. This paper presents a FPGA implementation of a decaying epsilon-greedy strategy adapted to the Q-Learning algorithm, aiming to optimize local path planning for mobile robots. Simulations and hardware implementation are provided to evaluate the performance of Q-Learning-based policy generators in a deterministic environment with a high obstacle density (30 %). Compared to similar works, the FPGA implementation results show that the proposed 16-bit linear feedback shift register (LFSR-based) policy generators offer a good trade-off between Q-Learning performance, logic resource usage, and energy efficiency, making them suitable for embedded real-time mobile robot path planning.

1 Introduction

Dans le domaine de la logistique, notamment dans les entrepôts automatisés, les centres de tri ou les hubs de distribution, la navigation autonome de robots mobiles est essentielle pour assurer un transport fluide et optimisé des marchandises. Ces environnements, souvent caractérisés par une forte densité d'obstacles tels que des étagères, d'autres robots et du personnel, sont dynamiques et requièrent une prise de décision rapide en temps réel pour éviter les collisions et garantir une efficacité opérationnelle [3]. L'intégration de solutions de planification de trajectoire efficaces et économes en énergie est donc cruciale pour améliorer la performance et l'autonomie de ces systèmes robotiques logistiques. Le Q-Learning est un algorithme fondamental de l'apprentissage par renforcement (Reinforcement Learning, RL), permettant à un agent de prendre des décisions optimales en interagissant avec son environnement par essais et erreurs [5]. Parmi ses nombreuses applications, la planification de trajectoire en robotique mobile occupe une place centrale. Elle vise à déterminer une trajectoire optimale

permettant à un agent de se déplacer d'un point de départ vers une cible tout en évitant les obstacles [6, 7]. Bien que le Q-Learning ait démontré de bonnes performances dans des environnements à faible densité d'obstacles, son efficacité diminue significativement lorsque cette densité augmente, entraînant une convergence plus lente et une dégradation des performances globales. Dans le domaine de la planification de trajectoire, une densité d'obstacles de l'ordre de 30 % est généralement considérée comme significative, en particulier lorsque les obstacles sont distribués de manière aléatoire ou selon des configurations complexes. La littérature distingue couramment trois niveaux de densité : inférieur à 10 % (faible), entre 10 % et 25 % (modérée), et au-delà de 25– 30 % (forte densité), notamment lorsque la structure de l'environnement complique la navigation. Par ailleurs, bien que le Q-Learning ait été appliqué avec succès dans divers domaines, son intégration dans des systèmes contraints en ressources et opérant en temps réel reste un défi. Dans ces contextes, où une faible latence et un traitement rapide des données sont requis, l'implémentation matérielle sur FPGA apparaît comme une solution pertinente [2].

Dans cet article, nous proposons une implémentation sur FPGA d'une nouvelle stratégie epsilon-greedy à décroissance visant à optimiser le Q-Learning pour la planification de trajectoire de robots mobiles. L'objectif est d'améliorer la vitesse de convergence, la stabilité des récompenses et

l'efficacité de l'exploration. Pour valider cette approche, l'algorithme proposé est implémenté sur une plateforme Xilinx Zynq et évalué dans un environnement maillé à 30 % de densité d'obstacles. L'article est structuré comme suit : la section 2 présente brièvement les principes du Q- Learning et introduit la stratégie de sélection d'action adoptée. La section 3 décrit le modèle de planification de trajectoire. La section 4 présente les résultats et comparaisons expérimentales. Enfin, la section 5 conclut et propose des perspectives.

2 Principes et architecture Q-Learning

Q-Learning est un algorithme d'apprentissage par renforcement hors-politique qui estime la valeur des paires état-action afin de dériver une politique optimale [4]. Il repose sur une matrice Q(*s*, *a*) de taille $|S|x|A|$, où $|S|$ représente le nombre d'états et $|A|$ le nombre d'actions possibles. Les valeurs Q(*s*, *a*) quantifient l'utilité d'exécuter l'action a dans l'état s. L'architecture fonctionnelle du Q-Learning est illustrée à la Figure 1. Elle est composée de deux blocs principaux : le Q-Learning Accelerator et le Policy Generator. Le bloc Q-Learning Accelerator repose sur des approches de calcul approximatif et de parallélisation du nombre d'actions (Z). Il est chargé de stocker la matrice Q et de mettre à jour les valeurs Q(*s*,*a*). Le bloc Policy Generator génère les actions exécutées par l'agent selon une politique de sélection d'actions.

