

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Outils mis à disposition - CIL4Sys Softeam</b>	<b>3</b>
2.1	Outil de simulation Sim4Sys . . . . .	3
2.2	Données à notre dispositions : TOMTOM et les taxis de Los Angeles . . . . .	3
<b>3</b>	<b>État de l'art : contrôle du trafic routier urbain</b>	<b>3</b>
3.0.1	Quantification du niveau des émissions des véhicules .	5
3.0.2	Contrôle de la vitesse des véhicules . . . . .	6
3.1	Contrôle de la signalisation au niveau des intersections . . . .	6
3.2	Problème d'optimisation . . . . .	8
3.3	Apprentissage par renforcement . . . . .	9
3.4	Exploration vs exploitation . . . . .	12
3.5	Modélisation de simulation . . . . .	12
3.6	Simulateur Sumo . . . . .	13
3.7	Framework Flow . . . . .	13
<b>4</b>	<b>Cas pratiques</b>	<b>13</b>
4.1	Cartpole . . . . .	13
4.2	Deep Q-learning pour une intersection . . . . .	17
4.2.1	Etats $\mathcal{E}_t$ . . . . .	18
4.2.2	Actions $\mathcal{A}_t$ . . . . .	18
4.2.3	Récompenses $r_t$ . . . . .	18
4.2.4	Architecture du DQN . . . . .	19
<b>5</b>	<b>Planning envisagé pour la suite du projet</b>	<b>20</b>
<b>6</b>	<b>Conclusion</b>	<b>21</b>
% Define some commands to keep the formatting separated from the content 1} 1} 1} 1} 1}		

## 1 Introduction

Alors que l'écologie devient un enjeu international d'importance, de plus en plus de recherches sont faites sur des transports verts, ou sur des moyens d'économiser de l'énergie. Dans le même temps, les avancées technologiques

laissent présager une arrivée de véhicules autonomes capables de communiquer avec leur environnement, ou du moins de l'intelligence artificielle comme assistance de conduite. Dans ce contexte, plusieurs études ont été menées dans le but de réduire les émissions de polluants des véhicules par l'intermédiaire de la fluidification du trafic routier. Parmi ce domaine de recherche, on distingue notamment les travaux se concentrant sur les zones extra-urbaine (autoroutes et grands axes routiers) [citer papiers sans commenter], des zones urbaines qui sont au cœur du sujet traité dans ce rapport. La start-up CIL4SYS nous propose, dans le cadre d'un projet fil-rouge, de nous pencher sur la possibilité de réduire les émissions de polluants dans l'hypothèse de feux tricolores intelligents capables d'envoyer des consignes de vitesse aux véhicules traversants un quartier en zone urbaine dans le but de limiter les accélérations et décélérations des véhicules sources de fortes consommations en carburant.

Le projet s'articule donc autour de CIL4SYS créée en juin 2015 par Philippe GICQUEL qui a travaillé pendant 25 ans en R\&D automobile chez EDAG et PSA Peugeot-Citroën. Le cœur de métier de CIL4SYS étant de proposer une solution permettant de générer des cahiers des charges et des spécifications techniques à partir diagrammes UML décrivant le fonctionnement désiré d'un système. Leur cible principale étant l'industrie automobile, un environnement de simulation a été spécifiquement développé dans le but de répondre à cette industrie. Mais aussi de SOFTEAM, grand groupe fournissant un panel de service divers, allant du consulting opérationnel et métier, à la modélisation et l'automatisation, en passant par l'innovation. Leur rôle, durant ce projet, sera de nous assister lors de la phase de machine-learning en nous apportant une expertise avancée en data science. Enfin TélécomParisTech constitue le dernier acteur de ce projet par l'intermédiaire de notre groupe d'étudiants chargé d'effectuer les premiers développements et essais ainsi que de proposer les solutions et méthodes à explorer.

L'objectif principal de ce projet est donc de démontrer à l'échelle d'un quartier que par un contrôle à la fois des feux tricolores et des vitesses des véhicules (dans un contexte où les véhicules sont communicants et peuvent recevoir des instructions de vitesse), il est possible de trouver un optimum de diminution des rejets de polluants et de CO<sub>2</sub>. Les méthodes mises en place au cours du projet devront faire appel à nos connaissances en apprentissage automatique afin de proposer une première approche de résolution.

Il nous est demandé de d'utiliser nos connaissances pour proposer une solution de machine learning à ce problème.

## **2 Outils mis à disposition - CIL4Sys Softeam**

### **2.1 Outil de simulation Sim4Sys**

Cil4sys a mis en place un logiciel de simulation, permettant de valider leur modèles grâce à une visualisation 3D. Ce simulateur peut être exporté vers un maquette réelle pour tester les résultats d'un algorithme. Elle permet donc un potentiel passage en production de solutions résultants d'algorithmes développé sur ce logiciel.

Un des enjeux du projet sera donc d'intégrer notre solution à Sim4Sys. Pour ce faire, nous allons écrire un script Python qui prendra en entrée un état du simulateur  $e \in \mathcal{E}$  et retournera une action  $a \in \mathcal{A}$  à réaliser. L'état est une description du système que nous détaillons dans le reste de ce document, et l'action à prendre peut être un changement d'état des feux et/ou une recommandation de vitesse pour les voitures. La figure~1 schématise l'interaction de notre script avec le simulateur Sim4Sys.

