Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

RÉPUBLIQUE DÉMOCRATIQUE POPULAIRE D'ALGÉRIE Ministère de

L'Enseignement Supérieur et de la Recherche Scientifique

Université de Tlemcen – ABU BAKR BELKAID - Algérie
FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

Projet de diplôme

Pour obtenir un Master en Informatique

Spécialité : Modèle Intelligent et Décision (MID) Sur le sujet :

Développement et simulation d'un système de contrôle intelligent et résilient des feux de circulation pour les intersections via un apprentissage par renforcement approfondi

---

Présenté par :

YACHEUR Adnane

Supervisé par :

M. MATALLAH Hocine (Encadreur) Enseignant chercheur, Université de Tlemcen
M. MOSBAH Mohamed (Co-Encadrant) Professeur d'informatique, Bordeaux INP
Mme EL KHALFI Zeineb (Co-Encadrante) Enseignant-chercheur, CESI LINEAC

Examiné le 27 juin par :

M. BENAMAR Abdelkrim (Président)
M. MEZIANE Abdelfettah (Examinateur)

Année académique : 2023/2024

Remerciements

Je tiens tout d'abord à exprimer ma gratitude envers Dieu, le Tout-Puissant, pour m'avoir accordé la force, la volonté et la patience tout au long de mes années d'études, et pour m'avoir permis de mener à bien ce travail.

Je tiens à remercier chaleureusement mon encadreur, Monsieur ***MOSBAH Mohamed ,*** *et mon co-encadreur, Madame* ***EL KHALFI Zeineb****, pour vos soutiens précieux et vos conseils éclairés tout au long du processus de rédaction de mon mémoire. Votre expertise, votre patience et votre engagement ont été d'une valeur inestimable, et j'ai énormément appris grâce à vous.*

J'adresse également mes remerciements les plus sincères à Monsieur ***MATALLAH Hocine****, chef du département, pour son assistance et ses orientations constructives, ainsi qu'à tout le personnel du département. Leur soutien et leur collaboration ont été inestimables.*

Ma gratitude va à chacun d'entre vous pour votre contribution précieuse à mon mémoire. Votre soutien indéfectible a été un véritable moteur dans la réalisation de ce travail de recherche.

Que Dieu bénisse tous ceux qui ont été impliqués et m'ont aidé à atteindre ce stade.

Dédicaces

Je dédie ce travail à:

Mes chers parents, qui ont tout sacrifié pour que je puisse arriver à ce stade-là, sans eux je n'aurais jamais pu y arriver. Que dieu les garde pour moi.

À mon frère Badreddine Yacine Yacheur, mes sœurs et toute ma grande famille, qui m'ont soutenu moralement tout au long de mon parcours universitaire.

Mes amis et mes camarades de la promotion.

Avec une profonde gratitude,

Y.Adnane

Résumé

Avec l'urbanisation rapide, l'optimisation des feux de circulation est devenue essentielle pour réduire les embouteillages et améliorer l'efficacité du transport urbain. Ce mémoire explore l'utilisation de l'apprentissage par renforcement profond (DRL) pour la gestion des feux de signalisation, en se concentrant sur l'optimisation proximale des politiques (PPO).

Les expérimentations ont été menées dans l'environnement de simulation SUMO (Simulation of Urban MObility) en utilisant des données réelles. Les résultats montrent que le modèle PPO multi-agents améliore significativement la fluidité du trafic et réduit les temps d'attente par rapport aux méthodes traditionnelles.

Mots-clés

Urbanisation, Feux de circulation, Apprentissage par renforcement profond (DRL), Optimisation proximale des politiques (PPO), Simulation de trafic (SUMO), NEAT (NeuroEvolution of Augmenting Topologies), Gestion du trafic urbain, Multi-agent, Intelligence artificielle (IA), Réduction des embouteillages, Efficacité des transports, Environnement de simulation

Table des matières

Abréviations

— DRL : l'apprentissage par renforcement profond.
— PPO : l'optimisation proximale des politiques.
— DQN : Le réseau Q profond
— RL : l'apprentissage par renforcement
— GERTRUDE : une société de gestion électronique de régulation en temps réel pour l'urbanisme, les déplacements et l'environnement
— Flou : un ensemble de véhicule
— SUMO : Simulation of Urban MObility

Liste des tableaux

Table des figures

Introduction Générale

Avec le développement rapide de l'urbanisation, la manière d'améliorer l'efficacité des feux de circulation est devenue une question urgente. Les systèmes traditionnels de contrôle des feux de circulation fonctionnent en suivant des cycles prédéfinis. Cependant, cette fixation de l'ordre et de la durée des feux de circulation est inefficace, ils ne peuvent pas s'adapter efficacement aux changements dynamiques du trafic. Cela conduit souvent à des temps d'attente plus longs pour les véhicules et à des embouteillages.

Afin de remédier à ce déficit, nous avons développé un modèle d'amélioration de la synchronisation des feux de circulation en utilisant l'apprentissage par renforcement profond (DRL). Dans ce schéma, les feux de signalisation peuvent émettre une phase appropriée en fonction de l'état du flux de circulation de chaque direction à l'intersection et ajuster dynamiquement la longueur de phase. Plus précisément, nous adoptons d'abord l'optimisation proximale des politiques (PPO) pour améliorer la convergence et la vitesse du modèle. Ensuite, nous élaborons la conception de l'état, de l'action et de la récompense avec l'état du véhicule défini par Discrete Traffic. Enfin, nous menons des expérimentations sur des données réelles de trafic via la plateforme de simulation de trafic SUMO. Ces données ont été fournies par l'entreprise renommée Gertrude. Les résultats montrent que par rapport au contrôle de synchronisation traditionnel, le système proposé peut réduire efficacement le temps d'attente des véhicules et la longueur de file d'attente dans différents modes de circulation (donne des chiffres comparatifs).

Ce mémoire est composé de trois chapitres. Le premier chapitre présente une introduction générale sur les travaux faits avant ou on les prit comme références pour commencer nos recherches, Le deuxième chapitre se concentre sur la modélisation mathématique des algorithmes principaux du système choisi. Nous avons combiné et appliqué des stratégies de contrôle efficaces afin d'optimiser le temps d'apprentissage et ainsi le temps d'avoir un résultat final satisfaisant. Dans le troisième chapitre, nous avons présenté et discuté des résultats obtenus à partir des simulations réalisées.

Chapitre 1

État de l'art

1.1 Introduction

Ce chapitre revêt une importance capitale dans notre mémoire, car il explore en profondeur l'état actuel de l'art du contrôle des feux de circulation, en mettant en lumière les défis persistants auxquels sont confrontées les infrastructures urbaines dans le monde entier. En effet, la gestion efficace du trafic routier est devenue une priorité majeure pour les autorités municipales et les urbanistes, compte tenu de l'impact significatif des embouteillages sur la qualité de vie, l'environnement et la sécurité publique. Dans le cadre plus large de notre mémoire sur l'intégration de l'intelligence artificielle (IA) dans le contrôle du trafic, ce chapitre s'inscrit comme une étape cruciale pour comprendre les fondements théoriques et pratiques des systèmes de contrôle des feux de circulation. En examinant les méthodes conventionnelles et les avancées récentes dans le domaine de l'IA, nous sommes en mesure de poser les bases nécessaires à notre exploration ultérieure des techniques d'apprentissage automatique et d'optimisation pour améliorer la gestion du trafic urbain. Cet examen approfondi est crucial pour plusieurs raisons.
Tout d'abord, il permet de contextualiser notre travail dans le paysage plus large de la recherche sur le contrôle du trafic routier, en identifiant les tendances émergentes, les défis persistants et les lacunes dans les approches existantes. En comprenant pleinement les limites des méthodes traditionnelles, nous sommes en mesure de justifier la pertinence et la nécessité d'explorer de nouvelles voies, notamment celles basées sur l'intelligence artificielle.De plus, ce chapitre fournit une base solide pour notre propre recherche en fournissant un cadre théorique et conceptuel pour nos travaux ultérieurs. En examinant les travaux antérieurs et en synthétisant les conclusions clés, nous sommes en mesure d'identifier les questions ouvertes et les opportunités de recherche prometteuses dans le domaine du contrôle du trafic.
Enfin, ce chapitre aide à sensibiliser le lecteur à l'importance cruciale de trouver des solutions innovantes et adaptatives pour relever les défis du trafic urbain. Alors que les villes continuent de croître et que les pressions sur les infrastructures routières augmentent, il est impératif d'adopter des approches dynamiques et évolutives pour assurer une mobilité urbaine efficace et durable.

1.2 Problématique

L'usage des feux tricolores remonte à une époque ancienne, c'est en 1868 que le premier système de feux tricolores fut installé à Londres [1], Même si ces premiers systèmes s'étaient révélés efficaces à l'époque, les embouteillages sont aujourd'hui devenus un enjeu important en milieu urbain du monde entier, entraînant des conséquences négatives à la fois sur le plan sociétal et environnemental, telles qu'une augmentation des temps de trajet, de la consommation inutile de carburant ainsi la pollution de l'environnement.

De plus, les embouteillages peuvent également avoir des conséquences plus graves, notamment des pertes en vies humaines lorsque les véhicules d'urgence sont bloqués dans leur trajet.

La solution commet Les systèmes traditionnels de contrôle des feux de circulation ont des limites et ne prennent pas en considération les conditions actuelles de la route pour s'adapter aux modèles de trafic qui ne sont pas stables tout le temps et change de manière dynamique, entraînant coordination inefficace des signaux, un flux de trafic sous-optimal, une congestion exacerbée et peu écologique . Il faut donc une approche plus intelligente et dynamique qui peut surveiller et analyser en permanence les modèles de trafic pour ajuster les horaires des signaux en réponse aux conditions changeantes.

Nous sommes incités par cette situation à explorer de nouvelles méthodes pour relever ce défi, dans le but d'assurer une gestion efficace de la circulation. Il y a un intérêt grandissant pour l'intégration des techniques d'intelligence artificielle (IA) dans les systèmes de contrôle des feux de circulation. En tirant parti des capacités de l'IA en matière d'analyse de données, d'apprentissage automatique et de contrôle adaptatif, dynamique les systèmes de contrôle des feux de circulation routière ont le potentiel d'optimiser la fluidité du trafic, de réduire les embouteillages et d'améliorer le trafic global.

1.3 Travaux Précédents

Les travaux existants sur le contrôle des feux tricolores peuvent être classés en deux catégories : les méthodes conventionnelles et les méthodes DRL.

1.3.1 Méthodes conventionnelles

Le contrôle des feux de circulation est en fait un problème de programmation linéaire en nombres entiers mixtes [2] qui est NP-difficile. Le contrôle des feux de circulation repose fortement sur des paramètres de circulation précis tels que la position et la vitesse des véhicules. Les techniques conventionnelles les mesurent à l'aide de capteurs [3]–[4], par exemple des capteurs infrarouges et des détecteurs à boucle inductive. Collecter les données de trafic et calculer les politiques pour de nombreuses intersections en temps réel, un débit de transmission et des ressources informatiques suffisantes sont nécessaires. Des techniques telles que les communications sans fil [5], [6] et l'informatique en nuage en périphérie [7] peuvent améliorer considérablement le taux de transmission de données et l'efficacité informatique. Les méthodes conventionnelles comprennent principalement les méthodes classiques basées sur l'optimisation [8]–[9], les algorithmes évolutionnaires [10]–[11] et la logique floue [12], [13].

1.3.1.1 les méthodes conventionnelles basées sur l'optimisation

Les méthodes classiques basées sur l'optimisation [8]–[9] ajustent généralement le temps de cycle et les divisions de phase en fonction de paramètres et de règles heuristiques. Le temps pour le feu vert est généralement plus long que nécessaire, ce qui entraîne une grande perte de temps pour les voitures ou les piétons qui attendent.

Méthodes pour l'intersection à route unique. La méthode Webster calcule la durée de cycle souhaitée, puis calcule les intervalles de temps verts, qui sont proportionnels aux ratios des volumes de voies critiques sur chaque phase. Hofri et al. [9] a proposé une politique de seuillage pour une seule intersection routière avec deux files d'attente, qui dépend des données de flux de trafic et des expériences de simulation ; cependant, elle peut ne pas être disponible dans toutes les situations réelles de circulation.

Méthodes pour les intersections de routes multiples. Both GreenWave [14] et Maxband [15] visent à réduire le nombre d'arrêts pour les véhicules ; cependant, GreenWave n'optimise

qu'une seule direction, et Maxband optimise deux directions opposées. GreenWave et Maxband nécessitent la même longueur de cycle pour toutes les intersections. Dans la méthode des feux tricolores auto-organisés (SOTL) [16], les demandes de signal vert de la phase en cours et d'autres phases concurrentes sont mesurées, puis certaines règles sont construites pour prendre des décisions (continu ou commutation). Ses performances dépendent fortement des règles. Dans la méthode de la pression maximale [17], la pression de toutes les phases est calculée, puis choisissez la phase avec la pression maximale. Dans la méthode du système de trafic adaptatif coordonné de Sydney (SCATS) [18], le degré de saturation (DS), c'est-à-dire le rapport entre le temps de vert effectif et le temps de feu de verdure disponible, pour chaque phase du plan de signalisation actuel est calculé, puis les DS pour les autres plans de signalisation sont calculées, et enfin le plan avec le DS total minimum est sélectionné. Pour réduire les grandes fluctuations des flux de circulation, Lammer et al. [19] a proposé une méthode pour ajuster la durée et l'ordre des phases vertes, ce qui permet de stabiliser les files d'attente et la durée du temps rouge.

Ces méthodes ont une faible évolutivité. Si le réseau routier est vaste, le nombre de paramètres est également important, ce qui entraîne beaucoup de temps consacré au calcul et à l'ajustement des paramètres. De plus, les règles heuristiques et certains paramètres sont fortement basés sur des connaissances d'experts sans garantie d'optimalité.

Les algorithmes évolutionnaires [20], [21], en particulier les algorithmes génétiques, sont généralement utilisés dans l'optimisation multi-objectifs, par exemple, la consommation de carburant, le nombre d'arrêts, le débit de circulation et le temps d'attente des véhicules [10]–[21]. En théorie, ces algorithmes sont basés sur le concept de survie du plus apte par l'optimisation stochastique et l'heuristique. Prenons l'exemple des algorithmes génétiques. Les solutions sont codées sous forme de chromosomes, et dans la génération suivante, par la reproduction, et une très faible probabilité de mutation, de nouveaux descendants sont générés, et ceux qui ont de grandes valeurs d'ajustement survivront. De cette façon, à chaque génération ou itération, les solutions se rapprochent des points optimaux locaux, voire globaux. Brian et al. [10] a proposé un algorithme génétique pour minimiser la consommation de carburant, les émissions, les retards et le nombre d'arrêts. Turky et al. [21] a utilisé un algorithme génétique pour contrôler dynamiquement la durée rouge et verte afin d'optimiser le débit du véhicule et du flux piétonnier. Cependant, les performances des algorithmes génétiques peuvent dépendre de certains paramètres tels que la taille de la population, le nombre de générations et la probabilité de mutation, et il n'y a aucune garantie de performance. Par exemple, les algorithmes génétiques peuvent ne pas obtenir de bons résultats si la taille de la population est petite. Par conséquent, les algorithmes évolutionnaires peuvent obtenir une solution défavorable dans un temps limité.

La logique floue [12], [13] peut être utilisée dans les problèmes de contrôle des feux de circulation, car l'ambiguïté des termes linguistiques (par exemple, basse, moyenne et haute vitesse) peut être représentée à l'aide d'ensembles flous. Pour compenser la fluctuation du flux de trafic, Ali et al. [12] a utilisé la logique floue pour surveiller et gérer l'alternance des conditions de circulation entre deux cycles successifs. Hawi et al. [13] a utilisé la logique floue et les réseaux de capteurs sans fil pour contrôler les feux de circulation, où les réseaux de capteurs sans fil collectent des données de trafic en temps réel et la logique floue produit l'ordre d'attribution des feux verts avec la quantité de trafic et le temps d'attente comme entrée. Dans les méthodes ci-dessus, la conception de la logique floue nécessite une grande expertise humaine et une mise à jour régulière des règles, et un raisonnement généralement précis ne peut pas être donné. De plus, les performances dépendent des règles et des paramètres heuristiques, et il n'y a aucune garantie de stabilité ou d'optimalité.