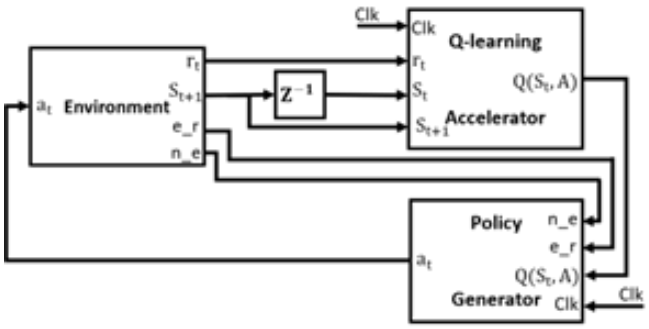

FIGURE 1 : Architecture fonctionnelle du Q-Learning.

La politique d'apprentissage repose sur une stratégie ϵ-greedy décroissante qui combine l'exploitation, consistant à choisir l'action maximisant le retour espéré, soit maxQ(*s*, *a*), et l'exploration, qui sélectionne une action aléatoire générée par un pseudo-générateur aléatoire (PRNG) [1], avec une décroissance adaptative de ϵ : au début de l'apprentissage, l'agent explore totalement l'environnement, puis à chaque épisode, le paramètre ϵ est réduit de 3 % lorsque la récompense totale obtenue dépasse celle de l'épisode précédent, ce qui permet d'adapter dynamiquement la stratégie au niveau d'apprentissage ; cette politique s'appuie sur plusieurs sous-blocs fonctionnels : le bloc Greedy Action identifie l'action optimale correspondant à la valeur maximale de $Q(S_t, a)$ grâce à une recherche du maximum inspirée des travaux de [8], le PRNG produit une action aléatoire rnd_at encodée sur $\lceil \log_2(Z) \rceil$ bits et une valeur aléatoire rnd_val servant de critère de décision, le bloc Updater Epsilon ajuste linéairement ϵ à la fin de chaque épisode uniquement si la récompense globale e_r dépasse le meilleur score atteint jusqu'alors, et un multiplexeur (MUX) arbitre entre exploration et exploitation en sélectionnant rnd_at si rnd_val est inférieure à ϵ, ou l'action optimale sinon, la sortie étant l'action a_t transmise à l'agent pour interaction avec l'environnement, garantissant ainsi un compromis adaptatif entre exploration et exploitation essentiel au bon déroulement de l'apprentissage.

3 Modélisation de la planification de trajectoire

Une approche largement utilisée pour modéliser l'environnement dans la planification de trajectoire de robots mobiles basée sur le Q-Learning est la discrétisation de l'espace en grille. Cette méthode divise l'environnement en une grille structurée, où chaque cellule représente soit un espace libre, soit un obstacle. Cette stratégie de modélisation est particulièrement adaptée aux robots mobiles évoluant dans des environnements intérieurs ou structurés, où les obstacles peuvent être clairement identifiés. Chaque cellule de la grille correspond à une position possible du robot, offrant une représentation simple et extensible de l'environnement. Dans ce cadre, l'espace d'action du robot est défini par l'ensemble des déplacements possibles d'une cellule vers une cellule voisine. Chaque cellule étant supposée plus petite que le rayon de braquage du robot, les transitions entre cellules adjacentes sont considérées comme réalisables. Pour simplifier le déplacement, quatre actions fondamentales sont définies : monter, descendre, aller à gauche et aller à droite. Chaque action correspond à un déplacement d'une unité depuis la cellule courante vers une cellule adjacente. Cette stratégie de déplacement permet à l'agent d'explorer l'environnement de manière systématique, en ajustant sa position en fonction des valeurs Q associées à chaque paire état-action.

4 Simulation et implémentation FPGA

L'environnement de simulation a été configuré pour évaluer les performances de l'agent un environnement maillé de taille 50×50, avec une densité d'obstacles fixée à 30 %, comme illustré à la Figure 2.
L'algorithme implémenté simule le comportement de l'agent au sein de ces environnements, en utilisant les paramètres suivants : taux d'apprentissage de 0,625, un facteur de réduction de la récompense de 0,875, un nombre maximal d'étapes par épisode de 500, un nombre maximal d'épisodes de 1000. Le système de récompenses est défini comme suit :

- Une récompense négative de -50 en cas de collision.
- Une récompense maximale de +200 lors de l'atteinte de l'objectif.
- Une pénalité de -5 pour les états intermédiaires non terminaux.
- Une pénalité de -10 pour dissuader les visites répétées d'un même état au cours d'un seul épisode.