### **2.2 Données à notre dispositions : TOMTOM et les taxis de Los Angeles**

Grâce à Cil4Sys, nous avons accès à l'API de TOMTOM. Cela nous permet d'extraire des informations relatives à des vitesses observées réellement sur le terrain sur une zone que l'on peut définir.

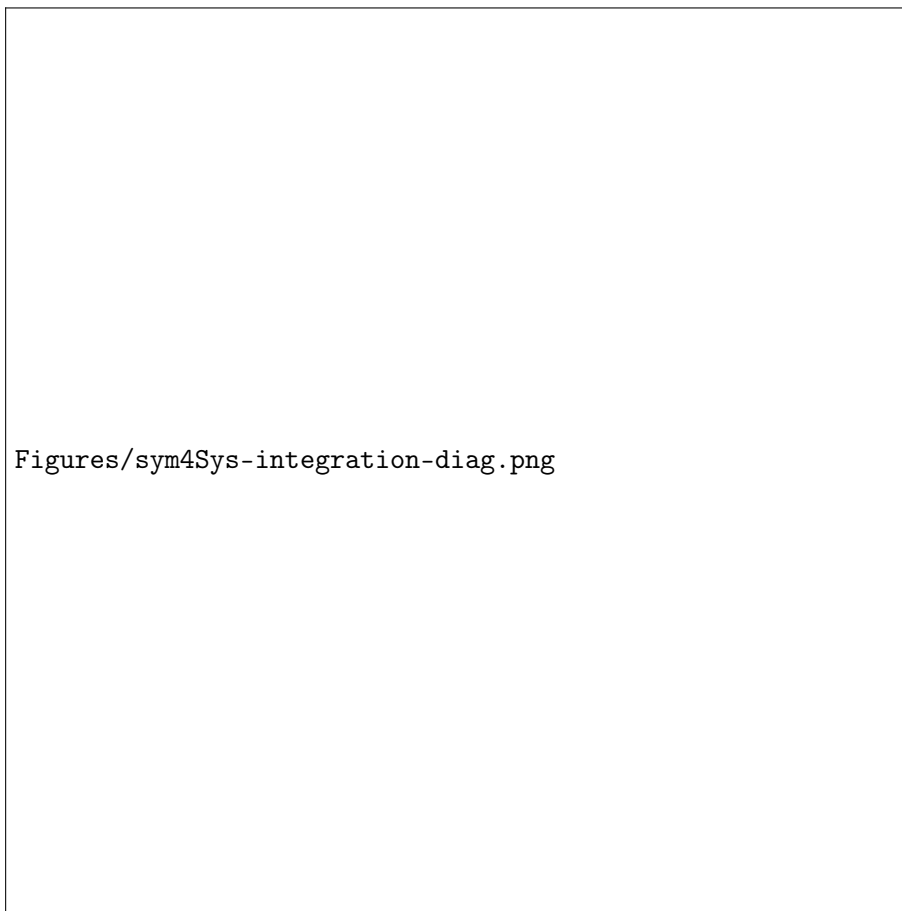
L'obtention de grandeurs réelles nous servira de calibrage et de validation de modèle quant au réalisme des données que nous allons manipuler.

Lors de nos recherches sur le sujet et souhaitant multiplier les informations qui pourraient nous servir par la suite, nous avons récupéré des données précises sur la circulation à Los Angeles des taxis. Ces éléments spécifiques concernant les accélérations, les vitesses, les émissions de CO2 pourraient nous servir dans le cas où nos premières solutions ne sont pas concluantes.

## **3 État de l'art : contrôle du trafic routier urbain**

Notre étude de l'état de l'art porte sur les trois axes induits par le sujet du projet, à savoir la réduction des émissions de polluants, le contrôle de la vitesse des véhicules et le contrôle de la signalisation tricolore.

Le contrôle du trafic routier urbain fait l'objet de nombreuses études : la décongestion du trafic, qu'il soit urbain ou sur les grands axes représente



Figures/sym4Sys-integration-diag.png

Figure 1: Interaction de notre solution a l'outil de simulation Sim4Sys

une problématique importante tant d’un point de vue environnemental qu’en termes d’urbanisme ou de sécurité routière.

Les études les plus anciennes sur le sujet proposent une optimisation basée sur l’historique de trafic et visent à prédire le trafic en utilisant une approche de type série temporelle. Nous avons focalisé nos recherches sur le trafic urbain, et non sur des grands axes routiers, ce qui nous amène à l’étude des intersections et à la gestion de la signalisation.

### 3.0.1 Quantification du niveau des émissions des véhicules

Afin de répondre à la problématique de la réduction des émissions de polluants, il convient dans un premier lieu de les quantifier. Ils dépendent à priori de la vitesse et de l’accélération des véhicules, mais également de leur type, de leur âge.

Selon les données utilisées dans le modèle, on peut utiliser les vitesses moyennes pour estimer les émissions ou un modèle dynamique qui prendra en compte les accélérations et arrêts par exemple.

On trouve également une modification à la fonction de coût classique qui intègre les temps de trajets et d’arrêt, qui y intègre la consommation de carburant que l’on peut considérer comme une approximation acceptable du niveau d’émissions dans un premier temps.

La recherche concernant la modélisation des niveaux d’émissions a un large historique et propose des solutions diverses, par exemple modélisation par fonction linéaire du nombre d’arrêts [?] et du temps total de trajet des véhicules, ou des modèles non linéaires, réseaux de neurones [?].