1.3.1.2 Les approches DRL mono-agent et multi-agents

Dans les approches DRL à agent unique, l'agent apprend la valeur Q pour toutes les actions de chaque état et choisit l'action avec la valeur Q approximative maximale sur cet état. Grâce à un processus d'apprentissage bootstrap, l'agent obtient une stratégie pour une tâche de bout en bout avec des données d'entrée brutes. Ces méthodes ne sont appliquées que dans les réseaux routiers de petite ou moyenne taille en raison de leur faible évolutivité. La DQN et ses variantes [35], [36] sont des méthodes couramment utilisées. Liang et al. [35] a proposé d'utiliser le double DQN en duel avec une relecture d'expérience prioritaire pour décider de la durée des feux de circulation. Les méthodes DRL à agent unique ont une vue globale des situations de trafic ; cependant, ils peuvent ne pas être pratiques dans les réseaux routiers à très grande échelle. Dans les approches DRL multi-agents [23], [38]–[37], chaque agent contrôle un feu de circulation à une intersection et considère également les états et les actions de ses quartiers pour coopérer avec eux. La somme de la récompense pondérée des agents voisins plus la sienne est puis traité comme une récompense globale. Liu et al. [39] a proposé un apprentissage en Q coopératif avec un algorithme d'approximation de fonction afin que chaque intersection locale puisse prendre des décisions en coopération sans aucun agent de supervision central. Chu et al. [23] a proposé une méthode DRL multi-agents basée sur l'acteur-critique pour stabiliser la procédure d'apprentissage, améliorer l'observabilité et réduire la difficulté d'apprentissage. Xu et al. [40] a proposé une approche DRL multi-agents pour optimiser les plans de synchronisation des signaux aux intersections. Wu et al. [38] a proposé une méthode DDPG multi-agents pour le contrôle des feux de signalisation, où la mémoire à court terme long (LSTM) [41] est utilisée pour améliorer la stabilité de l'environnement causée par un état observable partiel. Les méthodes DRL multi-agents permettent de contrôler les feux de circulation de manière coopérative entre les agents d'un réseau routier à très grande échelle. Cependant, ces méthodes peuvent généralement ne pas avoir une vue globale des situations de trafic nécessaires à la politique de contrôle favorable. De plus, comment équilibrer les services publics d'un agent et de ses quartiers est également un problème difficile.

Les algorithmes DRL ci-dessus utilisent les réseaux neuronaux de la boîte noire pour s'adapter à la fonction Q et les méthodes de descente de gradient stochastique pour minimiser la fonction de perte non convexe. Ces méthodes ne peuvent pas garantir l'évolutivité. Dans les méthodes DQN, le nombre de valeurs Q approximatives sous chaque état sur toutes les actions augmente de façon exponentielle avec l'augmentation du nombre d'intersections routières. L'algorithme DQN nécessite trop de ressources et de temps de calcul pour estimer les valeurs Q dans un grand réseau routier, ce qui entraîne une faible évolutivité. Les méthodes DRL multi-agents telles que DDPG multi-agents [38] peuvent également ne pas garantir l'évolutivité, car elles sont en boîte noire et la politique optimale peut ne pas être obtenue.

Contrairement aux méthodes existantes, nous appliquons des méthodes DRL, à savoir PPO, pour optimiser la politique de contrôle des feux de circulation dans une intersection à une seule route et un réseau routier en grille. Étant donné que l'algorithme PPO peut gérer des scénarios complexes à l'aide d'actions continues, il est utilisé dans les deux cas : une seule intersection de route et un réseau routier en grille. Nous obtenons la politique optimale dans une seule intersection routière en utilisant une méthode conventionnelle, et obtenons la politique optimale dans un réseau routier quadrillé selon un modèle de trafic spécifié, c'est-à-dire une politique de "vague verte", la vague verte dans le contrôle du trafic désigne une synchronisation des feux de signalisation pour permettre aux véhicules de traverser plusieurs intersections sans s'arrêter, favorisant ainsi un flux de circulation fluide. Les résultats de la simulation montrent que l'algorithme PPO fournit le contrôle optimal et produit un modèle de " vague verte " en temps réel.

1.4 Avantages et Inconvénients des Méthodes

Dans la comparaison des méthodes conventionnelles et des méthodes DRL pour le contrôle des feux de circulation, plusieurs avantages et inconvénients sont à considérer. Tout d'abord, en termes d'adaptabilité aux changements de circulation en temps réel, les méthodes conventionnelles sont souvent limitées par des paramètres préétablis, tandis que les méthodes DRL se distinguent par leur grande capacité à ajuster dynamiquement le contrôle des feux en fonction des variations du trafic. En ce qui concerne la performance potentielle, les méthodes conventionnelles peuvent souffrir d'efficacité sous-optimale dans des conditions complexes de circulation, contrairement aux méthodes DRL qui visent à atteindre une performance optimale ou quasi-optimale grâce à leur capacité d'apprentissage et d'adaptation en temps réel. Concernant l'échelonnabilité, les méthodes conventionnelles sont généralement adaptées aux réseaux routiers de petite et moyenne taille en raison de leur simplicité et de leur faible complexité computationnelle, tandis que les méthodes DRL, bien que plus complexes, sont mieux adaptées pour gérer des réseaux routiers de grande envergure grâce à leur capacité d'apprentissage. En ce qui concerne la complexité computationnelle, les méthodes conventionnelles présentent une faible complexité, les rendant plus faciles à implémenter et à utiliser, tandis que les méthodes DRL nécessitent des ressources computationnelles plus importantes en raison de leur apprentissage par renforcement et de la nécessité de traiter de grandes quantités de données.
Enfin, en ce qui concerne les exigences en matière de connaissances, les méthodes conventionnelles nécessitent une connaissance préalable des schémas de circulation et du timing des feux, ce qui peut limiter leur application dans des environnements inconnus. En revanche, les méthodes DRL ne nécessitent aucune connaissance préalable, car elles apprennent directement à partir des données disponibles en temps réel. En résumé, bien que les méthodes conventionnelles offrent une simplicité d'implémentation, les méthodes DRL présentent des avantages significatifs en termes d'adaptabilité, de performance et de capacité à traiter des réseaux routiers de grande envergure, malgré une complexité computationnelle plus élevée. Selon le contexte et les exigences particulières du problème de contrôle des feux de circulation, il est important de choisir entre les méthodes conventionnelles et DRL. Les méthodes traditionnelles peuvent être adéquates pour les réseaux routiers de petite à moyenne taille avec des schémas de circulation prédictibles. Pour les infrastructures routières à grande échelle avec une circulation complexe, les techniques DRL permettent une plus grande adaptabilité et un meilleur potentiel de performance. Lorsque les ressources computationnelles sont restreintes, il est possible que les méthodes conventionnelles soient plus pratiques.

1.5 Explorations de l'Intelligence Artificielle en monde réale

En l'absence de connaissance préalable des schémas de circulation, les méthodes DRL peuvent tirer des enseignements de l'expérience sans avoir besoin de collecter ou d'analyser des données approfondies. Plusieurs recherches récentes mettent en lumière les avantages de l'utilisation de l'intelligence artificielle (IA) dans la gestion des feux de circulation. Google a mené des essais visant à réduire les temps d'attente aux feux tricolores en utilisant un système basé sur l'IA [42], avec le potentiel de diminuer les émissions de gaz à effet de serre. Cette approche repose sur des prédictions relatives aux conditions de circulation, permettant une coordination optimisée des feux de signalisation.

FIGURE 1.1 – Interface utilisateur du système d'optimisation du trafic urbain de Google

Les résultats initiaux montrent une prometteuse réduction de la consommation de carburant de 10 à 20. Cependant, Google n'a pas encore divulgué en détail les sources de données ou les équipements de collecte associés, suscitant des interrogations sur la méthodologie utilisée. D'autres recherches récentes explorent des solutions basées sur l'Internet des objets (IoT) et le traitement d'images pour rendre les feux tricolores "intelligents". Ces systèmes utilisent des caméras connectées pour surveiller en temps réel la densité du trafic aux carrefours, avec une analyse des images traitées par des algorithmes comme OpenCV pour détecter le nombre de véhicules. Cette analyse informe alors l'ajustement de la synchronisation des feux de circulation pour optimiser le flux de véhicules et réduire les temps d'attente. Malgré les avantages de cette approche, elle présente également des limitations et des aspects à améliorer.

1.6 Conclusion

En résumé, ce chapitre a permis de dresser un état des lieux complet du contrôle des feux de circulation, mettant en évidence les lacunes des méthodes traditionnelles et les opportunités offertes par l'intégration de l'intelligence artificielle. Nous avons examiné les différentes approches, leurs avantages et leurs inconvénients, tout en soulignant l'importance croissante de développer des solutions plus intelligentes et dynamiques pour relever les défis du trafic urbain. En préparant le lecteur pour le chapitre suivant, nous mettons en lumière l'importance de passer de la théorie à la pratique en explorant des cas concrets d'application de l'IA dans le contrôle du trafic routier. Nous examinerons en détail les initiatives existantes, les projets de recherche et les défis à relever pour transformer les concepts théoriques en solutions tangibles et durables pour la gestion du trafic urbain. Ce faisant, nous contribuons à ouvrir de nouvelles perspectives dans le domaine de la mobilité urbaine et à jeter les bases d'un avenir plus fluide et plus durable pour les villes du monde entier.

Chapitre 2

Pilotage des feux en utilisant le DRL

2.1 Introduction

La congestion routière est un problème omniprésent dans les environnements urbains, entraînant une perte de temps, une pollution accrue et une qualité de vie réduite pour les résidents. La résolution des embouteillages nécessite une gestion efficace du flux de circulation, qui repose traditionnellement sur des systèmes de contrôle des feux de circulation à heure fixe. Cependant, ces systèmes ne parviennent souvent pas à s'adapter aux conditions de circulation dynamiques, ce qui entraîne des inefficacités et une augmentation des embouteillages. Pour surmonter ces défis, les chercheurs se sont tournés vers l'intelligence artificielle (IA) et les algorithmes d'optimisation pour développer des stratégies de contrôle du trafic plus adaptatives et plus efficaces. Dans ce chapitre, nous explorons deux de ces approches : l'algorithme NeuroEvolution of Augmenting Topologies (NEAT) et les algorithmes d'apprentissage par renforcement (RL). Ces algorithmes offrent des pistes prometteuses pour améliorer le contrôle du trafic en tirant respectivement parti des principes évolutifs et de l'optimisation dynamique des politiques. L'algorithme NEAT utilise des algorithmes génétiques pour développer des réseaux de neurones capables de s'adapter de manière autonome aux conditions de circulation changeantes. En affinant de manière itérative les architectures de réseau sur la base des retours de performances, NEAT vise à développer des stratégies efficaces de contrôle des feux de circulation. D'autre part, les algorithmes d'apprentissage par renforcement entraînent les agents à interagir avec des environnements de simulation de trafic, en apprenant des politiques de contrôle optimales par essais, erreurs et retours de renforcement. En plus de discuter de la mise en œuvre et de l'exécution de ces algorithmes, nous explorerons également l'importance d'une modélisation précise et de la création d'un environnement réaliste à l'aide d'outils tels que Simulation of Urban MObility (SUMO) et NetEdit. Une modélisation précise des réseaux routiers, de la dynamique des flux de circulation et des détails des infrastructures est essentielle pour fournir aux modèles d'IA des environnements réalistes dans lesquels s'entraîner. En incorporant des données de trafic en temps réel et en simulant divers scénarios de trafic, nous pouvons améliorer les capacités d'apprentissage des modèles d'IA et développer des stratégies robustes de contrôle du trafic. Grâce à une évaluation et une comparaison empirique, nous visons à analyser l'efficacité, l'évolutivité et l'efficience informatique des algorithmes NEAT et d'apprentissage par renforcement pour le contrôle du trafic. En comprenant les forces et les limites de chaque approche, nous pouvons identifier des opportunités de recherche et de développement supplémentaires dans le domaine de l'optimisation du trafic. Dans l'ensemble, ce chapitre donne un aperçu de l'application de l'IA et des algorithmes d'optimisation pour le contrôle du trafic et souligne l'importance d'une modélisation précise et de la création d'un environnement réaliste dans la formation de modèles de gestion du trafic efficaces. En tirant parti de ces techniques avancées, nous pouvons œuvrer à la réduction des embouteillages et à l'amélioration de l'efficacité des systèmes de transport

urbain.

2.2 Modélisation des intersections à l'aide de SUMO et de Google Maps

La modélisation des intersections est une étape essentielle dans la conception et l'optimisation des systèmes de gestion du trafic urbain. Une gestion efficace de la circulation aux intersections peut réduire considérablement la congestion, réduire les temps de trajet et améliorer la fluidité globale de la circulation. L'objectif de cette thèse est d'entraîner un modèle d'apprentissage par renforcement (RL) pour optimiser la gestion du trafic. Pour y parvenir, nous avons utilisé SUMO (Simulation of Urban Mobility), un simulateur de trafic open-source, et Google Maps pour nous assurer que nos modèles d'intersection représentent fidèlement les configurations réelles. Nous avons modélisé deux types d'intersections : une grille 3x3 et des intersections réelles depuis le périphérique bordelais. Cette section fournit une description complète du processus de modélisation de ces intersections, de l'aération de la circulation et de la configuration des voies.

2.2.1 Aperçu des outils de simulation de trafic

2.2.1.1 SUMO (Simulation de la mobilité urbaine)

SUMO est un puissant outil de simulation de trafic open source conçu pour gérer les réseaux routiers à grande échelle. Il offre des capacités de modélisation détaillées et flexibles, permettant aux utilisateurs de simuler des scénarios de trafic complexes et d'étudier diverses stratégies de gestion du trafic. Les principales caractéristiques de SUMO sont les suivantes :

- Simulation microscopique du trafic : SUMO modélise les mouvements individuels des véhicules, permettant une analyse détaillée de la dynamique du trafic.

- Importation de réseau flexible : Les utilisateurs peuvent importer des réseaux routiers à partir de différents formats, y compris OpenStreetMap et VISUM.

- Feux de circulation personnalisables : SUMO permet une configuration et un contrôle détaillés des feux de circulation aux intersections.

- Documentation étendue et soutien communautaire : SUMO dispose d'une documentation complète et d'une communauté d'utilisateurs active, ce qui la rend accessible aux chercheurs et aux praticiens.

2.2.1.2 Netedit

Netedit est un éditeur de réseau intégré à la suite SUMO qui fournit une interface graphique pour la création et la modification de réseaux routiers. Il simplifie le processus de mise en place d'intersections complexes et de configuration des feux de circulation. Les principales caractéristiques de Netedit sont les suivantes :

- Édition graphique du réseau : Les utilisateurs peuvent créer et modifier visuellement des réseaux routiers, en ajoutant facilement des intersections, des voies et des feux de circulation.

- Visualisation en temps réel : Les modifications apportées au réseau peuvent être visualisées en temps réel, ce qui permet un retour d'information et des ajustements immédiats.

- Intégration avec SUMO : Netedit est entièrement intégré à SUMO, permettant une transition transparente entre la création de réseau et la simulation de trafic.

2.2.2 Modélisation des intersections depuis le périphérique de Bordeaux : Carrefour cours de la libération

Pour une approche de modélisation plus réaliste, des intersections spécifiques du périphérique de Bordeaux ont été choisies. La rocade bordelaise est connue pour ses configurations complexes et ses volumes de trafic intenses, ce qui en fait un excellent sujet d'étude pour la gestion du trafic dans des scénarios réels.

2.2.2.1 Sélection d'intersection

Deux intersections distinctes ont été sélectionnées à l'aide de Google Maps pour obtenir des configurations précises et des données réalistes sur les flux de circulation. Les intersections ont été choisies en fonction de leur complexité et de la densité du trafic. Des caractéristiques clés telles que le nombre de voies, les types de véhicules, et les schémas de circulation typiques ont été notées pour une modélisation précise.

2.2.2.2 Création de modèles

Netedit a été utilisé pour créer et modifier les réseaux routiers, les intersections et les feux de circulation. Une attention particulière a été accordée pour s'assurer que les intersections modélisées reflètent fidèlement les configurations réelles observées dans Google Maps. Les données de Google Maps ont été importées pour garantir que les modèles SUMO correspondent aux configurations réelles, incluant la géométrie de la route, l'emplacement des signaux et la configuration des voies.

2.2.2.3 Configuration détaillée des voies et de la circulation

Chaque intersection modélisée a été configurée pour inclure toutes les voies présentes dans la réalité, y compris les voies d'autobus et les voies réservées aux véhicules d'urgence. Les routes principales et secondaires ont été modélisées pour refléter les configurations du monde réel, et les attributs des voies, tels que la largeur, les limites de vitesse et les types de véhicules, ont été ajustés pour correspondre aux paramètres réels. Les flux de circulation ont été définis à l'aide de fichiers de flux (rou.xml), spécifiant le nombre de véhicules entrant et sortant de chaque intersection par heure. Ces fichiers étaient basés sur des données de trafic réelles obtenues de Google Maps et des rapports de trafic local. Les flux de circulation ont été ajustés pour refléter les conditions réelles, avec des modèles de trafic variant selon les heures de pointe et les heures creuses.