FIGURE 2 : Schéma de l'environnement maillé 50 × 50 avec 30 % d'obstacles.

Ce schéma de récompense discret permet à l'agent d'estimer efficacement les valeurs Q(s, a) et de favoriser l'apprentissage de trajectoires optimales. L'évolution des récompenses cumulées au fil des épisodes pour cet environnement à 30 % d'obstacles, est présentée à la Figure 3. L'axe horizontal représente le nombre d'épisodes, tandis que l'axe vertical indique la récompense totale obtenue par l'agent à chaque épisode. Les résultats montrent que la combinaison de la fonction de récompense discrète et du mécanisme d'exploration adaptatif est déterminante dans les performances améliorées de l'algorithme Q-Learning.

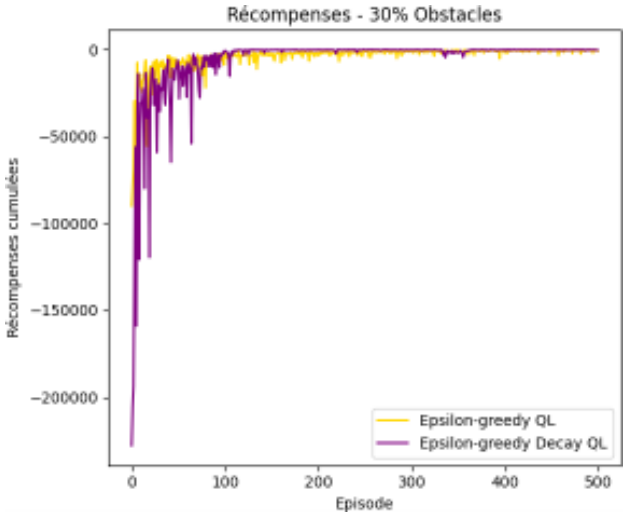

FIGURE 3 : Variation des récompenses au fil des épisodes dans des environnements maillés 50 × 50 avec une densité d'obstacles de 30 %.

La Figure 4 présente les résultats d'expériences de simulation réalisées dans un environnement maillé symétrique couramment utilisé dans les tests de planification de trajectoire. Afin d'évaluer le générateur de politiques proposé, les simulations sont menées dans un scénario expérimental constitué d'un environnement maillé multi-obstacles de 12 × 12 cellules. Un générateur de politique (PG) basés sur un LSFR de différentes tailles de mots binaires ont été étudiés. En tenant compte de la stratégie proposée, les résultats montrent qu'un PG basé sur un LFSR 16 bits offre de meilleures performances par rapport aux autres tailles de mots binaires. La politique optimale est atteinte en 77.4 us après 98 épisodes, et en 711 us après 91 épisodes, respectivement selon les configurations testées.

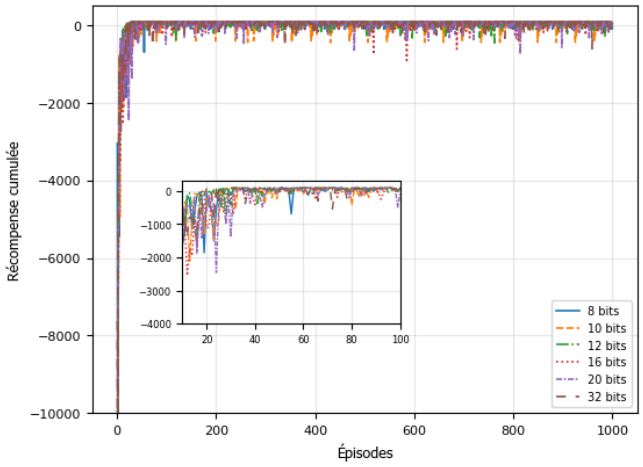

FIGURE 4 : Récompenses totales dans un environnement maillé 12 × 12 utilisant des générateurs de politiques basés sur LFSR.