Un modèle répandu est le VT-Micro [?], développé par le Virginia Tech Transportation Institute. Le modèle propose une estimation granulaire des émissions, à la fois selon des variables catégorielles et selon des variables concernant le comportement réel d’un véhicule. Les variables catégorielles concernant le véhicule (le type, le carburant utilisé, l’année de construction) dont la classification a été effectuée avec des arbres de régression, selon les quantités émises de différents polluants (CO<sub>2</sub>, NO, HC) mais aussi le mode de conduite. On distingue plusieurs modes de conduite : autoroute, route, ville, par exemple.

Enfin, le modèle utilise des variables basées sur la conduite réelle et en particulier la vitesse instantanée et l’accélération du véhicule. Le modèle de régression employé dépend de termes d’accélération et de vitesse linéaires, quadratiques et cubiques.

On note également que les simulateurs de trafic proposent une estimation des émissions des véhicules.

### 3.0.2 Contrôle de la vitesse des véhicules

Avec le développement des véhicules autonomes et l'amélioration des moyens de communication inter-véhicules (capteurs), des modèles de contrôle ou suggestion de vitesse des véhicules sont également développés avec l'objectif d'optimiser les intersections.[?] Des études ont également montré qu'un contrôle de la vitesse à l'abord des intersection permet de limiter les arrêts, et ainsi de diminuer les émissions de polluants [?] [?] [?].

Dans le cadre du contrôle de la vitesse des véhicules, on peut considérer soit une intersection seule en définissant une distance à partir de laquelle le véhicule entre dans la zone. A partir de ce moment la file de véhicules se situant sur une même voie se comporte en suivant le véhicule de tête, en utilisant l'hypothèse que les véhicules communiquent entre eux afin d'éviter toute collision. Le véhicule de tête quant à lui reçoit ses instructions d'un agent planificateur qui connaît l'état de l'environnement, et envoie ses instructions aux véhicules de tête pour permettre un passage plus fluide.

Dans cette configuration, chaque véhicule qui est dernier dans la file envoie une information de vitesse au nouvel entrant, qui réglera sa vitesse sur celle du précédent. Le mécanisme permet d'éviter des collisions et seuls les véhicules de tête sont concernés par la prise de décision relative à l'intersection.

Le modèle propose également d'étudier un réseau d'intersections. Dans ce cadre le modèle d'intersection isolée est conservé mais étendu. A la sortie de l'intersection le véhicule conserve en mémoire les informations sur ceux qui le suivent. Il communique cette information à l'intersection suivante, ce qui permet d'avoir une estimation sur le nombre de véhicules qui demandent à rentrer dans l'intersection au delà de la zone délimitée précédemment. Cette information est une indication de volume auquel s'attend l'agent régulateur de l'intersection.

### 3.1 Contrôle de la signalisation au niveau des intersections

L'un des points importants à prendre en compte dans l'optimisation de trafic routier est une bonne définition de l'optimum à atteindre. En effet un optimum individuel ne permet pas d'obtenir un équilibre optimal au sens de Pareto [?]. Dans notre cas cela correspondrait au cas où un ou plusieurs véhicules minimisent leur fonction objectif (leur temps de trajet par exemple) et où le reste des véhicules ne progresse pas. Ce point nous amènera à poser des contraintes dans les fonctions objectif utilisées.

Cela se traduit par le développement de modèles multi-agents, où l'on

cherche à optimiser le système au niveau de l'agent intersection. Les modèles multi-agents sont adaptés à la régulation de trafic dans la mesure où l'on ne cherche pas à optimiser une unique intersection, mais un réseau d'intersections reliées les unes aux autres. Le système multi-agent permet de définir des agents intersection, contrôlées par un agent à un niveau supérieur dont l'objectif est de coordonner les intersections entre elles (la région).

Le modèle multi-agent développé par Jin et Ma, 2018 [?] décompose le système comme suit :

- Un agent de trajectoire : cela représente une combinaison possible origine-destination au sein d'une intersection à laquelle est associé un feu tricolore. Tous les agents trajectoire d'une même intersection possèdent donc le même environnement et sont représentés par une matrice de compatibilité basée sur la configuration de l'intersection (pour éviter les collisions). Ces agents possèdent deux états : actif (si la trajectoire peut être effectuée) ou inactif (le feu correspondant est rouge).
- Un agent intersection qui donne des instructions aux agents de trajectoire qui lui sont subordonnés, afin d'assurer une stratégie optimale collectivement.
- Les agents régions sont les plus hauts du système. Ils regroupent des intersections et donnent des instructions aux intersections avec une vision plus large, permettant de générer des scénarios de ligne de feux vert (*green wave scenario*).
- Les agents de même type sont considérés dans ce modèle comme homogènes entre eux, c'est-à-dire qu'ils possèdent les mêmes fonctions de coût ou récompense, le même environnement et les mêmes paramètres d'apprentissage.

L'approche utilisée dans cet article est l'algorithme d'apprentissage par renforcement SARSA (*State Action Reward State Action*) qui à chaque étape actualise la base de connaissances des agents trajectoire (le plus bas niveau) et qui définit le contrôle optimal au niveau de l'intersection.