2.2.2.4 Utilisation de SUMO et Netedit

L'utilisation de SUMO et Netedit est essentielle pour créer des modèles précis et effectuer des simulations de trafic réalistes. Netedit fournit une interface graphique intuitive pour créer et modifier des réseaux routiers, des intersections et des feux de circulation, simplifiant le processus de mise en place et de configuration. Les modifications peuvent être visualisées en temps réel, ce qui permet des ajustements immédiats et précis. L'importation de données réelles de Google Maps dans Netedit garantit que les modèles correspondent aux configurations réelles des intersections, incluant tous les détails nécessaires pour une modélisation précise. Cette approche assure que les modèles SUMO reflètent fidèlement les conditions observées dans le monde réel, offrant des simulations fiables et réalistes pour l'étude et l'optimisation de la gestion du trafic.

```
<route id="est2_to_sud" edges=" est2 sud2_go"/>

<route id="est_nourth_to_nord" edges="est_nourth est nord_go"/>
<route id="est_nourth_to_ouest" edges="est_nourth est ouest_go"/>
<route id="est_nourth_to_sud" edges="est_nourth est sud_go sud2_go"/>

<route id="nord_to_ouest" edges=" nord ouest_go"/>
<route id="nord_to_sud" edges=" nord sud_go sud2_go"/>
<route id="nord_to_est" edges=" nord est_go"/>


<route id="ouest_to_sud" edges=" ouest sud_go sud2_go"/>
<route id="ouest_to_est" edges=" ouest est_go"/>
<route id="ouest_to_nord" edges=" ouest nord_go"/>

<route id="sud_to_est" edges=" sud2 sud est_go"/>
<route id="sud_to_nord" edges=" sud2 sud nord_go"/>
<route id="sud_to_ouest" edges=" sud2 sud ouest_go"/>


<flow id="flow_ns" route="nord_to_sud" begin="0" end="25000" vehsPerHour="320" departSpeed="max" departPos="base" departLane="best"/>
<flow id="flow_nw" route="nord_to_ouest" begin="0" end="25000" vehsPerHour="320" departSpeed="max" departPos="base" departLane="best"/
<flow id="flow_ne" route="nord_to_est" begin="0" end="25000" vehsPerHour="201" departSpeed="max" departPos="base" departLane="best"/>
<flow id="flow_sw" route="sud_to_ouest" begin="0" end="25000" vehsPerHour="201" departSpeed="max" departPos="base" departLane="best"/>
<flow id="flow_sn" route="sud_to_nord" begin="0" end="25000" vehsPerHour="320" departSpeed="max" departPos="base" departLane="best"/>
<flow id="flow_se" route="sud_to_est" begin="0" end="25000" vehsPerHour="320" departSpeed="max" departPos="base" departLane="best"/>

<flow id="flow_e2n" route="est2_to_nord" begin="0" end="25000" vehsPerHour="101" departSpeed="max" departPos="base" departLane="best"/
<flow id="flow_e2w" route="est2_to_ouest" begin="0" end="25000" vehsPerHour="101" departSpeed="max" departPos="base" departLane="best"
<flow id="flow_e2s" route="est2_to_sud" begin="0" end="25000" vehsPerHour="51" departSpeed="max" departPos="base" departLane="best"/>
```

FIGURE 2.1 – Fisher .rou pour le carrefour cours de la libération

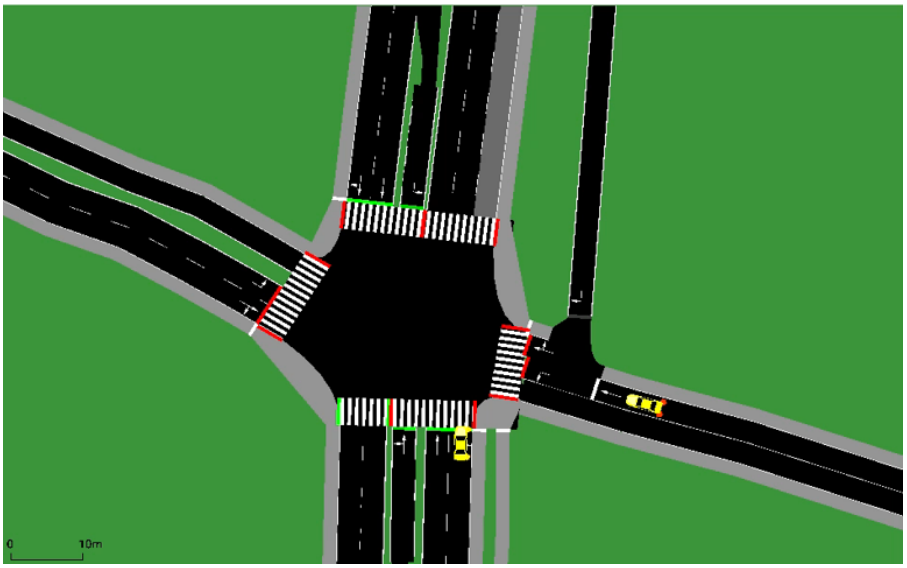

FIGURE 2.2 – Représentation de carrefour cours de la libération

FIGURE 2.3 – vue de Google maps de carrefour cours de la libération

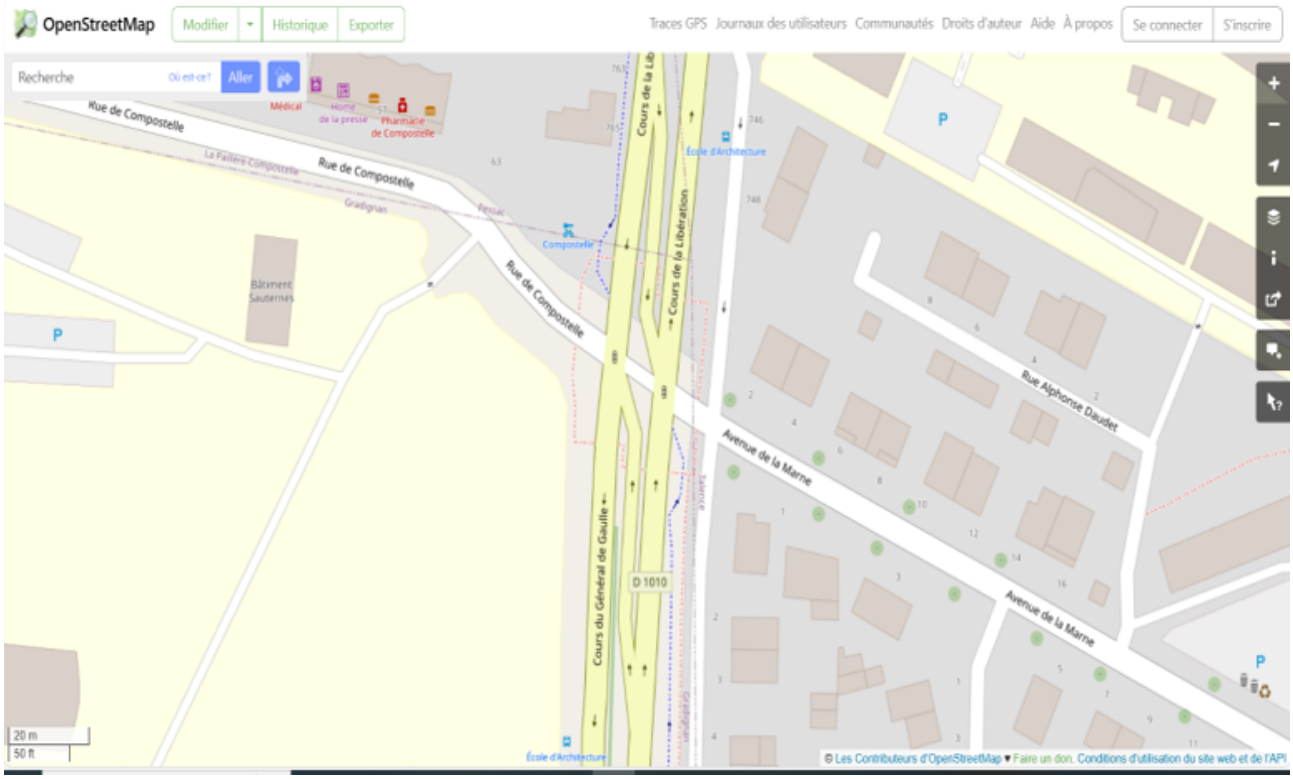

FIGURE 2.4 – vue google street map de carrefour cours de la libération

2.2.3 Modélisation d'une intersection de grille 3x3

La première intersection modélisée dans ce projet est une grille 3x3. Cette configuration, simple mais efficace, et permet de tester les algorithmes de gestion du trafic dans un environ-

nement contrôlé. Une grille 3x3 est composée de neuf intersections disposées en trois rangées et trois colonnes. Chaque intersection est configurée de manière identique, avec des routes perpendiculaires qui se croisent. Cette section détaille la structure de la grille, la création du réseau dans SUMO, et la configuration des flux de circulation.

2.2.3.1 Structure détaillée de la grille 3x3

Les intersections de la grille 3x3 sont conçues pour permettre une gestion efficace et uniforme du trafic. Chaque intersection dispose de quatre directions possibles (nord, sud, est, ouest), et chaque direction comporte des voies spécifiques pour circuler dans les deux sens. Cette uniformité facilite l'étude des stratégies de gestion du trafic. Les routes menant à chaque intersection se composent de quatre voies : deux voies pour le sens aller et deux voies pour le sens retour, garantissant ainsi une circulation bidirectionnelle fluide. Ces voies sont configurées pour accueillir différents types de véhicules, y compris les voitures, les autobus et les véhicules d'urgence. Les intersections sont également équipées de signaux permettant des déplacements dans toutes les directions (tout droit, virage à gauche, virage à droite), offrant une flexibilité dans la gestion du trafic.

2.2.3.2 Création de la grille dans SUMO

La création du réseau de grille a été réalisée à l'aide de Netedit, un éditeur de réseau intégré à la suite SUMO. Chaque intersection a été ajoutée manuellement et les routes reliant les intersections ont été dessinées pour assurer une disposition uniforme. Les attributs des voies tels que la largeur, les limites de vitesse et les types de véhicules ont été configurés pour refléter des environnements urbains réalistes. Des feux de circulation ont été installés à chaque intersection, avec des phases de signalisation définies pour permettre un déplacement sûr et efficace des véhicules dans toutes les directions. Les flux de trafic ont été spécifiés à l'aide de fichiers de routage (rou.xml) dans SUMO, précisant le nombre de véhicules entrant et sortant de chaque intersection par heure. Ces flux ont été conçus pour simuler les schémas de circulation urbains typiques, avec des variations de densité pendant les heures de pointe et les heures creuses. (image sur le Fisher .rou)

```
    <output>
        <output-file value="routese14000.rou.xml"/>
    </output>

</configuration>
-->

<routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/routes_file.xsd">
    <vehicle id="0" depart="0.00">
        <route edges="2Ni 1Ei 0Ei 0Wo"/>
    </vehicle>
    <vehicle id="1" depart="2.00">
        <route edges="8Ei 5Si 2Si 1Ei 0Ei 0No"/>
    </vehicle>
    <vehicle id="2" depart="4.00">
        <route edges="8Ei 5Si 2Si 1Ei 1No"/>
    </vehicle>
    <vehicle id="3" depart="6.00">
        <route edges="6Si 3Si 0Si"/>
    </vehicle>
    <vehicle id="4" depart="8.00">
        <route edges="8Ei 5Si 2Si 1Ei 0Ei 0Wo"/>
    </vehicle>
    <vehicle id="5" depart="10.00">
        <route edges="0Ni 1Wi 2Wi 5Ni 8Ni 8Eo"/>
    </vehicle>
    <vehicle id="6" depart="12.00">
        <route edges="8Ei 7Ei 6Ei 6Wo"/>
    </vehicle>
    <vehicle id="7" depart="14.00">
        <route edges="2Ni 5Ni 8Ni 7Ei"/>
    </vehicle>
```

FIGURE 2.5 – Fisher .rou pour le 3x3 grille

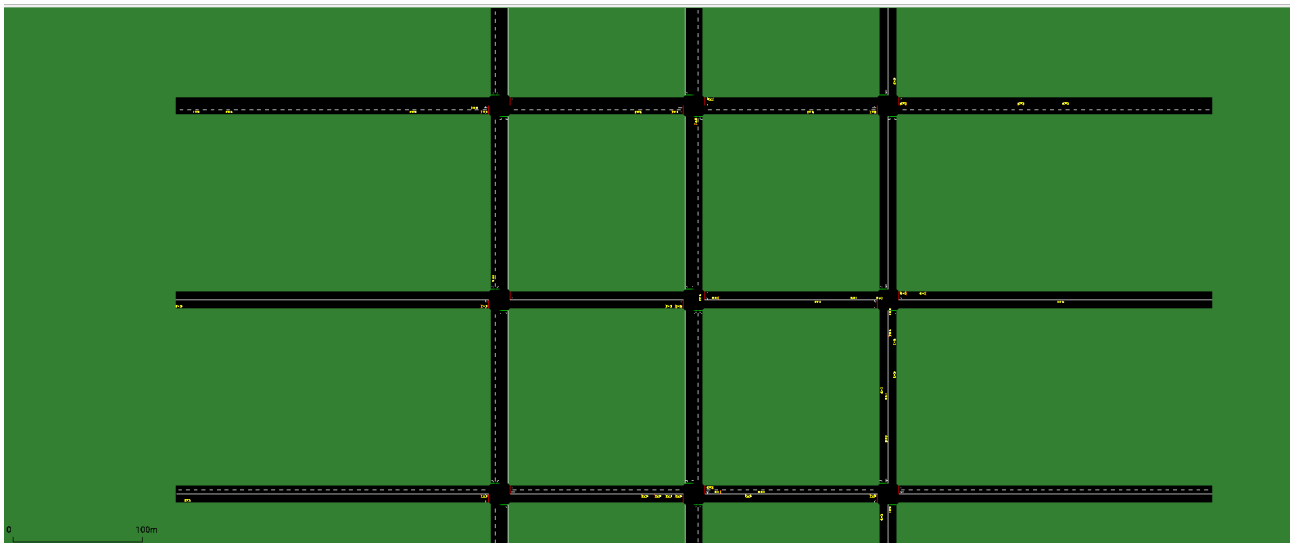

FIGURE 2.6 – Représentation de 9 intersections (3x3 grille)

2.3 Mise en Œuvre de l'IA

Le RL (Reinforcement Learning ou Apprentissage par Renforcement) et le NN (Neural Network ou Réseau de Neurones) sont deux concepts clés de l'intelligence artificielle (IA) et de l'apprentissage automatique (machine learning). Voici une explication détaillée de ces termes :

2.3.0.1 (RL) Apprentissage par Renforcement

L'apprentissage par renforcement est une branche de l'IA où un agent (un programme informatique) apprend à prendre des décisions optimales dans un environnement donné afin de maximiser une récompense ou minimiser une pénalité. L'agent n'est pas explicitement programmé pour effectuer une tâche spécifique, mais il découvre lui-même la meilleure stratégie à adopter en interagissant avec son environnement et en recevant des récompenses ou des pénalités en fonction de ses actions.[43] Les principaux éléments du RL sont :

- Agent : L'entité qui prend des décisions et effectue des actions dans l'environnement.
- Environnement : Le monde dans lequel l'agent évolue et interagit.
- État : Une représentation de la situation actuelle de l'environnement.
- Action : Ce que l'agent peut faire pour modifier l'état de l'environnement.
- Récompense : Un signal de retour (positif ou négatif) que l'agent reçoit après avoir effectué une action, indiquant si cette action était bonne ou mauvaise.

L'objectif de l'agent est d'apprendre une stratégie (une politique) qui maximise la récompense cumulée à long terme en choisissant les meilleures actions possibles dans chaque état.[44]

2.3.0.2 (NN) Réseaux de Neurones

Un réseau de neurones est un modèle d'apprentissage automatique inspiré du fonctionnement du cerveau humain. Il est composé de nœuds (neurones artificiels) interconnectés qui transmettent des signaux entre eux. Les réseaux de neurones sont particulièrement utiles pour résoudre des problèmes complexes qui impliquent la reconnaissance de motifs, la classification, le traitement du langage naturel, etc.[45] Les principaux éléments d'un réseau de neurones sont :

- Neurones : les unités de calcul qui reçoivent des entrées, les pondèrent, les combinent et produisent une sortie.

- Connexions : les liens entre les neurones qui transmettent les signaux d'un neurone à l'autre, pondérés par des poids synaptiques.

- Fonction d'activation : une fonction mathématique qui introduit de la non-linéarité dans le réseau, permettant de modéliser des relations complexes entre les entrées et les sorties.

- Couches : Les neurones sont organisés en couches (entrée, cachées, sortie) où chaque couche effectue un niveau de traitement différent.