Les résultats d'implémentation FPGA des architectures proposées de générateurs de politiques basés sur des LFSR sont présentés dans le Tableau 1, pour la carte FPGA Xilinx UltraScale+ ZCU104. Les architectures ont été décrites au niveau RTL à l'aide du langage matériel VHDL, en utilisant l'environnement de développement Vivado 2022.1. Le Tableau 1 fournit également une comparaison entre les implémentations proposées et des travaux antérieurs similaires, en considérant un même nombre d'actions Z=4, et des architectures à base de LFSR reposant sur la même technologie FPGA (Xilinx ZCU). Le générateur de politiques basé sur un LFSR 16 bits proposé consomme uniquement 15 mW, avec une utilisation de 101 LUTs et 18 FFs. Comparativement aux travaux précédents, l'originalité principale des PGs proposés réside dans l'intégration d'une stratégie epsilon-greedy décroissante, permettant un ajustement progressif de l'exploration. Cette approche renforce l'exploitation, améliore les retours cumulés après apprentissage, et offre un bon compromis entre temps d'apprentissage et consommation de ressources logiques.

5 Conclusion et perspectives

Les résultats expérimentaux obtenus à travers les différentes simulations montrent que l'architecture de générateur de politiques proposée, intégrant une stratégie epsilon-greedy décroissante fondée sur l'évolution des récompenses cumulées au fil des épisodes, offre un bon compromis entre

TABLE 1 : Comparaison des implémentations sur FPGA.

	Réf [1]	Proposition
Taille LFSR (bits)	32	16
Générateur politique	epsilon-greedy	Adaptative décroissant greedy
Nombre actions (Z)	4	4
Fréquence (MHz)	1800	923
LUTs	68	101
FFs	32	18
Power (mW)	4	15

temps d'apprentissage et consommation de ressources logiques. Les résultats d'implémentation sur FPGA confirment qu'un générateur de politique basé sur un LFSR 16 bits assure une meilleure convergence dans les environnements considérés, tout en nécessitant peu de ressources logiques et une faible consommation dynamique. En particulier, le générateur de politiques exécutant la stratégie epsilon-greedy décroissante, fonctionne à 923 MHz, utilise 101 LUTs, 18 FFs, et consomme seulement 15 mW. Dans le cadre de travaux futurs, cette approche sera étendue à l'évaluation et l'implémentation d'un générateur de politiques pseudo- aléatoire (PRNG) optimisé pour une architecture double Q-Learning Accelerator.

Références

[1] Gian Carlo CARDARILLI, Luca DI NUNZIO, Rocco FAZZOLARI, Daniele GIARDINO, Marco MATTA, Marco RE et Sergio SPANÒ : An action-selection policy generator for reinforcement learning hardware accelerators. *In Applications in Electronics Pervading Industry, Environment and Society : APPLEPIES 2020 8*, pages 267–272. Springer, 2021.

[2] Lucileide MD DA SILVA, Matheus F TORQUATO et Marcelo AC FERNANDES : Parallel implementation of reinforcement learning q-learning technique for fpga. *IEEE Access*, 7:2782–2798, 2018.

[3] Seth FARRELL, Chenghao LI, Hongzhan YU, Ryo YOSHIMITSU, Sicun GAO et Henrik I CHRISTENSEN : Safe human robot navigation in warehouse scenario. *arXiv preprint arXiv :2503.21141*, 2025.

[4] Beakcheol JANG, Myeonghwi KIM, Gaspard HARERIMANA et Jong Wook KIM : Q-learning algorithms : A comprehensive classification and applications. *IEEE access*, 7:133653–133667, 2019.

[5] ZH QIN, Ning LI, XT LIU, XL LIU, Q TONG et XH LIU : Overview of research on model-free reinforcement learning. *Computer science*, 48(3):180–187, 2021.

[6] Mohamed REDA, Ahmed ONSY, Amira Y HAIKAL et Ali GHANBARI : Path planning algorithms in the autonomous driving system : A comprehensive review. *Robotics and Autonomous Systems*, 174:104630, 2024.

[7] Younes REGRAGUI et Najem MOUSSA : A real-time path planning for reducing vehicles traveling time in cooperative-intelligent transportation systems. *Simulation Modelling Practice and Theory*, 123:102710, 2023.

[8] Bilgiday YUCE, H Fatih UGURDAG, Sezer GÖREN et Günhan DÜNDAR : Fast and efficient circuit topologies forfinding the maximum of n k-bit numbers. *IEEE Transactions on Computers*, 63(8):1868–1881, 2014.