L'apprentissage par renforcement pour les problèmes d'optimisation de trafic ont fait l'objet d'autres recherches, notamment utilisant le Q-learning au lieu de SARSA [?]. La différence entre les deux algorithmes se situant dans la définition de la récompense : SARSA prend en compte la politique suivie par l'agent quand le Q-learning considère l'action suivante comme maximisant la récompense. Ces deux algorithmes proches présentent chacun

des avantages et inconvénients : quand le Q-learning est plus rapide et simple à mettre en oeuvre, le SARSA permet une approche plus conservatrice, notamment si on souhaite l'utiliser dans un environnement réel et non de simulation.

L'utilisation d'un apprentissage par renforcement nous permettrait une flexibilité sur la définition de la fonction de récompense des agents, et ainsi compléter le modèle original d'une fonction de contrôle sur la vitesse des véhicules, permis par le contrôle situé au niveau de l'intersection mais également d'utiliser une mesure des émissions de polluants dans le modèle.

### 3.2 Problème d'optimisation

Plus classiquement, il est possible de donner une formulation d'un point de vue optimisation du problème. En effet, l'objectif principal demandé par CIL4SYS est de trouver des lois de commande des feux et des véhicules de manière à réduire au maximum les émissions de CO2. Au cours de ce projet les émissions de CO2 seront modélisées uniquement par l'intermédiaire des accélérations. Une première approche peut donc être de minimiser :

$$\min_{\theta(t)} \mathbb{E} \left[ \sum_{j=1}^m \int_{t=0}^{t_j^{sortie}} |a_j(t, \theta(t), \mathcal{C}_j(\xi))| dt \right]$$

où  $a_j(t)$  représente les accélérations subies par le véhicule  $j$  tout au long de sa trajectoire  $\mathcal{C}_j$  affectée aléatoirement par la variable aléatoire  $\xi$  et  $\theta(t)$  représente la loi de commande imposée aux véhicules.

On constate d'ors et déjà que la solution à un tel problème d'optimisation est évidente puisqu'il suffit d'imposer une loi de commande de telle sorte que les véhicules restent à l'arrêt pour minimiser les émissions. Le problème initial d'optimisation se doit donc d'être complété par l'intermédiaire de contraintes afin répondre à un certain réalisme. Une première contrainte à envisager est le temps de trajet pour la réalisation du parcours  $\mathcal{C}$  du véhicule. En utilisant les données TOMTOM, il sera donc possible de déterminer au sein du quartier qui sera choisi par la suite quel est le temps de trajet moyen réalisé par un véhicule pour une trajectoire imposée  $\bar{t}(\mathcal{C})$ . Une contrainte seuil sur le temps de trajet pourra alors être imposé avec pour référence le temps de trajet calculé. Le problème de minimisation se reformule alors de la manière suivante :



$$\begin{cases} \min_{\theta(t)} \mathbb{E} \left[ \sum_{j=1}^m \int_{t=0}^{t_j^{sortie}} |a_j(t, \theta(t), \mathcal{C}(\xi))| dt \right] \\ \text{s.t. } t_j^{sortie} \leq \alpha \cdot \bar{t}(\mathcal{C}) \end{cases} \quad (1)$$

La formalisation du problème d'optimisation reste à compléter et on sera amené à ajouter d'autres contraintes au cours du projet au fur et à mesure de sa prise en main. La fonction de coût à minimiser sera elle aussi amenée à être changée si par exemple le nombre de voitures qui traversent le quartier devient aléatoire lui aussi etc. . .

### 3.3 Apprentissage par renforcement

Comme dit précédemment, une des méthodes utilisée dans la littérature actuelle pour effectuer de la fluidification de trafic est l'apprentissage par renforcement. C'est la méthode d'apprentissage sur laquelle nous avons décidée de nous concentrer car novatrice dans le domaine de l'optimisation de trafic routier. Dans cette section, nous présentons les grandes lignes de cette méthode d'apprentissage.

L'apprentissage par renforcement est une méthode statistique de prise de décision dans un environnement donné. L'environnement est modélisé par un état, et l'acteur peut réaliser une action qui affectera l'état. L'algorithme est guidé lors de la phase d'entraînement par une fonction de récompense. Après avoir pris une action  $a$  à un état  $e$ , l'observation du nouvel état permet le calcul d'une récompense,  $r \in \mathbb{R}$ .

Notons  $\mathcal{E}$  l'ensemble des états possibles de notre environnement,  $\mathcal{A}$  l'ensemble des actions possibles, et  $r_t$  la récompense obtenue au pas de temps  $t$ . En modélisant la récompense cumulative comme un processus de Markov, elle ne dépend que de l'action prise et de l'état du système aux temps  $t \geq t_0$  et est définie de la manière suivante:

$$R_{t_0} = \sum_{t=t_0}^{\infty} \gamma^{t-t_0} r_{t+1}$$

Où le coefficient  $\gamma \in [0, 1]$  permet de donner plus de poids aux récompenses proches de  $t_0$  dans le temps et à la somme de converger.

Le principe de l'apprentissage par renforcement est alors de chercher une fonction  $Q^* : \mathcal{E} \times \mathcal{A} \rightarrow \mathbb{R}$  qui estime le retour cumulé  $R_{t_0}$  pour une action  $a \in \mathcal{A}$  réalisée à un état  $e \in \mathcal{E}$ .