Pendant l'entraînement, les poids des connexions sont ajustés de manière à minimiser l'erreur entre la sortie prédite par le réseau et la sortie attendue. Cet ajustement se fait généralement par rétropropagation du gradient (backpropagation).[46] Les réseaux de neurones sont souvent utilisés en conjonction avec l'apprentissage par renforcement, où un agent RL utilise un réseau de neurones pour approximer la fonction de valeur ou la politique d'action, ce qui lui permet de prendre des décisions plus complexes dans des environnements riches.[47]

2.3.1 Implémentation de l'IA : Algorithmes RL & NN

2.3.1.1 NeuroÉvolution des topologies croissantes (NEAT)

L'algorithme NEAT (NeuroEvolution of Augmenting Topologies) est une méthode d'optimisation évolutive utilisée pour faire évoluer de manière autonome des architectures de réseaux neuronaux artificiels en fonction des exigences de la tâche. Contrairement aux architectures de réseaux neuronaux conçues manuellement, NEAT permet aux réseaux de se développer et de s'adapter de manière dynamique, ce qui le rend particulièrement utile pour les tâches dans lesquelles la conception d'une structure de réseau optimale est difficile. NEAT commence avec une population de réseaux simples et fait évoluer leurs architectures au fil des générations en ajoutant des neurones et des connexions par des mutations et des processus de recombinaison. L'algorithme NEAT fonctionne en plusieurs étapes : initialisation, évaluation, sélection, reproduction et mutation. Initialement, des réseaux neuronaux simples sont générés et évalués en fonction de leurs performances dans la tâche donnée, telles que mesurées par une fonction de fitness. Les réseaux les plus performants sont sélectionnés pour la reproduction, où leurs structures sont combinées, et des mutations sont introduites pour créer de nouvelles générations. L'approche innovante de NEAT en matière de mutation comprend l'ajout ou la suppression de neurones et de connexions, permettant au réseau d'explorer un large éventail d'architectures. Ce processus évolutif se poursuit de manière itérative, conduisant à des réseaux neuronaux de plus en plus complexes et efficaces capables de gérer des tâches complexes. L'une des principales caractéristiques de NEAT est sa capacité à conserver un historique des changements structurels, ce qui permet de suivre les innovations et de s'assurer que les caractéristiques utiles sont préservées et propagées d'une génération à l'autre. Ce suivi historique facilite également le croisement des réseaux en faisant correspondre des structures similaires, favorisant l'hérédité des traits bénéfiques tout en évitant les mutations préjudiciables. Bien que NEAT excelle dans l'évolution des architectures réseau pour des tâches spécifiques, il se heurte à des limites dans la gestion d'environnements dynamiques et de grande dimension aussi efficacement que les méthodes d'apprentissage par renforcement profond comme PPO. L'approche évolutive de NEAT peut être gourmande en calcul et peut avoir du mal à atteindre l'évolutivité requise pour les applications à grande échelle. Néanmoins, NEAT fournit une méthode puissante pour découvrir de manière autonome des architectures de réseau innovantes, ce qui le rend précieux dans les scénarios où la conception manuelle n'est pas pratique.

2.3.1.2 Optimisation des politiques proximales (PPO)

L'optimisation des politiques proximales (PPO) est un algorithme d'apprentissage par renforcement très apprécié conçu pour optimiser des politiques complexes pour des tâches de prise

de décision séquentielles. Il équilibre efficacement l'exploration et l'exploitation pour assurer des mises à jour politiques stables et efficaces. Le principe principal de la PPO est d'affiner de manière itérative une politique pour améliorer ses performances dans un environnement donné tout en maintenant la stabilité et la sécurité. Dans l'apprentissage par renforcement, une stratégie représente une correspondance entre les états et les actions qu'un agent utilise pour interagir avec son environnement. PPO introduit le concept d'optimisation proximale des politiques, où les mises à jour des politiques sont contraintes de rester proches de la politique actuelle, empêchant ainsi des changements drastiques qui pourraient déstabiliser le processus d'apprentissage. L'algorithme PPO passe par plusieurs étapes clés : collecte de données, estimation des avantages, mise à jour de la politique et optimisation de la politique proximale. Pendant la collecte des données, l'agent interagit avec l'environnement, recueillant des trajectoires d'états, d'actions, de récompenses et d'états ultérieurs. L'estimation de l'avantage implique le calcul de la fonction d'avantage, qui mesure les performances des actions entreprises par la stratégie actuelle par rapport à une fonction de valeur de référence. Cette fonction se représente comme suivant :

L'estimation d'avantage $A(s_t, a_t)$ mesure l'avantage relatif d'une action (a_t) par rapport à une référence, qui est généralement la valeur d'état $V(s_t)$).

La valeur d'action $Q(s_t, a_t)$ est estimée à l'aide d'une fonction d'évaluation, telle qu'un réseau de neurones, qui prend l'état (s_t) et l'action (a_t) en entrée et produit une estimation de la récompense attendue.

La valeur d'état $V(s_t)$ est également estimée à l'aide d'une fonction d'évaluation similaire, mais elle prend seulement l'état en entrée et produit une estimation de la récompense attendue si l'agent se trouve dans cet état.

Dans le contexte de la formule :

$$A(s_t, at) = Q(s_t, a_t)V(s_t) \tag{2.1}$$

$Q(s_t, a_t)$ représente la récompense attendue si l'action a_t est prise à l'état st.

$V(s_t)$ représente la récompense attendue à long terme si l'agent se trouve dans l'état st.

$A(s_t, a_t)$ est l'estimation d'avantage pour l'action a_t à l'état st.

Où $Q(s_t, a_t)$ est la valeur d'action estimée et $V(s_t)$ est la valeur d'état estimée.

L'étape de mise à jour de la stratégie optimise ensuite la fonction objective en Utilisant un algorithme d'optimisation (tel que la descente de gradient) pour ajuster les paramètres de la politique afin de maximiser la fonction objectif. La fonction objectif est souvent une forme de fonction de perte, qui est minimisée ou maximisée selon le cas. Après, l'étape d'optimisation de la stratégie proximale applique la fonction objectif tronquée pour Limite les changements de politique pour assurer la stabilité et la convergence. En réalisant les Étapes suivantes :

- Les changements de politique sont régis par une fonction de perte "clippée" ou une contrainte de région de confiance.

- Les mises à jour de politique sont limitées pour qu'elles ne s'éloignent pas trop de la politique précédente, ce qui maintient la stabilité et empêche les changements drastiques.

- La fonction de perte "clippée" contraint les mises à jour de la politique pour qu'elles ne s'éloignent pas trop de la politique précédente. Elle utilise le ratio de probabilité entre la nouvelle politique et l'ancienne politique $r - t(\theta)$ pour moduler l'importance des mises à jour.

$$L - \mathrm{CLIP}(\theta) = E - t\left[\min\left(r - t(\theta)a_t, \mathrm{clip}\left(r - t(\theta), 1 - \epsilon, 1 + \epsilon\right) a_t\right)\right] \tag{2.2}$$

où θ représente les paramètres de la politique.

$r - t(\theta)$ est le ratio de probabilité entre la nouvelle politique et l'ancienne politique pour l'action prise à l'étape t.

a_t est l'estimation d'avantage pour l'action prise à l'étape t.

ϵ est un hyperparamètre qui définit une plage acceptable de changement de politique.

En utilisant la fonction min min et la fonction de clip, cette formule garantit que les mises à jour de politique sont limitées à un certain seuil, ce qui favorise la stabilité et la convergence de l'apprentissage. Enfin, l'étape de la répétition ou l'algorithme Itère le processus pour améliorer progressivement la politique, répète les étapes 1 à 4 pour plusieurs itérations ou épisodes d'apprentissage. Au fil du temps, la politique est mise à jour pour mieux s'adapter à l'environnement et maximiser les récompenses cumulatives. L'efficacité et la robustesse du PPO sont attribuées à sa capacité à effectuer des mises à jour contrôlées, garantissant que la politique évolue progressivement et de manière stable. Cela le rend adapté à un large éventail d'applications, notamment la robotique, les jeux et la conduite autonome, où des performances stables et fiables sont cruciales. L'algorithme a obtenu des résultats de pointe sur divers benchmarks d'apprentissage par renforcement, ce qui en fait un choix populaire parmi les chercheurs et les praticiens.

2.3.1.3 Comparaison : PPO vs NEAT

PPO et NEAT, bien qu'ils soient tous deux utilisés dans l'intelligence artificielle, servent des objectifs distincts et excellents dans différents scénarios. PPO, un algorithme d'apprentissage par renforcement profond, est conçu pour entraîner les agents à prendre des décisions séquentielles dans des environnements dynamiques. Il se concentre sur l'affinement des politiques par le biais de mises à jour itératives, garantissant la stabilité et les performances. La force du PPO réside dans sa capacité à gérer des états et des espaces d'action de grande dimension, ce qui le rend adapté à des tâches complexes telles que la conduite autonome, le contrôle robotique et les jeux stratégiques.

En revanche, NEAT est un algorithme évolutif visant à faire évoluer les architectures de réseaux neuronaux pour s'adapter à des tâches spécifiques. Il est particulièrement efficace lorsque la structure optimale du réseau est inconnue ou lorsque la conception manuelle n'est pas pratique. Le processus évolutif de NEAT lui permet de découvrir des architectures réseau innovantes, mais il peut être moins efficace dans des environnements nécessitant une adaptation et une évolutivité rapides. Alors que NEAT peut faire évoluer des architectures qui fonctionnent bien sur les tâches statiques, la capacité de PPO à améliorer de manière itérative les politiques dans des environnements dynamiques en fait souvent un choix plus optimal pour les applications d'apprentissage par renforcement profond. En résumé, alors que NEAT fournit une méthode puissante pour faire évoluer les réseaux neuronaux, l'approche d'apprentissage par renforcement profond de PPO offre des solutions plus robustes et évolutives pour les environnements complexes et dynamiques. L'accent mis par PPO sur la stabilité et l'amélioration progressive garantit des performances fiables, ce qui en fait un choix privilégié pour les applications nécessitant une adaptation et un apprentissage continus. La rigueur mathématique et les mises à jour contrôlées du PPO le rendent particulièrement adapté aux tâches de prise de décision de grande dimension et séquentielles.

2.4 Programme

2.4.1 Explication détaillée de la simulation du trafic et de l'implémentation de l'algorithme NEAT

2.4.1.1 Bibliothèques utilisées

1. NEAT-Python (‘neat‘) : cette bibliothèque est utilisée pour implémenter la NeuroEvolution of Augmenting Topologies (NEAT). Elle facilite l'évolution des réseaux neuronaux en

utilisant des algorithmes génétiques.

2. ‘visualize‘ : bien que ce ne soit pas une bibliothèque standard et qu’elle ne soit pas utilisée dans le code fourni, elle fait généralement référence à un module utilisé pour visualiser l’évolution des réseaux neuronaux.

3. ‘pickle‘ : ce module Python est utilisé pour la sérialisation et la désérialisation des structures d’objets Python, ce qui permet de sauvegarder et de charger des modèles ou des données.

4. ‘os‘ : ce module fournit un moyen d’utiliser des fonctionnalités dépendantes du système d’exploitation, comme la lecture ou l’écriture dans le système de fichiers.

5. ‘sys‘ : ce module donne accès à certaines variables utilisées ou maintenues par l’interpréteur et à des fonctions qui interagissent fortement avec l’interpréteur.

6. ‘traci‘ : il s’agit d’une bibliothèque Python utilisée pour interfacer avec le simulateur de trafic Simulation of Urban MObility (SUMO). Elle permet le contrôle en temps réel et l’accès à la simulation.

7. ‘time‘ : ce module fournit diverses fonctions liées au temps.

8. ‘subprocess‘ : ce module permet de lancer de nouveaux processus, de se connecter à leurs canaux d’entrée/sortie/erreur et d’obtenir leurs codes de retour.

9. ‘multiprocessing‘ : ce module prend en charge le lancement de processus en utilisant une API similaire au module threading.

10. ‘xml.etree.ElementTree (ET)‘ : ce module est utilisé pour analyser et créer des données XML, ce qui est utile pour la gestion des fichiers de configuration et de sortie de SUMO.

11. ‘random‘ : ce module implémente des générateurs de nombres pseudo-aléatoires pour diverses distributions.

2.4.1.2 Déroulement du code et exécution

Le script commence par définir diverses variables globales qui gèrent l’état et la configuration de la simulation ainsi que le contrôle des feux de circulation. Cela inclut des variables comme generation, changed-trafic-time , pinalite-trafic-change-time et orange-time. Une autre variable critique est traffic-light-id, qui stocke l’identifiant du feu de circulation à contrôler dans la simulation SUMO (cela aidera dans les prochaines étapes pour utiliser un système multi-agent où chaque agent à son traffic-light-id).

Ensuite, la fonction ‘parse-vehroutes‘ est définie. Cette fonction analyse le fichier XML de sortie de la simulation SUMO, spécifiquement le fichier des routes des véhicules, et calcule le temps total pris par tous les véhicules dans la simulation en sommant les temps de trajet pour chaque véhicule. Ceci est essentiel pour évaluer l’efficacité de l’algorithme de contrôle du trafic.

La fonction ‘time-taken‘ est un simple conteneur qui appelle ‘parse-vehroutes‘ pour retourner le temps de trajet total. Ensuite, la fonction ‘get-current-traffic-phase‘ récupère la phase actuelle du trafic d’un feu de circulation spécifié en utilisant la fonction ‘traci.trafficlight.getPhase‘, retournant 0, 2, ou 4 selon la phase actuelle. Cette fonction est cruciale pour comprendre l’état actuel du signal de trafic.

La fonction ‘get-input-values‘ rassemble l’état actuel de la simulation de trafic, comme le nombre de véhicules en attente dans chaque direction. Elle parcourt les étapes de la simulation, recueillant les nombres de véhicules à partir de voies spécifiques en utilisant ‘traci.lane. getLastStepVehicleNumber‘. Ces données forment l’entrée du réseau neuronal, lui fournissant le contexte nécessaire pour prendre des décisions.

La fonction ‘change-traffic‘ est responsable de changer la phase du feu de circulation en fonction de la sortie du réseau neuronal. Elle s’assure que le feu reste dans la phase orange pendant une durée définie avant de passer à la nouvelle phase, maintenant ainsi la séquence appropriée des signaux de trafic et garantissant la sécurité.

Pour simuler des schémas de trafic réalistes, la fonction ‘generate-trafic‘ crée un fichier XML

avec des routes de trafic prédéfinies et des trajets de véhicules. Elle utilise une distribution aléatoire pour décider des points de départ et d'arrivée de chaque trajet, assurant certaines probabilités pour des routes spécifiques. Cette variabilité ajoute du réalisme à la simulation, mettant au défi le réseau neuronal de bien performer dans différentes conditions.

La fonction 'Train' est où la formation d'un seul génome (réseau neuronal) au sein du cadre NEAT se produit. Elle exécute la simulation SUMO pendant un nombre spécifié d'étapes, recueille les valeurs d'entrée, active le réseau neuronal pour obtenir des valeurs de sortie, et change la phase du feu de circulation en conséquence. L'aptitude du génome est calculée en fonction des temps de trajet des véhicules et des pénalités encourues pour des changements fréquents des feux de circulation, orientant ainsi le processus d'évolution.

La fonction 'evaluate' sert de noyau à l'algorithme NEAT. Elle évalue une population de génomes en les parcourant, en générant du trafic si nécessaire, en exécutant la simulation SUMO et en formant chaque génome. L'aptitude de chaque génome est calculée et assignée en fonction de la performance du réseau neuronal dans le contrôle des feux de circulation, guidant ainsi le processus de sélection pour la génération suivante.

La fonction 'run-neat' configure la configuration NEAT, crée une population, ajoute des reporters pour surveiller les progrès, et exécute l'algorithme NEAT pendant un nombre spécifié de générations. Ce processus itératif cherche à faire évoluer des réseaux neuronaux de plus en plus performants. Le meilleur génome est sauvegardé dans un fichier pour une utilisation future, assurant que la meilleure solution trouvée peut être réutilisée ou analysée davantage.

Le démarrage de la simulation SUMO est géré par la fonction 'run-sumo', qui lance SUMO en utilisant le fichier de configuration spécifié précédemment. Cette fonction garantit que l'environnement de simulation de trafic est correctement initialisé et prêt pour l'interaction avec le réseau neuronal.

Enfin, l'exécution principale du script est protégée par 'if –name–==', garantissant que le code suivant s'exécute uniquement si le script est exécuté directement. Il crée le répertoire de sortie s'il n'existe pas, initialise le compteur de génération, et lance l'algorithme NEAT, orchestrant l'ensemble du processus, de la simulation de trafic à l'évolution du réseau neuronal.

2.4.1.3 Actions et évaluation du réseau neuronal

L'action principale entreprise par le réseau neuronal est la décision de changer la phase du feu de circulation en fonction de l'état actuel de la simulation de trafic. Ce processus de prise de décision repose sur les valeurs d'entrée collectées à partir de la simulation, comme le nombre de véhicules en attente dans chaque voie. Ces valeurs sont alimentées dans le réseau neuronal, qui produit ensuite des valeurs de sortie indiquant la nouvelle phase du feu de circulation.

L'efficacité du réseau neuronal est évaluée en fonction de sa capacité à minimiser le temps de trajet total pour tous les véhicules dans la simulation. Ce processus d'évaluation implique l'exécution de la simulation SUMO pour un nombre fixe d'étapes, pendant lesquelles le réseau neuronal prend des décisions sur les phases des feux de circulation. Le nombre de véhicules en attente dans chaque voie est collecté à intervalles réguliers et fourni en entrée au réseau neuronal. Sur la base de ces entrées, le réseau génère des valeurs de sortie, guidant les changements de phase des feux de circulation.

L'aptitude de chaque génome (réseau neuronal) est calculée en sommant le temps de trajet total de tous les véhicules et en intégrant les pénalités pour des changements fréquents de feux de circulation. Cette valeur d'aptitude est utilisée par l'algorithme NEAT pour faire évoluer le réseau neuronal, visant à trouver une solution optimale pour le contrôle des feux de circulation. Au fil de plusieurs générations, l'algorithme NEAT sélectionne et affine les génomes, améliorant progressivement leur performance.

En résumé, ce script exploite l'algorithme NEAT et le simulateur de trafic SUMO pour

faire évoluer un réseau neuronal capable de contrôler efficacement les feux de circulation dans un environnement simulé. Le réseau neuronal est formé et évalué en fonction de sa capacité à minimiser le temps de trajet des véhicules, guidant ainsi le processus évolutif pour découvrir la stratégie de contrôle du trafic la plus efficace. Le meilleur réseau neuronal est sauvegardé, garantissant que la solution optimale peut être réutilisée ou analysée davantage dans des études futures.

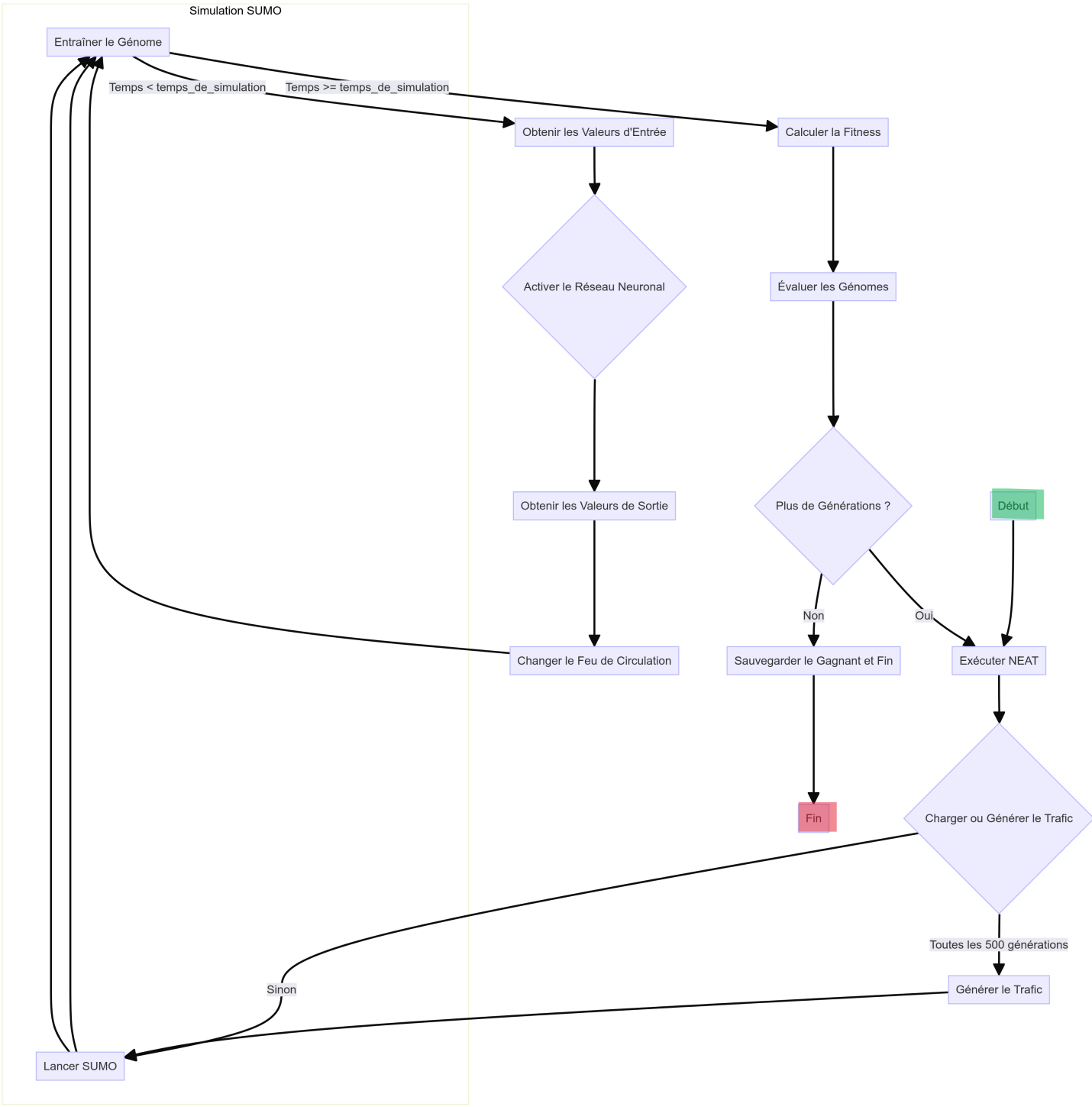

FIGURE 2.7 – Algorithme Neat utilisé

2.4.2 Explication détaillée de la simulation du trafic et de l'implémentation de l'algorithme PPO

2.4.2.1 Un seul agent

2.4.2.1.1 Bibliothèques Utilisées :

1. sumo rl : pour intégrer SUMO avec les algorithmes d'apprentissage par renforcement.
2. sys : pour manipuler les paramètres et fonctions du système.
3. os : pour les interactions avec le système d'exploitation, notamment pour accéder aux variables d'environnement.
4. stable baselines3 : pour les algorithmes d'apprentissage par renforcement (ici, PPO, A2C, DQN).
5. Torch : pour les opérations sur les tenseurs et les réseaux de neurones.

2.4.2.1.2 Déroulement du code et exécution

Le script commence par importer les bibliothèques nécessaires : Le script débute par l'importation des bibliothèques nécessaires, incluant 'argparse', 'sys', 'os', des composants de 'stable-baselines3' tels que 'PPO', 'A2C', et 'DQN', ainsi que des utilitaires de torch et la bibliothèque spécifique 'sumo-rl' pour l'interfaçage avec SUMO (Simulation of Urban Mobility).

Ensuite, il vérifie si la variable d'environnement 'SUMO-HOME' est définie. Cette variable indique où se trouvent les outils SUMO sur le système. Si elle n'est pas définie, le programme quitte avec un message d'erreur : « Please declare the environment variable 'SUMO-HOME'

Le script poursuit en vérifiant la présence de la variable d'environnement 'SUMO-HOME'. Si elle est définie, il ajoute le chemin des outils SUMO à la variable de chemin du système. Sinon, il affiche un message d'erreur et quitte. Ensuite, la fonction 'make-env' est définie pour créer un environnement SUMO. Cette fonction est appelée pour chaque processus d'environnement vectorisé : La fonction 'make-env' est définie pour créer une instance de 'SumoEnvironment', un environnement simulé de gestion du trafic basé sur SUMO. Elle prend un argument 'rank' et initialise l'environnement avec des fichiers réseau et de route spécifiques, une durée de simulation et d'autres paramètres nécessaires. Cette fonction retourne une autre fonction '-init' qui initialise et renvoie l'environnement lorsque appelée.

L'intérieur de 'make-env' initialise une instance de 'SumoEnvironment', qui est l'environnement de simulation de trafic. Les fichiers de réseau et de route sont spécifiés ('OUR-MAP.net.xml' et 'OUR-TRAFFIC.rou.xml'), ainsi que d'autres paramètres comme 'num-seconds' (durée de la simulation) et 'use-gui' (pour l'affichage graphique).

Dans le bloc principal, plusieurs hyperparamètres pour l'algorithme PPO sont définis Dans le bloc principal du script, plusieurs hyperparamètres pour l'algorithme PPO sont définis, tels que 'alpha' pour le taux d'apprentissage, 'gamma' pour le facteur de discount, 'steps' pour le nombre d'étapes, 'batch-size' pour la taille du lot, 'clip-range', 'vf-coef' pour le coefficient de la fonction de valeur, 'ent-coef' pour le coefficient de l'entropie, et 'max-grad-norm' pour la norme maximale des gradients.

Ensuite, l'environnement vectorisé est créé en utilisant 'SubprocVecEnv', qui permet de paralléliser les simulations pour une exécution plus rapide : Ensuite, l'environnement vectorisé est créé en utilisant 'SubprocVecEnv', ce qui permet de paralléliser les simulations en utilisant plusieurs processus, améliorant ainsi l'efficacité et la vitesse d'exécution. Chaque processus d'environnement est initialisé en appelant 'make-env' pour le nombre de CPU spécifié ('num-cpu'). Cet environnement est ensuite normalisé en utilisant 'VecNormalize' pour normaliser les observations et les récompenses.

‘VecNormalize‘ est utilisé pour normaliser les observations et les récompenses, ce qui facilite l’apprentissage pour le modèle.

Le modèle PPO est créé avec les paramètres spécifiés. L’algorithme PPO est choisi pour sa robustesse et son efficacité dans les tâches de contrôle continu :

Le modèle PPO est créé avec les paramètres spécifiés, tels que la politique utilisée (‘MlpPolicy‘), l’environnement, le taux de discount (‘gamma‘), le taux d’apprentissage (‘learning-rate‘), le nombre d’étapes par mise à jour (‘n-steps‘), le nombre d’époques (‘n-epochs‘), la taille du lot (‘batch-size‘), la plage de clipping (‘clip-range‘), et le niveau de verbosité (‘verbose‘).

Le modèle est ensuite entraîné avec un nombre total d’étapes de temps de 250000 : Le modèle est ensuite entraîné en utilisant la méthode ‘learn‘ avec un nombre total d’étapes de temps (‘total-timesteps‘) de 250000. Cette méthode permet au modèle d’apprendre et d’optimiser sa politique de gestion des feux de circulation en interagissant avec l’environnement simulé. Après l’entraînement, le modèle est sauvegardé pour une utilisation future :

Après l’entraînement, le modèle est sauvegardé en utilisant la méthode ‘save‘ avec le nom de fichier spécifié (‘"ppo-model-save"‘), permettant ainsi de réutiliser le modèle entraîné pour des simulations futures ou des analyses supplémentaires.

Enfin, l’environnement est fermé proprement :

Enfin, l’environnement est fermé proprement en utilisant la méthode ‘close‘ pour libérer les ressources et s’assurer que toutes les simulations sont correctement terminées.

2.4.2.1.3 Environnement SUMO

L’environnement SUMO, encapsulé par ‘SumoEnvironment‘, interagit avec le simulateur SUMO pour gérer le trafic routier. SUMO est un simulateur de trafic microscopique qui modélise le déplacement de chaque véhicule individuellement.

1. Observations : les observations dans SUMO incluent des informations sur l’état actuel du trafic, telles que le nombre de véhicules en attente, leur position, la durée d’attente, etc. Ces informations sont cruciales pour l’agent de renforcement pour prendre des décisions éclairées.

2. Actions : les actions prises par l’agent PPO influencent les phases des feux de circulation. Par exemple, une action pourrait être de passer d’un feu rouge à un feu vert, ou d’étendre la durée d’un feu vert pour réduire les temps d’attente.

3. Récompenses : les récompenses sont calculées en fonction des performances de l’action entreprise. Une métrique courante est la réduction des temps d’attente des véhicules. Des récompenses positives encouragent des actions qui diminuent les temps d’attente, tandis que des récompenses négatives découragent des actions inefficaces. Le cycle de l’apprentissage par renforcement se déroule comme suit :

1. Obtention des observations : l’agent reçoit une observation de l’état actuel du trafic.

2. Prise d’Actions : l’agent choisit une action basée sur l’observation reçue.

3. Envoi des actions à SUMO : les actions choisies sont envoyées à SUMO pour modifier les phases des feux de circulation.

4. Modification des Feux de Circulation : SUMO met à jour l’état des feux de circulation selon les actions reçues.

5. Calcul des récompenses : une nouvelle observation est obtenue, et une récompense est calculée en fonction de l’efficacité des actions prises.

6. Répétition du Cycle : ce processus est répété tout au long de la simulation, permettant à l’agent d’apprendre et d’optimiser la gestion du trafic.

2.4.2.2 Multi-Agent

2.4.2.2.1 Bibliothèques utilisées

On a Utilisé la même bibliothèque qu'un seul agent :

1. sumo-rl : intègre SUMO avec les algorithmes d'apprentissage par renforcement.
2. sys : manipule les paramètres et fonctions du système.
3. os : interagit avec le système d'exploitation, notamment en accédant aux variables d'environnement.
4. stable-baselines3 : fournit des algorithmes d'apprentissage par renforcement (notamment PPO).
5. torch : supporte les opérations sur les tenseurs et les utilitaires de réseaux neuronaux.

2.4.2.2.2 Aperçu de l'exécution du code

Le script commence par les mêmes étapes que celui d'un agent unique en important les bibliothèques nécessaires. Ensuite, il définit la fonction 'make-env' pour créer un environnement SUMO.

À l'intérieur de 'make-env', 'PettingZooAECWrapper' est configuré avec des fichiers réseau ('3x3grid/3x3Grid2lanes.net.xml') et de routes spécifiques ('3x3grid/routes14000.rou.xml'), ainsi qu'une durée de simulation ('num-seconds') et d'autres paramètres essentiels.

L'agent ego ('ego-agent') est instancié avec PPO et les paramètres spécifiés, incluant le type de politique ('MlpPolicy'), l'environnement, les hyperparamètres, et le niveau de verbosité ('verbose'). Cet agent sert d'agent principal influençant les phases des feux de circulation en fonction de sa politique.

Pour chaque agent partenaire ('partner'), également instancié avec PPO et des paramètres similaires, l'environnement est initialisé avec 'env.getDummyEnv(i)', représentant une configuration d'environnement factice pour chaque agent partenaire. Chaque agent partenaire est ajouté à l'environnement en utilisant 'env.add-partner-agent(partner, player-num=i + 1)', où 'player-num' indique le numéro séquentiel des agents partenaires à partir de 1.

Le modèle PPO pour les agents ego et partenaires est ensuite entraîné en utilisant la méthode 'learn' sur un nombre total d'étapes ('total-timesteps'), permettant à chaque agent d'interagir avec l'environnement simulé et d'optimiser sa politique de gestion du trafic.

Après l'entraînement, les modèles finaux sont sauvegardés en utilisant la méthode 'save' pour une utilisation future ou une analyse supplémentaire. Enfin, le script assure la fermeture correcte de l'environnement en utilisant la méthode 'close' pour libérer les ressources et terminer les simulations proprement.

2.4.2.2.3 Aperçu de l'environnement multi-Agent

La configuration multi-agent étend le cadre d'un agent unique en définissant un agent ego et plusieurs agents partenaires. Chaque agent partenaire partage le même espace d'observations tel que défini par l'environnement. L'agent ego est initialisé et sert d'agent principal influençant les phases des feux de circulation en fonction de sa politique. Les agents partenaires, numérotés séquentiellement, suivent une optimisation de politique similaire grâce à l'entraînement de modèles partagés.

2.4.2.3 Conclusion

En résumé, ce script utilise SUMO pour la simulation de l'environnement de trafic et PPO pour l'optimisation du contrôle des feux de circulation. Les agents d'apprentissage par renforcement observent les conditions de circulation, prennent des décisions sur les actions à entreprendre et reçoivent des récompenses en fonction de l'efficacité de leurs actions. À travers un

apprentissage itératif, les agents améliorent la gestion du trafic, réduisant les temps d'attente et améliorant l'efficacité globale du transport urbain.

Ce projet utilise la bibliothèque 'sumo-rl' (https ://github.com/LucasAlegre/sumo-rl) pour créer des environnements compatibles avec SUMO pour les algorithmes d'apprentissage par renforcement.

Cette approche structurée assure une clarté dans la mise en place et l'entraînement d'un système multi-agent pour le contrôle du trafic, exploitant efficacement les techniques d'apprentissage par renforcement à travers des agents coopératifs.