La fonction  $\pi(e, a)$  retourne la probabilité de réaliser une action  $a$  à l'état  $e$ . Nous avons donc  $\sum_{a \in \mathcal{A}} \pi(e, a) = 1$ . Définissons  $Q^\pi(e, a)$ , la fonction qui prédit l'espérance de la récompense cumulative sous  $\pi$  sachant  $e$  et  $a$ :

$$Q^\pi(e, a) = \mathbb{E}_\pi[R_t | e_t = e, a_t = a]$$

Définissons  $\mathcal{P}_{ee'}^a$ , la probabilité de passer d'un état  $e$  à un état  $e'$  sachant un état  $e$  et une action  $a$  donnés:

$$\mathcal{P}_{ee'}^a = \mathbb{P}(e_{t+1} = e' | e_t = e, a_t = a)$$

Définissons aussi  $\mathcal{R}_{ee'}^a$ , l'espérance de la récompense  $r_{t+1}$  sachant un état  $e$ , et une action  $a$  à l'instant  $t$  ainsi qu'un état  $e'$  à l'instant  $t + 1$ :

$$\mathcal{R}_{ee'}^a = \mathbb{E}(r_{t+1} | e_t = e, e_{t+1} = e', a_t = a)$$

Il est alors possible de d'exprimer  $Q^\pi$  de la manière suivante:

$$Q^\pi(e, a) = \mathbb{E}_\pi[R_t | e_t = e, a_t = a] \tag{2}$$

$$= \mathbb{E}_\pi \left[ \sum_{t=t_0}^{\infty} \gamma^{t-t_0} r_t | e_{t_0} = e, a_{t_0} = a \right] \tag{3}$$

$$= \mathbb{E}_\pi \left[ r_{t+1} + \gamma \sum_{t=t_0}^{\infty} \gamma^{t-t_0} r_{t_0+1} | e_{t_0} = e, a_{t_0} = a \right] \tag{4}$$

$$= \sum_{e'} \mathcal{P}_{ee'}^a \left[ \mathcal{R}_{ee'}^a + \gamma \mathbb{E}_\pi \left( \sum_{t=t_0}^{\infty} \gamma^{t-t_0} r_{t_0+1} | e_{t+1} = e' \right) \right] \tag{5}$$

$$= \sum_{e'} \mathcal{P}_{ee'}^a \left[ \mathcal{R}_{ee'}^a + \gamma \sum_{a'} \mathbb{E}_\pi \left( \sum_{t=t_0}^{\infty} \gamma^{t-t_0} r_{t_0+1} | e_{t+1} = e', a_{t+1} = a' \right) \right] \tag{6}$$

$$= \sum_{e'} \mathcal{P}_{ee'}^a \left[ \mathcal{R}_{ee'}^a + \gamma \sum_{a'} \pi(e', a') Q^\pi(e', a') \right] \tag{7}$$

Si la politique d'action à choisir dans un état donné consiste à maximiser la récompense cumulative, alors:

$$\pi^*(e) = \operatorname{argmax}_a Q^*(e, a)$$

Cependant, nous ne connaissons pas la fonction  $Q^*$ , nous utilisons donc un modèle statistique pour l'approcher. En prenant alors:

$$\pi(e) = \operatorname{argmax}_a Q^\pi(e, a)$$

L'équation de  $Q^\pi(e, a)$  sous cette politique se simplifie alors:

$$Q^\pi(e, a) = r + \gamma Q^\pi(e', \pi(e'))$$

Approcher  $Q^\pi$  revient alors à choisir un paramètre  $\theta$  pour notre modèle de manière à minimiser  $\delta$ , l'erreur de différence temporelle de notre modèle approche  $Q_\theta^\pi(e, a)$ :

$$\theta \in \operatorname{argmin}_\theta \mathcal{L}(\delta)$$

$$\delta = Q_\theta^\pi(e, a) - (r + \gamma \operatorname{argmax}_{a'} Q_\theta^\pi(e', a'))$$

Où  $\mathcal{L}$  est une fonction de perte. En pratique, l'optimisation est réalisée par "batches" de transitions  $B$  à l'aide d'une descente de gradient stochastique [?].

En effet, après chaque action prise, le calcul de la récompense obtenue est réalisé. Cette mémoire  $\mathcal{M}$  est composée de quadruplets  $(a, e, e', r) \in \mathcal{A} \times \mathcal{E}^2 \times \mathbb{R}$ . À chaque action, nous mettons en mémoire le quadruplet obtenu et réalisons une descente de gradient sur un batch de cardinal  $|B|$ .

Les paramètres de notre modélisation sont donc nombreux:

- $\theta$ , les paramètres de notre modèle statistique
- $\gamma$ , le poids données aux récompenses plus tôt dans le temps
- $\mathcal{L}$ , la fonction de perte
- La fonction de calcul de récompense,  $\psi : \mathcal{E} \longrightarrow \mathbb{R}$

### 3.4 Exploration vs exploitation

Au début de l'apprentissage, la fonction approchée par notre modèle ne sera pas de bonne qualité, et donc notre politique de prendre l'action avec la plus grande valeur de  $Q_\theta^\pi$  peut potentiellement ne pas converger. Pour encourager l'algorithme à explorer son espace d'action, on introduit un paramètre  $\epsilon \in [0, 1]$ . On modifie alors notre politique d'action de manière à faire une action aléatoire dans  $\epsilon\%$  des cas:

$$\pi(e) = \begin{cases} \text{random}(\mathcal{A}), & \text{si } \text{random}([0, 1]) < \epsilon. \\ \underset{a}{\operatorname{argmax}} Q^\pi(e, a), & \text{sinon.} \end{cases}$$

Il est aussi possible de faire varier  $\epsilon$  dans le temps, avec par exemple des valeurs plus grosses en début d'entraînement pour ensuite diminuer.

### 3.5 Modélisation de simulation

Modélisons l'ensemble des environnements  $\mathcal{E}$  de la manière suivante. Chaque élément  $e$  de cet ensemble représente l'état du système à un temps donné. Pour rester simple, prenons dans un premier temps la position  $(x_i)$ , la vecteur vitesse  $(\dot{x}_i)$ , l'accélération  $(\ddot{x}_i)$ , l'émission  $(\kappa_i)$ , et le temps totale passe a vitesse nulle  $(w_i)$  de chacune des  $V$  voitures ainsi que l'état des  $F$  feux dans l'environnement de simulation:

$$v_i = (x_i, \dot{x}_i, \ddot{x}_i, \kappa_i, w_i) \in \mathbb{R}^{3 \times 3 + 2}, 0 < i \leq V$$

$$f_j \in \{0, 1\}, 0 < j \leq F$$

$$e = (v, f) \in \mathbb{R}^{11V} \times \{0, 1\}^F$$

Une fonction de récompense pourrait être par exemple:

$$\psi(e) = \frac{1}{V} \sum_{i=0}^V (|\dot{x}_i| - |\ddot{x}_i| - \kappa_i)$$

Pour favoriser le flux mais minimiser les accélérations ainsi que les émissions. Il est maintenant possible de formuler des contraintes d'optimisation:

$$\phi_1(e) = \sum_{i=0}^V \kappa_i < c_1$$

$$\phi_2(e) = \sum_{i=0}^V w_i < c_2$$

Concernant la fonction de perte  $\mathcal{L}$ , prenons la somme des moindres carrés. L'optimisation à réaliser devient :

$$\begin{aligned} (\theta, \gamma) &\in \underset{\theta, \gamma}{\operatorname{argmin}} \mathcal{L}^\psi(\delta) \\ s.c. \phi_i(e) &< c_i, i \in \{1, 2\} \end{aligned}$$

### 3.6 Simulateur Sumo

L'environnement d'entraînement de notre algorithme est le simulateur de trafic routier Sumo~[?], développé par le centre aérospatial allemand. Ce simulateur permet de définir un réseau routier arbitrairement complexe et de définir des flux routiers sur cette géométrie. Il est ensuite possible de requêter le simulateur pour obtenir les positions, vitesse, accélérations, émission des véhicules pour prendre des décisions sur l'état des deux.

La Figure 2 présente un exemple de simulation sur Sumo.

### 3.7 Framework Flow

Le framework Flow~[?], développé par une équipe de chercheurs à UC Berkeley, est une abstraction open source permettant de simplifier la mise en place d'algorithme par renforcement pour le simulateur SUMO.

## 4 Cas pratiques

### 4.1 Cartpole

Dans le but de se familiariser avec les concepts liés à l'apprentissage par renforcement, nous avons étudié sa mise en oeuvre dans le cadre d'un modèle "jouet" bien connu, celui du pendule inversé. C'est un problème classique de contrôle utilisé pour développer des algorithmes d'apprentissage par renforcement dans un cadre simple.

OpenAI Gym [?] est un framework développé pour faciliter la recherche en apprentissage par renforcement. Cet outil permet de charger des environnements déjà implémentés avec lesquels il est ensuite possible d'interagir. OpenAI Gym permet aux développeurs de ne pas se préoccuper de la partie



Figures/sumo-screenshot.png

Figure 2: Capture d'écran du simulateur Sumo en action.

simulation physique des systèmes pour ne se concentrer que sur le développement d’algorithmes de RL.

Il est possible de charger des environnements de type jeux Atari, jeux de plateau, robots 2D ou 3D (cf. figure 3).

Même si Gym ne propose pas d’environnement propre au trafic routier, il constitue un framework référence pour l’apprentissage par renforcement, duquel s’inspire en partie le framework Flow que nous utiliserons au cours du projet. La prise en main de Gym constitue donc une bonne entrée en matière.

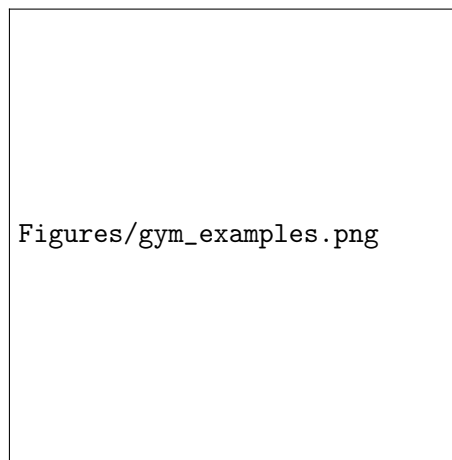


Figure 3: Exemples d’environnement OpenAI Gym

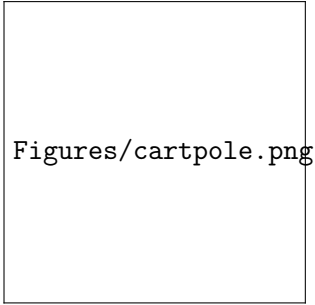
Le pendule inversé est constitué d’un mat en métal positionné sur un chariot mobile. Le chariot peut se déplacer sur la gauche et à la droite et la tâche de contrôle consiste à maintenir le mat en équilibre. Ce problème de contrôle est facilement soluble par les outils de contrôle classiques (e.g. commande optimale). Cet environnement constitue en revanche un banc d’essai intéressant pour implémenter des premiers algorithmes de RL.