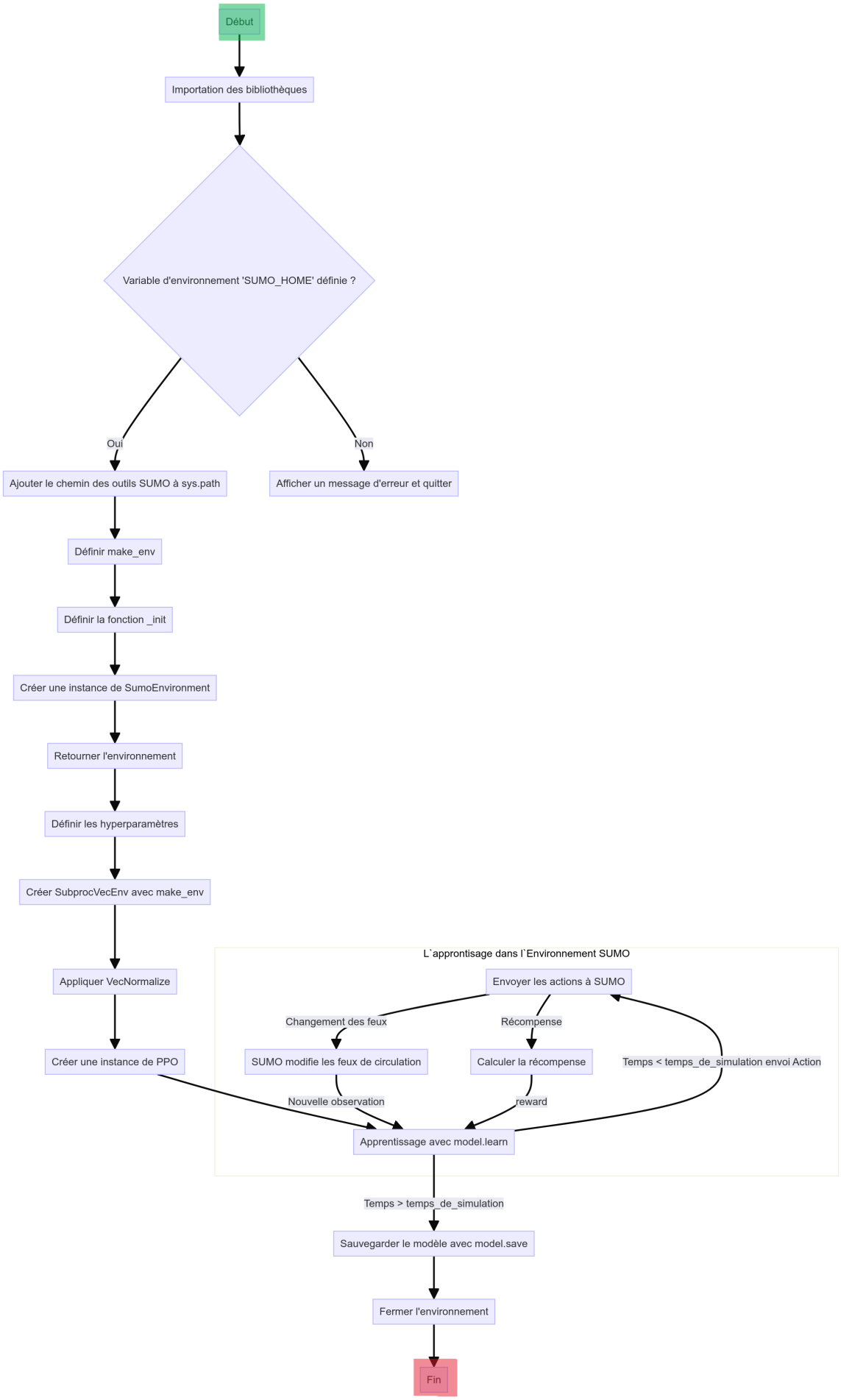

FIGURE 2.8 – Algorithme PPO utilisé mono-agent

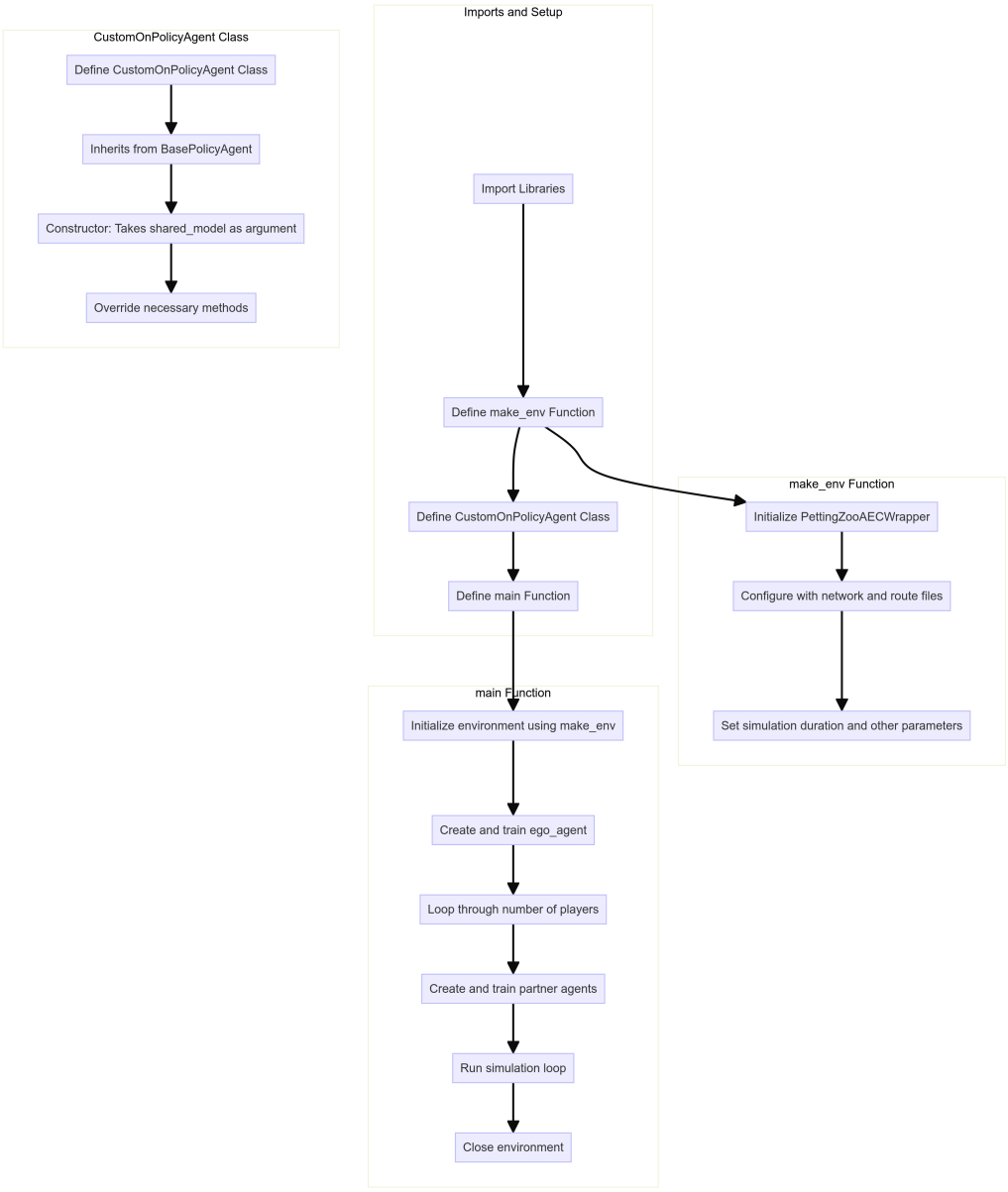

FIGURE 2.9 – Algorithme PPO utilisé multi agent

2.5 Conclusions

En résumé, les chapitres sur l'algorithme NEAT et l'apprentissage par renforcement (RL) pour le contrôle du trafic avec SUMO explorent deux méthodes distinctes pour optimiser le trafic. NEAT utilise des principes évolutifs pour faire évoluer des réseaux neuronaux, améliorant progressivement les stratégies de gestion du trafic à travers des générations successives. En revanche, l'apprentissage par renforcement forme un agent à interagir directement avec l'environnement de simulation, ajustant ses politiques de contrôle en réponse aux retours pour minimiser la congestion.

Les deux approches visent à améliorer l'efficacité du trafic, mais chacune a ses avantages et ses limites. NEAT excelle dans la découverte de nouvelles architectures de réseau, bien qu'il puisse être limité par sa complexité computationnelle. L'apprentissage par renforcement est mieux adapté à l'optimisation en temps réel, mais nécessite une formation approfondie.

Le chapitre des résultats analysera l'efficacité, la scalabilité et la performance computationnelle de ces approches. Une modélisation précise et un environnement réaliste avec SUMO et

NetEdit sont essentiels pour la formation des modèles, améliorant leur capacité à généraliser et à s'adapter aux conditions réelles. L'intégration des données en temps réel, comme celles de Google Maps, renforce encore cette fidélité, permettant aux modèles de réagir dynamiquement aux variations de trafic.

En conclusion, en combinant une modélisation précise avec des données de trafic en temps réel, nous pouvons former des modèles d'IA capables de développer des stratégies de gestion du trafic robustes et efficaces, répondant aux défis du monde réel.

Chapitre 3

Présentation et discussion des résultats

Dans ce chapitre, nous présentons les résultats obtenus à partir des simulations réalisées sur la plateforme SUMO, en utilisant des données de trafic réelles. Nous discuterons des performances de notre modèle en termes de réduction du temps d'attente des véhicules et de la longueur des files d'attente, et comparerons ces résultats avec ceux des méthodes traditionnelles.

3.1 Introduction

Ce chapitre présente les résultats obtenus de l'application de l'apprentissage par renforcement profond (DRL) et de la NeuroEvolution des topologies augmentantes (NEAT) pour contrôler les feux de circulation. Les expériences ont été réalisées en utilisant des approches à agent unique et multi-agents. La performance de ces modèles a été évaluée par rapport aux systèmes de gestion de la circulation conventionnels afin de démontrer les améliorations potentielles du flux de trafic et de la réduction des temps d'attente.

3.2 Récapitulatif de la méthodologie

Nous avons effectué des simulations en utilisant l'outil SUMO (Simulation de la Mobilité Urbaine), en modélisant diverses intersections, y compris une structure détaillée de grille 3x3. Les algorithmes implémentés incluent l'Optimisation Proximale des Politiques (PPO) pour le DRL et NEAT. Les configurations à agent unique et multi-agents ont été testées pour comparer leurs performances.

3.3 Résultats expérimentaux

3.3.1 Résultats de l'algorithme NEAT avec un seul agent

3.3.1.1 Simulation et Calcul du Temps

3.3.1.1.1 Environnement de simulation :

Pour l'algorithme NEAT (NeuroEvolution des Topologies Augmentantes), l'intersection de Bordeaux a été simulée avec un flux continu de véhicules avec un nombre de 3 600 véhicules par heure, 1060 véhicules du nord et du sud et 500 d'est et 940 d'ouest.

3.3.1.1.2 Méthodologie d'évaluation du réseau neuronal : L'objectif est d'évaluer l'efficacité de NEAT pour optimiser les feux de circulation. Le réseau de neurones est jugé à la fin de la simulation sur la base du temps total nécessaire pour que tous les véhicules traversent l'intersection et son objective et de minimiser ce temps d'attendre général pour un nombre des véhicules fixe, mais un flou différent.Cette approche permet de juger l'efficacité globale du réseau neuronal pour gérer le trafic de manière optimale.

3.3.1.1.3 Résultats expérimentaux :

Métrique	Valeur (Modèle NEAT)	Valeur (Modèle Conventionnel)
Temps total accumulé	8505	9729

TABLE 3.1 – Comparaison des Temps totaux accumulés

Les résultats montrent que l'algorithme NEAT optimise efficacement la gestion du trafic, réduisant le temps total nécessaire pour que tous les véhicules traversent l'intersection par rapport aux systèmes traditionnels.

— **Avec le modèle NEAT** : 8505 secondes
— **Sans le modèle NEAT** : 9729 secondes

Le modèle NEAT montre une amélioration significative du temps total accumulé, le réduisant de 1224 secondes par rapport à la méthode traditionnelle.

3.3.2 Résultats de l'algorithme PPO

L'algorithme PPO a été testé dans des configurations à agent unique et multi-agents. La performance a été mesurée par le temps d'attente moyen accumulé au fil des épisodes.

3.3.2.1 Approche à agent unique

Dans la configuration à agent unique, le modèle IA a surpassé de manière significative le système de contrôle de trafic traditionnel, réalisant plus de 50% de réduction du temps d'attente moyen accumulé.

3.3.2.1.1 Environnement de simulation un seul agent :

Pour l'algorithme PPO (Proximal Policy Optimization), l'intersection de Bordeaux a été simulée avec différentes configurations de flux de trafic. On changeait le flux pour éviter que notre agent ait un sur-apprentissage. Le nombre de voitures par heure est :

- Du 0 à 25 000 secondes
Sud nord : 850 chacun, Est : 300 , ouest : 450
- Peut 25 000 à 50 000 secondes
Sud nord : 500 chacun, Est : 350 , ouest : 500
- Du 50 000a 75 000 seconde
Sud nord : 850 chacun, Est : 300 , ouest : 450
- Peut 75 000 a 100 000 secondes
Sud nord : 500 chacun , Est : 350 , ouest : 500
et ça pendant 100 000 secondes sur le simulateur.

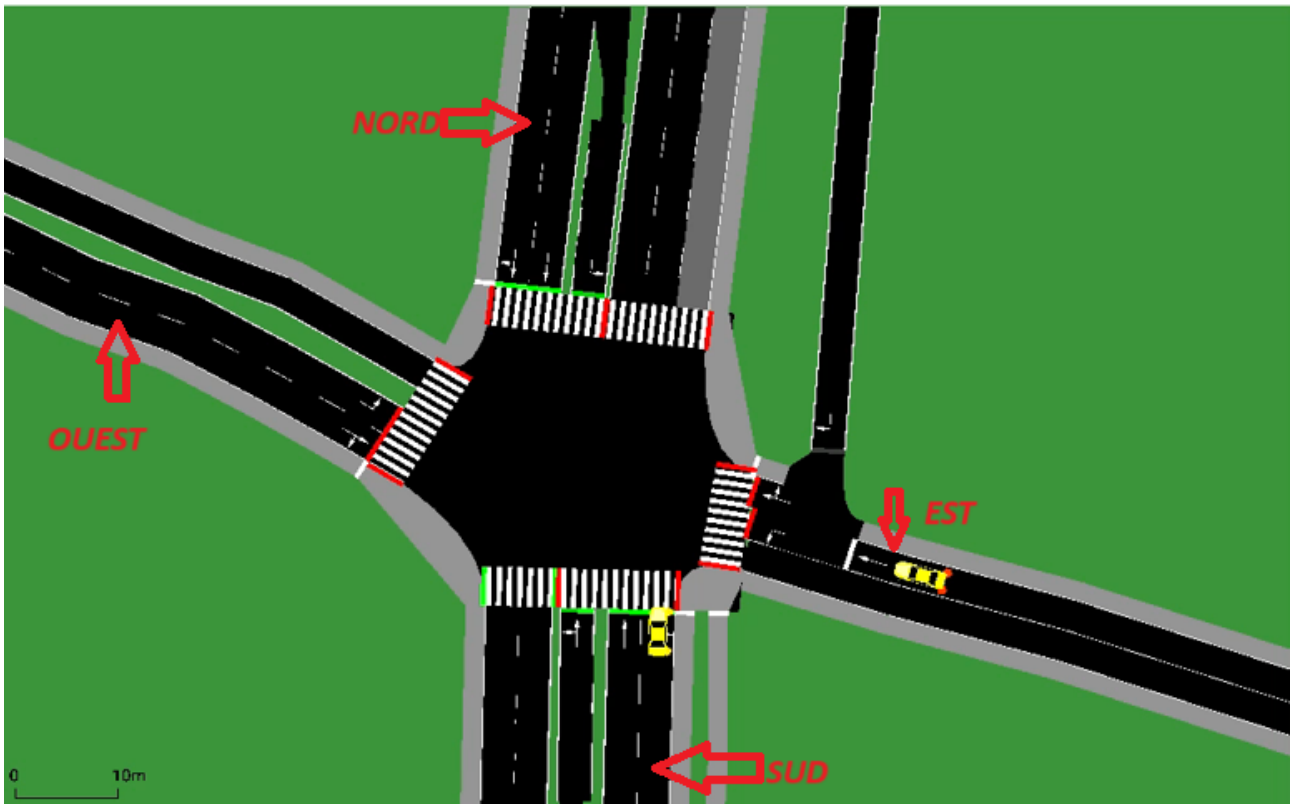

FIGURE 3.1 – Schéma de l'intersection de Bordeau

3.3.2.1.2 Méthodologie de calcul du temps d'attente :

Pour chaque étape de la simulation, le temps d'attente cumulé des véhicules dans les files d'attente a été calculé en utilisant les fonctions de SUMO. Voici comment ce fut :

Récupération des voies contrôlées : Les identifiants des voies contrôlées par le feu de circulation sont obtenus.

Boucle sur les voies : Pour chaque voie, les identifiants des véhicules présents dans la voie lors de la dernière étape de simulation sont récupérés.

Boucle sur les véhicules : Pour chaque véhicule, le temps d'attente actuel est récupéré.

Mise à jour du temps d'attente cumulé : Ce temps d'attente est ajouté au temps d'attente cumulé.