Nous avons choisi dans un premier temps de nous concentrer sur le deep Q-learning. Dans cette approche, l’agent est un réseau de neurones qui prend en entrée l’état du système et donne en sortie l’action à entreprendre.

Dans l’exemple suivant, un agent de deep Q-learning a été codé dans la classe `DQNSolver` :

```
class DQNSolver:

    def __init__(self, observation_space, action_space):
```



Figures/cartpole.png

Figure 4: Environnement pendule inversé

```
self.exploration_rate = EXPLORATION_MAX
self.action_space = action_space
self.memory = deque(maxlen=MEMORY_SIZE)
self.model = Sequential()
self.model.add(Dense(24, input_shape=(observation_space,),
                    activation="relu"))
self.model.add(Dense(24, activation="relu"))
self.model.add(Dense(self.action_space, activation="linear"))
self.model.compile(loss="mse", optimizer=Adam(lr=LEARNING_RATE))

def remember(self, state, action, reward, next_state, done):
    self.memory.append((state, action, reward, next_state, done))

def act(self, state):
    if np.random.rand() < self.exploration_rate:
        return random.randrange(self.action_space)
    q_values = self.model.predict(state)
    return np.argmax(q_values[0])

def experience_replay(self):
    if len(self.memory) < BATCH_SIZE:
        return
    batch = random.sample(self.memory, BATCH_SIZE)
    for state, action, reward, state_next, terminal in batch:
        q_update = reward
        if not terminal:
            q_update = (reward + GAMMA *
                       np.amax(self.model.predict(state_next)[0]))
```



```

        q_values = self.model.predict(state)
        q_values[0][action] = q_update
        self.model.fit(state, q_values, verbose=0)
    self.exploration_rate *= EXPLORATION_DECAY
    self.exploration_rate = max(EXPLORATION_MIN,
                                self.exploration_rate)

```

## 4.2 Deep Q-learning pour une intersection

L'étape suivante consiste à implémenter un algorithme de DQL à un système de trafic routier. Dans cette partie, l'environnement est simulé par l'intermédiaire de SUMO. Le système le plus simple que l'on puisse imaginer pour le contrôle du trafic routier par feux tricolores est une intersection unique. La figure 5 montre une capture d'écran de l'interface graphique de SUMO au cours d'une simulation.

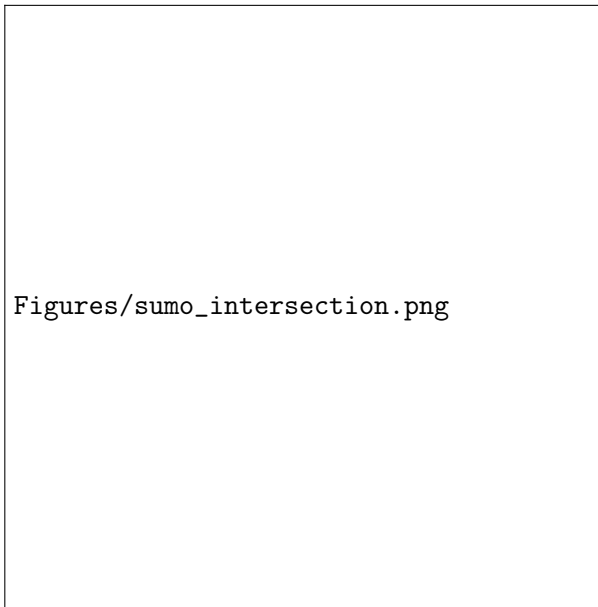


Figure 5: Intersection SUMO

La démarche est la suivante : on définit une architecture (ici une intersection), une demande de trafic et la période de temps de la simulation. Cela constitue un scénario qu'il est possible de répéter autant de fois que l'on veut. L'intervalle de temps entre chaque pas de simulation peut être configuré, ce qui permet de choisir la "vitesse" d'exécution de la simulation.

Dans notre cas, une heure de simulation correspond par exemple à un peu moins d'une minute d'exécution.

Il faut ensuite définir le triplet suivant  $(\mathcal{E}_t, \mathcal{A}_t, r_t)$  pour les états, les actions et les récompenses du système.

#### 4.2.1 Etats $\mathcal{E}_t$

Une façon de définir l'état de l'intersection consiste à "découper" chaque voie en cellules, et définir une matrice de positions et de vitesses comme suit (cf. figure 6): pour la matrice de positions on associe à chaque cellule la valeur 1 si un véhicule est présent dans la cellule, 0 sinon; pour la matrice de vitesses, on prend la vitesse du véhicule présent dans la cellule, normalisée par la vitesse limite.