L'objectif est d'évaluer l'efficacité de PPO pour optimiser les feux de circulation. Plusieurs scénarios de charge de route ont été testés :

Charge Nord-Sud : Les véhicules circulent uniquement du nord au sud :

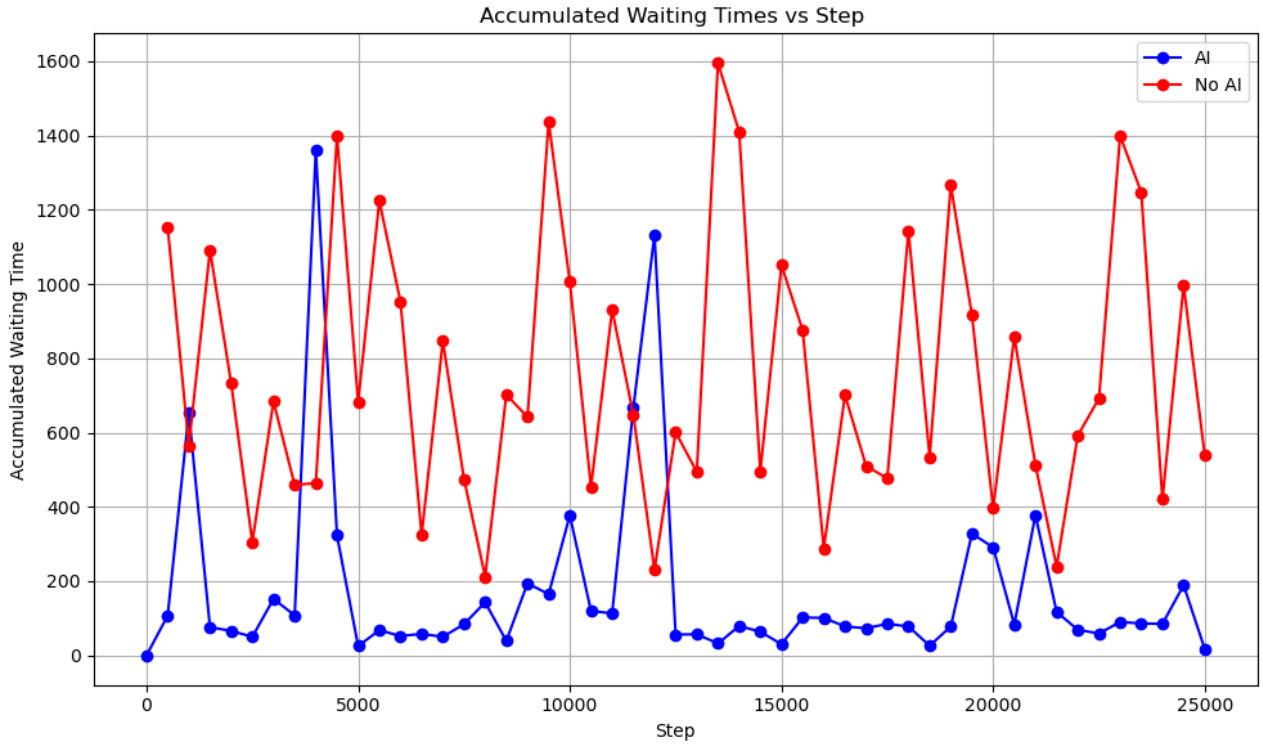

FIGURE 3.2 – Résultats de la simulation avec charge Nord-Sud

Charge Est-Ouest : Les véhicules circulent uniquement d'est en ouest :

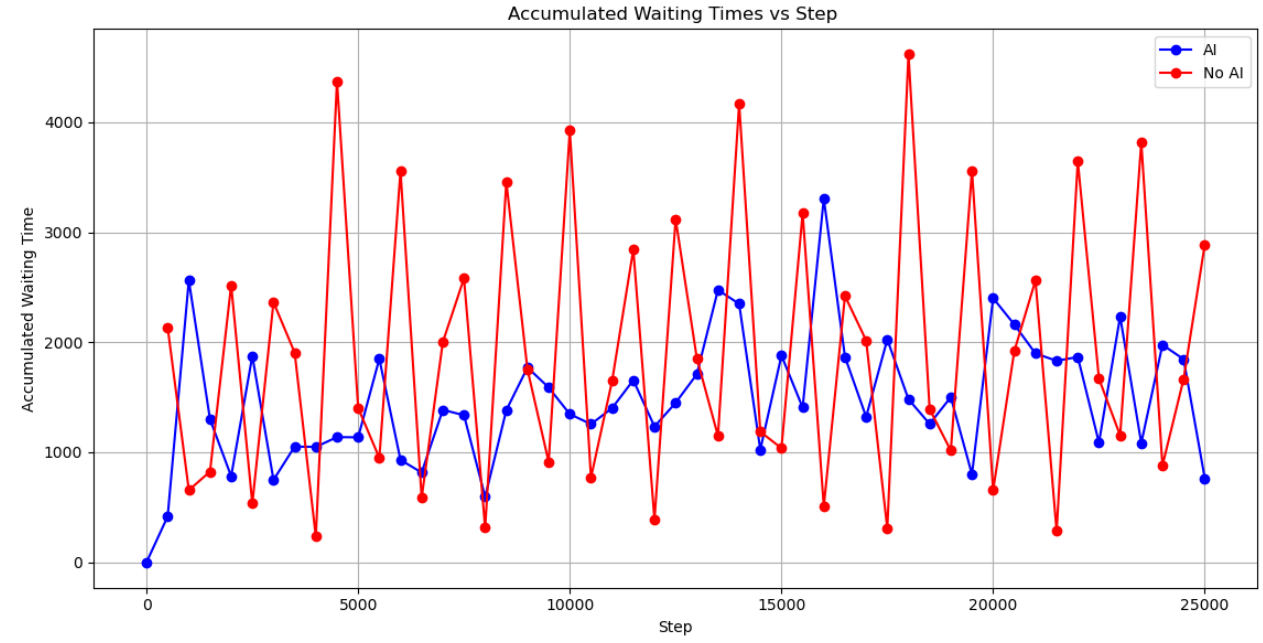

FIGURE 3.3 – Résultats de la simulation avec charge Est-Ouest

Véhicules uniquement du Sud et Nord : Les véhicules entrent et sortent uniquement par le sud et le nord :

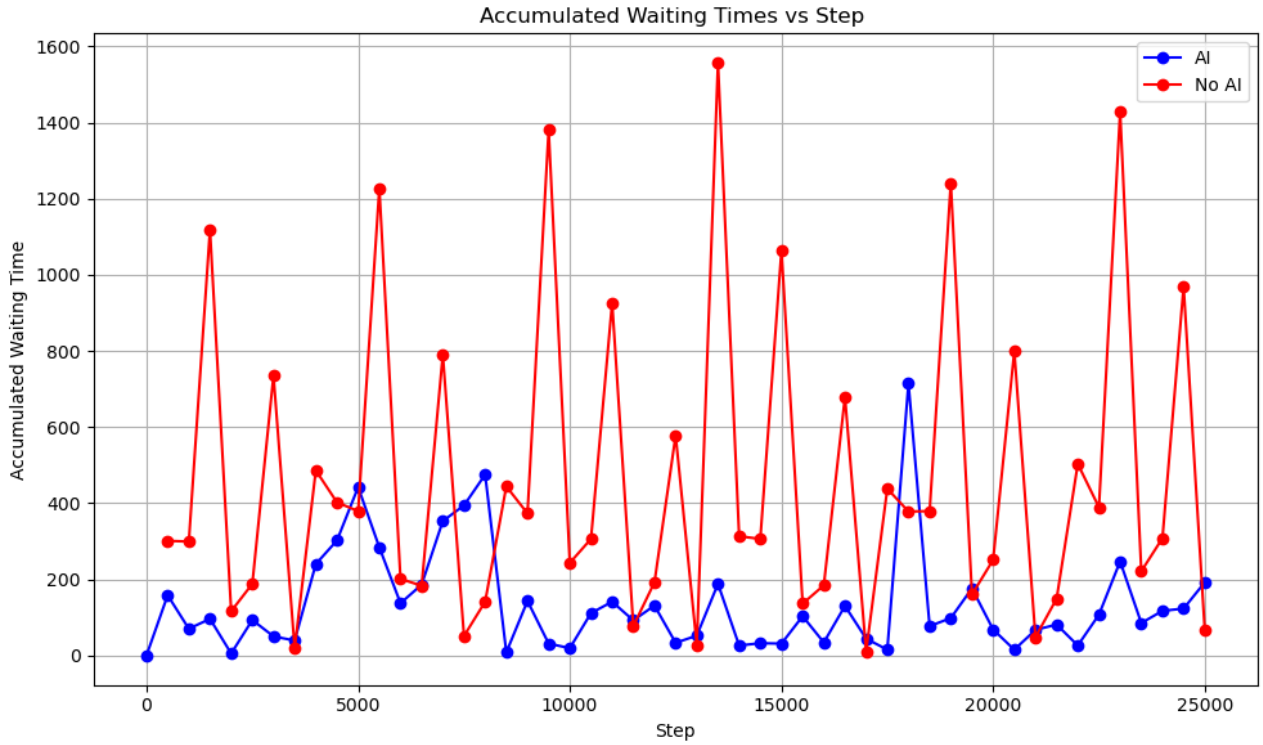

FIGURE 3.4 – Résultats de la simulation avec véhicules uniquement du Sud et du Nord

Véhicules uniquement de l'Est et de l'Ouest : Les véhicules entrent et sortent uniquement par l'est et l'ouest :

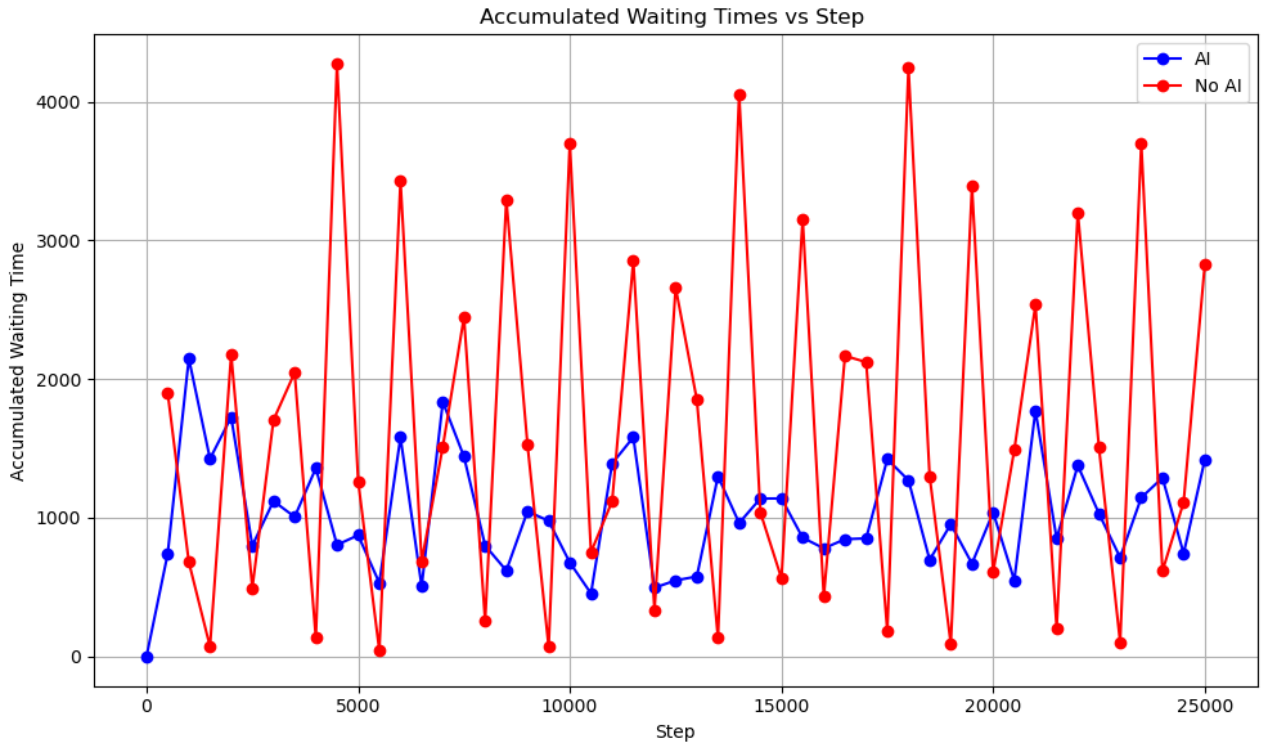

FIGURE 3.5 – Résultats de la simulation avec véhicules uniquement de l'Est et de l'Ouest

Résultats expérimentaux : Les résultats montrent que l'utilisation de PPO permet de réduire significativement le temps total accumulé et le temps d'attente moyen des véhicules comparé

aux systèmes traditionnels de gestion du trafic tout en respectant le temps maximal de feu rouge et de feu vert Comme le montre le tableau suivant :

sud_nord_to_all_g	sud_nord_to_all_r	nord_to_est_g	nord_to_est_r	est_ouest_to_all_g	est_ouest_to_all_r
8	0	0	8	0	11
8	13	0	21	13	0
0	13	0	29	13	0
13	0	0	13	0	34
0	13	0	13	0	47
0	13	0	13	0	21
0	13	0	8	0	16
8	26	0	0	0	13
0	34	0	8	8	0
8	13	0	8	0	21
8	0	8	0	0	8
8	0	0	8	0	18
0	13	0	16	0	8
8	13	0	0	8	0
0	13	8	0	0	8
18	13	0	0	8	0
0	13	0	13	0	13
8	13	0	0	0	13
13	0	0	26	0	26
0	0	13	39	13	0
0	0	0	13	0	50

TABLE 3.2 – tableaux avec le temps passé par chaque feu

Comment pouvez-vous vérifier que le modèle respecte à la fois le temps maximal de 50 secondes pour le feu vert à l'intersection de Bourdeaux indiqué par Gertrude, ainsi que le temps maximal de 60 secondes pour le feu rouge, tout en maintenant le temps minimal de 5 secondes pour le feu vert.

3.3.2.2 Approche multi-agents

3.3.2.2.1 Méthodologie de simulation :

L'intersection de 3x3 gril (9 carrefours) a été simulée avec différentes configurations de flux de trafic de 1800 véhicules par heure avec des trajets aléatoires.

À chaque étape de la simulation, le temps d'attente total des véhicules dans les files d'attente a été calculé en utilisant des fonctions similaires à la fonction d'un seul agent, pour chaque agent. Chaque agent utilise ses propres données et son propre laps de temps dans ce processus.

Les performances du modèle PPO multi-agents ont été évaluées en utilisant cette même mesure. Les résultats sont représentés graphiquement comme suit :

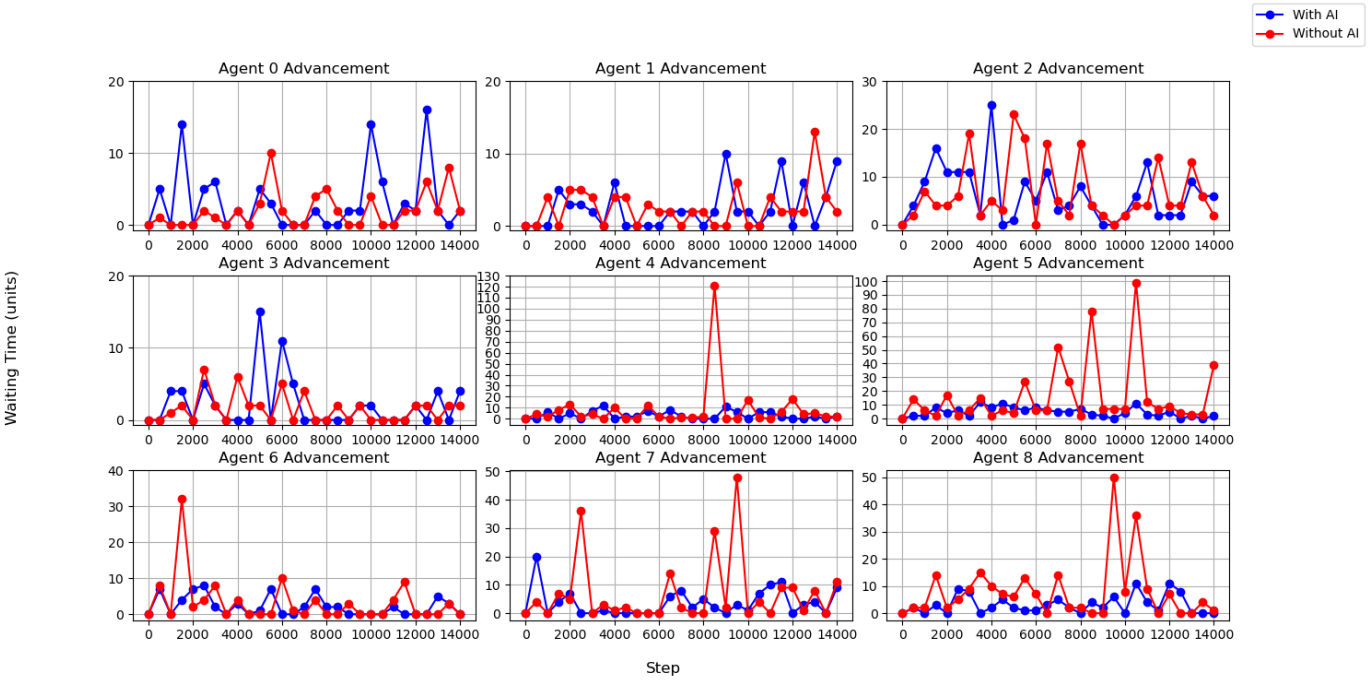

FIGURE 3.6 – Résultats Multi-Agents PPO

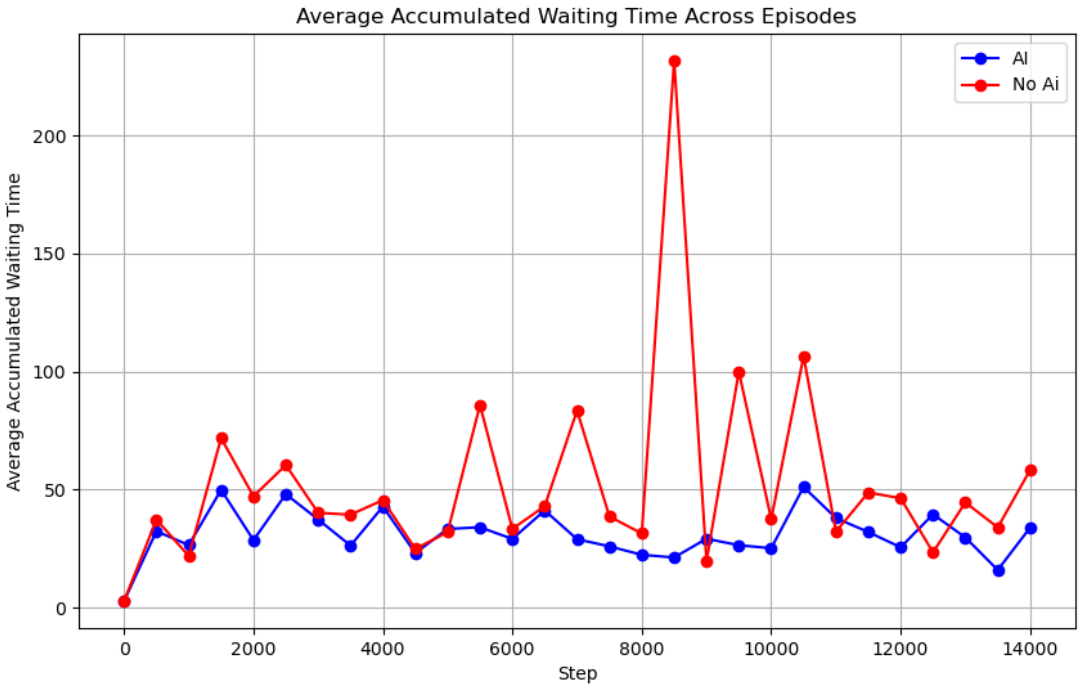

FIGURE 3.7 – Temps d'Attente Moyen Accumulé Multi-Agents PPO

La comparaison démontre clairement que l'approche multi-agents utilisant PPO offre une solution de contrôle de trafic plus efficace, comme en témoigne la réduction des temps d'attente pour tous les agents.

Métrique	Valeur (Modèle PPO - IA)	Valeur (Modèle PPO - Sans IA)
Max Temps d'Attente Moyen Accumulé	50	225

TABLE 3.3 – Comparaison des temps d'Attente Moyens Accumulés

3.4 Analyse détaillée

L'analyse détaillée des résultats révèle les points suivants :

- **Algorithme NEAT** : Le modèle NEAT montre un avantage clair par rapport aux méthodes de contrôle du trafic conventionnelles, réduisant de manière significative le temps total accumulé pour les véhicules. Cela indique l'efficacité du modèle dans l'optimisation du flux de trafic et la réduction de la congestion.
- **Algorithme PPO** :
  - **Approche à Agent Unique** : Le modèle PPO à agent unique réduit efficacement les temps d'attente, montrant son potentiel pour des systèmes de trafic plus simples.
  - **Approche Multi-Agents** : Le modèle PPO multi-agents excelle dans les scénarios plus complexes, démontrant une performance et une évolutivité améliorées par rapport aux méthodes conventionnelles et au PPO à agent unique.

3.5 Discussion

Les résultats indiquent que le DRL, en particulier avec des configurations multi-agents, peut considérablement améliorer les systèmes de gestion du trafic.

Les principaux défis rencontrés incluent la complexité computationnelle de l'entraînement des modèles et les temps de simulation étendus requis et les bibliothèques utilisées, car la bibliothèque comme pantrzoorl et sumorl exige que tous les carrefours soient les mêmes, alors que dans notre vie quotidienne, tous les carrefours ne se ressemblent pas. Néanmoins, les améliorations du flux de trafic et des temps d'attente soulignent le potentiel de ces approches basées sur l'IA.

3.6 Conclusion

Cette étude démontre l'efficacité du DRL et de NEAT dans l'optimisation du contrôle des feux de circulation. L'approche PPO multi-agents, en particulier, offre des avantages substantiels dans des environnements de trafic complexes. Les recherches futures devraient explorer des techniques de coordination avancées entre les agents et des essais de mise en œuvre dans le monde réel pour valider davantage ces conclusions.

Conclusion générale

L'étude présentée dans ce mémoire a exploré diverses méthodes pour optimiser la gestion du trafic urbain en utilisant des algorithmes d'intelligence artificielle, notamment le NEAT (NeuroEvolution of Augmenting Topologies) et le PPO (Proximal Policy Optimization). Les résultats expérimentaux ont démontré l'efficacité de ces approches dans la réduction de la congestion et l'amélioration de l'efficacité globale du transport urbain. L'utilisation de l'environnement de simulation SUMO (Simulation of Urban MObility) a permis de créer des scénarios réalistes pour tester et affiner les stratégies développées.

En conclusion, les algorithmes basés sur l'apprentissage par renforcement ont montré un potentiel significatif pour la gestion intelligente du trafic. En particulier, l'approche multi-agent a permis de coordonner plusieurs agents pour optimiser les phases des feux de circulation de manière plus efficiente par rapport aux approches traditionnelles. Les résultats obtenus montrent que ces techniques peuvent contribuer à des systèmes de transport plus fluides et efficaces, réduisant ainsi les temps d'attente et les émissions de gaz à effet de serre.

Perspective

L'une des perspectives les plus prometteuses de cette recherche est la création de notre propre bibliothèque stable pour la simulation et l'optimisation du trafic urbain. En développant une bibliothèque comme SUMO-RL, nous pouvons garantir que les observations et les paramètres de simulation sont cohérents et reproductibles. Cela facilitera non seulement la comparaison des différentes approches, mais permettra également d'implémenter et de tester de nouvelles stratégies de gestion du trafic avec plus de facilité et de précision.

Une autre perspective intéressante serait d'explorer l'intégration de technologies émergentes telles que les véhicules autonomes et les systèmes de transport intelligents. Ces technologies peuvent bénéficier de l'apprentissage par renforcement pour prendre des décisions en temps réel, améliorant ainsi la fluidité du trafic et la sécurité routière.

Enfin, il serait pertinent d'élargir le cadre de cette recherche en collaboration avec les municipalités et les planificateurs urbains pour mettre en œuvre ces solutions dans des environnements réels. Des études de cas pratiques et des projets pilotes pourraient fournir des données précieuses pour affiner davantage les algorithmes et adapter les solutions aux besoins spécifiques des différentes villes.

En résumé, la recherche présentée dans ce mémoire ouvre la voie à de nombreuses applications et améliorations potentielles dans le domaine de la gestion du trafic urbain. Avec la continuité des efforts de recherche et développement, les approches basées sur l'intelligence artificielle ont le potentiel de transformer significativement la manière dont nous gérons et optimisons les réseaux de transport urbain.

Bibliographie

[1] Fatima Zahra Fagroud, El Habib Benlahmar. Vers un Feu Tricolore Intelligent basé sur l'Internet des objets et le traitement d'images. Colloque sur les Objets et systèmes Connectés - COC'2021, IUT d'Aix-Marseille, Mar 2021, MARSEILLE, France. ffhal-03593725

[2] W.-H. Lin and C. Wang, "An enhanced 0-1 mixed-integer LP formulation for traffic signal control," IEEE Transactions on Intelligent transporta- tion systems (TITS), vol. 5, no. 4, pp. 238–245, 2004.

[3] L. P. J. Rani, M. K. Kumar, K. Naresh, and S. Vignesh, "Dynamic traffic management system using infrared (IR) and Internet of Things (IoT)," in IEEE International Conference on Science Technology Engineering Management (ICONSTEM), pp. 353–357, 2017.

[4] S. Panichpapiboon and P. Leakkaw, "Traffic density estimation : A mobile sensing approach," IEEE Communications Magazine, vol. 55, no. 12, pp. 126–131, 2017.

[5] M. Zhu, X.-Y. Liu, and X. Wang, "Deep reinforcement learning for un- manned aerial vehicle-assisted vehicular networks," IEEE Transactions on Intelligent Transportation Systems (TITS), submitted, p. 15, 2021.

[6] Z. Ma, M. Xiao, Y. Xiao, Z. Pang, H. V. Poor, and B. Vucetic, "High- reliability and low-latency wireless communication for internet of things : challenges, fundamentals, and enabling technologies," IEEE Internet of Things Journal (IOTJ), vol. 6, no. 5, pp. 7946–7970, 2019.

[7] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city internet of things," IEEE Internet of Things Journal (IOTJ), vol. 7, no. 9, pp. 8099– 8110, 2020. [8] P. Koonce and L. Rodegerdts, "Traffic signal timing manual.," tech. rep., United States. Federal Highway Administration, 2008.

[9] M. Hofri and K. W. Ross, "On the optimal control of two queues with server setup times and its analysis," SIAM Journal on Computing, vol. 16, no. 2, pp. 399–420, 1987.

[10] H. Wang, P. Hu, P. Wang, and H. Wang, "A genetic timing scheduling model for urban traffic signal control," Elsevier Information Sciences, 2021.

[11] B. "Brian" Park, I. Yun, and K. Ahn, "Stochastic optimization for sustainable traffic signal control," Taylor Francis International journal of sustainable transportation, vol. 3, no. 4, pp. 263–284, 2009.

[12] M. E. M. ALI, A. DURDU, S. A. CELTEK, and A. Yilmaz, "An adaptive method for traffic signal control based on fuzzy logic with webster and modified webster formula using sumo traffic simulator," IEEE Access, 2021.

[13] R. Hawi, G. Okeyo, and M. Kimwele, "Smart traffic light control using fuzzy logic and wireless sensor network," in IEEE Computing Conference, pp. 450–460, 2017.

[14] R. P. Roess, E. S. Prassas, and W. R. McShane, Traffic engineering. Pearson/Prentice Hall, 2004.

[15] J. D. Little, M. D. Kelson, and N. H. Gartner, "Maxband : A versatile program for setting signals on arteries and triangular networks," 1981.

[16] S.-B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights : A realistic simulation," in Springer Advances in applied selforganizing systems, pp. 45–55, 2013.

[17] P. Varaiya, "The max-pressure controller for arbitrary networks of signalized intersections," in Springer Advances in dynamic network modeling in complex transportation systems,

pp. 27–66, 2013.

[19] P. Lowrie, "Scats-a traffic responsive method of controlling urban traffic," Sales information brochure published by Roads Traffic Authority, Sydney, Australia, 1990.

[20] P. W. Shaikh, M. El-Abd, M. Khanafer, and K. Gao, "A review on swarm intelligence and evolutionary algorithms for solving the traffic signal control problem," IEEE transactions on intelligent transportation systems (TITS), vol. 23, no. 1, pp. 48–63, 2020.

[21] A. M. Turky, M. S. Ahmad, and M. Z. M. Yusoff, "The use of genetic algorithm for traffic light and pedestrian crossing control," International Journal of Computer Science and Network Security, vol. 9, no. 2, pp. 88– 96, 2009.

[22] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight : A reinforcement learning approach for intelligent traffic light control," in ACM Interna- tional Conference on Knowledge Discovery , Data Mining (SIGKDD), pp. 2496–2505, 2018.

[23] T. Chu, J. Wang, L. Codeca', and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," IEEE Transactions on Intelligent Transportation Systems (TITS), vol. 21, no. 3, pp. 1086–1095, 2019.

[24] P. Balaji, X. German, and D. Srinivasan, "Urban traffic signal control using reinforcement learning agents," IET Intelligent Transport Systems, vol. 4, no. 3, pp. 177–188, 2010.

[25] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning- based multi-agent system for network traffic signal control," IET Intel- ligent Transport Systems, vol. 4, no. 2, pp. 128–135, 2010.

[26] P. Mannion, J. Duggan, and E. Howley, "An experimental review of reinforcement learning algorithms for adaptive traffic signal control," Springer Autonomic road transport support systems, pp. 47–66, 2016.

[27] B. Bakker, S. Whiteson, L. Kester, and F. C. Groen, "Traffic light control by multiagent reinforcement learning systems," in Springer Interactive Collaborative Information Systems, pp. 475–510, 2010.

[28] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," IET Intelligent Transport Systems, vol. 11, no. 7, pp. 417–423, 2017.

[29] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight : A reinforcement learning approach for intelligent traffic light control," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, pp. 2496–2505, ACM, 2018.

[30] M. Aslani, M. S. Mesgari, and M. Wiering, "Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events," Elsevier Transportation Research Part C : Emerging Technologies, vol. 85, pp. 732–752, 2017.

[31] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Traffic light control in non-stationary environments based on multi agent q-learning," in IEEE conference on intelligent transportation systems (ITSC), pp. 1580–1585, 2011.

[32] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," Advances in Neural Information Processing Systems (NeurIPS), vol. 1, 2016.

[33] X. Zang, H. Yao, G. Zheng, N. Xu, K. Xu, and Z. Li, "Metalight : Value-based meta-reinforcement learning for traffic signal control," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 1153–1160, 2020.

[34] C. Chen, H. Wei, N. Xu, G. Zheng, M. Yang, Y. Xiong, K. Xu, and Z. Li, "Toward a thousand lights : Decentralized deep reinforcement learning for large-scale traffic signal control," in Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), vol. 34, pp. 3414–3421, 2020.

[35] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," IEEE Transactions on Vehicular Technology (TVT), vol. 68, no. 2, pp. 1243–1253, 2019.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, p. 529, 2015.

[37] K. Zhang, Z. Yang, and T. Bas¸ar, "Multi-agent reinforcement learning : A selective overview of theories and algorithms," Springer Handbook of Reinforcement Learning and Control, pp. 321–384, 2021.

[38] T. Wu, P. Zhou, K. Liu, Y. Yuan, and D. O. Wu, "Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks," IEEE Transactions on Vehicular Technology (TVT), vol. PP, no. 99, pp. 1–1, 2020.

[39] W. Liu, J. Liu, J. Peng, and Z. Zhu, "Cooperative multi-agent traffic signal control system using fast gradient-descent function approximation for V2I networks," in IEEE International Conference on Communications (ICC), pp. 2562–2567, 2014.

[40] M. Xu, K. An, L. H. Vu, Z. Ye, J. Feng, and E. Chen, "Optimizing multi-agent based urban traffic signal control system," Taylor Francis Journal of Intelligent Transportation Systems, vol. 23, no. 4, pp. 357– 369, 2019.

[41] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks : Lstm cells and network architectures," Neural computation, vol. 31, no. 7, pp. 1235–1270, 2019.

[42] Google. (s. d.). Greenlight : Accélérer le passage au vert pour réduire les émissions. Récupéré sur https ://sites.research.google/greenlight/

[43] Sutton, R. S., Barto, A. G. (2018). Reinforcement learning : An introduction. MIT press.

[44] Kaelbling, L. P., Littman, M. L., Moore, A. W. (1996). Reinforcement learning : A survey. Journal of artificial intelligence research, 4, 237-285.[3]

[45] Haykin, S. (2004). Neural networks : a comprehensive foundation. Prentice Hall PTR.

[46] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[47] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

ملخص

يقدم هذا البحث دراسة متعمقة حول تطوير نظام مبتكر للتحكم في إشارات المرور للتقاطعات الحضرية باستخدام تقنيات التعلم المعزز. الهدف الرئيسي هو تصميم نظام ذكي ومرن قادر على تحسين إدارة حركة المرور، مما يعزز انسيابية وسلامة الطرق. بالاعتماد على نهج قائم على الذكاء الاصطناعي، يهدف هذا المشروع إلى اقتراح حلول فعالة وقابلة للتكرار لمشاكل حركة المرور الحضرية. تشمل الآفاق المستقبلية دمج التقنيات الناشئة مثل المركبات الذاتية وأنظمة النقل الذكية، بالإضافة إلى التعاون مع السلطات المحلية لتنفيذ عملي لهذه الحلول

الكلمات المفتاحية: التحكم في إشارات المرور، الذكاء الاصطناعي، التعلم العميق بالتعزيز، حركة المرور الحضرية، التحسين، المحاكاة، ، إدارة حركة المرور، التقاطعات الطرقية,PPO,NEAT, SUMO.

Résumé

Ce mémoire présente une étude approfondie sur le développement d'un système novateur de contrôle des feux de circulation pour les intersections urbaines en utilisant des techniques d'apprentissage par renforcement. L'objectif principal est de concevoir un système intelligent et résilient capable d'optimiser la gestion du trafic, améliorant ainsi la fluidité et la sécurité routière. En s'appuyant sur une approche basée sur l'intelligence artificielle, ce projet vise à proposer des solutions efficaces et reproductibles pour les problèmes de circulation urbaine. Les perspectives futures incluent l'intégration de technologies émergentes telles que les véhicules autonomes et les systèmes de transport intelligents, ainsi que la collaboration avec les autorités locales pour une mise en œuvre pratique de ces solutions.

Mots-clés : Contrôle des feux de circulation, Intelligence artificielle, Apprentissage par renforcement profond (DRL), Trafic urbain, Optimisation, Simulation, Systèmes de transport intelligents, Gestion du trafic, PPO, NEAT, SUMO.

Abstract

This thesis presents an in-depth study on the development of an innovative traffic light control system for urban intersections using reinforcement learning techniques. The main objective is to design an intelligent and resilient system capable of optimizing traffic management, thereby improving road fluidity and safety. By leveraging an artificial intelligence-based approach, this project aims to propose efficient and reproducible solutions for urban traffic problems. Future perspectives include the integration of emerging technologies such as autonomous vehicles and intelligent transport systems, as well as collaboration with local authorities for practical implementation of these solutions.

Keywords: Traffic light control, Artificial intelligence, Deep reinforcement learning, Urban traffic, Optimization, Simulation, Intelligent transport systems, Autonomous vehicles, Traffic management, Road intersections, PPO, NEAT, SUMO.