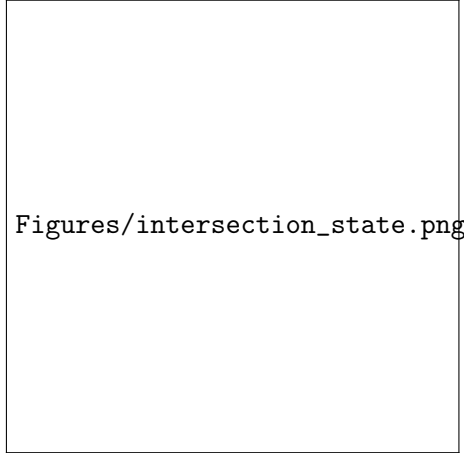


Figure 6: Etat de l'intersection

#### 4.2.2 Actions $\mathcal{A}_t$

L'agent a le choix entre deux actions  $\mathcal{A}_t = \{0, 1\}$  :

- 0  $\rightarrow$  feu vert pour la direction Est-Ouest
- 1  $\rightarrow$  feu vert pour la direction Nord-Sud

#### 4.2.3 Récompenses $r_t$

La récompense est définie à partir des délais encourus par les véhicules sur les voies menant à l'intersection. Pour chaque véhicule, les temps de décéléra-

tion, d'arrêt et d'accélération conduisent à un retard en comparaison avec un scénario sans intersection. On peut donc définir un retard cumulé pour l'ensemble des véhicules sur le réseau. A chaque fois qu'un véhicule traverse l'intersection, on soustrait son retard du retard cumulé. De cette façon il est possible de définir une récompense de la manière suivante :

$$r_t = C_d^{g^-} - C_d^{g^+} \quad (8)$$

où  $C_d^{g^-}$  est le retard cumulé au début de la phase de feu vert après l'action  $a_t$  et  $C_d^{g^+}$  le retard cumulé à la fin de la phase.

Avec une telle définition, on voit que l'agent de RL reçoit après chaque action un signal qui l'"informe" de la qualité de l'action entreprise. Une récompense positive traduit une diminution du retard cumulé et une récompense négative une augmentation de ce retard.

Choisir une récompense  $r_t$  revient d'une certaine façon à choisir ce que l'on veut optimiser. D'autres possibilités pourraient être la minimisation de la longueur des files d'attentes, la minimisation du niveau d'émissions, etc. . .

#### 4.2.4 Architecture du DQN

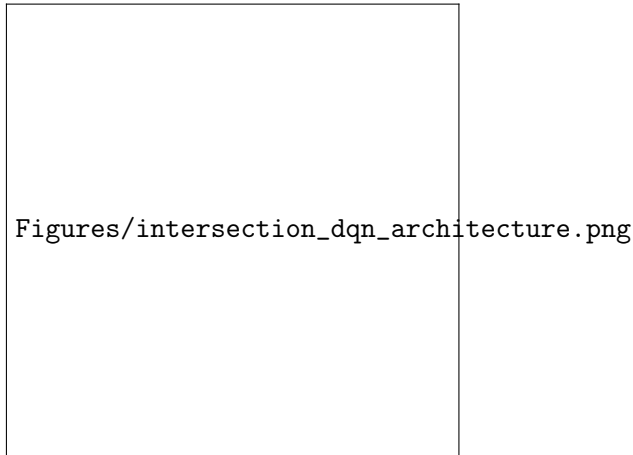


Figure 7: Architecture du DQN

## 5 Planning envisagé pour la suite du projet

Pour les dix prochaines semaines (avant l'échéance de fin de période trois, mi avril), l'objectif premier est de se familiariser avec l'outil de simulation SIM4SYS tout en commençant le travail de programmation sur un carrefour simple en ne prenant en compte que les feux tricolores.

Suite à cela, un binôme se penchera sur l'optimisation du premier modèle, pendant que l'autre implémentera le travail sur le logiciel de simulation en prenant compte des constantes évolutions résultantes de l'optimisation.

La deuxième étape sera de faire évoluer le modèle en prenant en compte un quartier complet mais toujours sans faire varier la vitesse des véhicules. Puis, tout en optimisant ce modèle, nous devrions arriver à la soutenance de fin de troisième période avec un modèle qui devrait montrer des éléments intéressants. Figure ~8

Lors de la dernière période de travail, nous nous concentrerons sur la montée en puissance dans l'algorithme : prendre en compte la régulation de la vitesse des véhicules. De même, en parallèle d'une phase d'optimisation, nous implémenterons ce modèle sur le simulateur SIM4Sys. Il nous restera donc une période de feature engineering pour spécialiser notre modèle et répondre ainsi à toutes les demandes de Cil4Sys. Figure ~9.

## 6 Conclusion

La problématique posée d’optimisation du trafic urbain pour limiter les émissions de polluants nous a amenés dans un premier temps à réaliser un état de l’art sur les méthodes mises en œuvre dans le domaine de l’optimisation de trafic urbain, et sur la réduction des émissions de polluants. Cette étude nous a amenés à considérer un modèle d’apprentissage par renforcement pour répondre à nos besoins, ce type de modèle ayant fait ses preuves dans la littérature.

Nous utiliserons à cet effet un simulateur de trafic Sumo, qui nous permettra de tester les algorithmes développés et leur efficacité. Nous pourrons améliorer notre simulation en l’approchant des conditions réelles grâce aux données de trafic fournies par l’API Tomtom qui nous donnera une évaluation réaliste de demande de trafic.

Les modèles développés seront par la suite intégrés au simulateur SIM4Sys développé par l’entreprise CIL4Sys. Nous itérerons sur le développement des modèles de manière itérative afin de pouvoir en parallèle les améliorer de manière continue et garantir leur bonne intégration au logiciel SIM4Sys.

Figure 8: Planning partie 1

Figure 9: Planning partie 2