Rapport Final

Projet Fil Rouge : Contrôle du trafic routier dans une agglomération de taille modérée

Equipe Telecom :
Nicolas LOUIS
Julien GUEGAN
Cyrille NOUBOUE

Encadrant CIL4SYS :
Philippe GICQUEL

Tuteur Académique:
Olivier FERCOQ

Table des matières

Introduction

La percée du parti politique français Europe Ecologie - Les Verts aux dernières élections européennes montre, s'il en est, la préoccupation croissante des français vis-à-vis des questions écologiques. Ce n'est qu'un des nombreux exemples illustrant la prise de conscience en marche en France à propos du dérèglement climatique.
Face aux défis que posent ces questions, créer un transport vert est une des priorités. En tant que premier émetteur de CO_2 en France métropolitaine[1], représentant près de 30% des émissions, le transport routier, aérien et maritime pollue davantage que l'industrie et les activités tertiaires. Concernant le transport routier, il y a bien des solutions techniques qui émergent, comme par exemple les voitures électriques, mais aussi d'ordre comportemental comme la promotion du covoiturage, des modes de transport alternatifs (vélo électriques, trottinettes, transports en commun…). Malgré tout, dans cette période de transition, plusieurs autres initiatives doivent émerger rapidement pour tenter de gagner la course contre la montre qui s'est engagée.
Partant du constat que l'ensemble du parc automobile ne peut être remplacé instantanément par des véhicules écologiques, CIL4SYS[2] se propose de réduire les émissions de gaz à effet de serre en fluidifiant le trafic routier d'agglomérations de tailles modérées. En limitant les embouteillages qui représentent une source de pollution importante, CIL4SYS espère contribuer significativement aux enjeux du moment. Pour cela, les résultats récents obtenus par des algorithmes d'apprentissage dans de nombreux domaines laissent présager que cette intuition est potentiellement envisageable.
Pour autant, avant de mettre en production des solutions techniques viables, il convient de tester cette intuition. En s'appuyant sur les travaux d'une équipe d'étudiants de l'année dernière, le projet Fil Rouge que les auteurs de ce rapport se proposent de mener est d'évaluer à travers des simulations, les plus réalistes possibles, la pertinence d'un trafic routier contrôlé en temps réel et adaptatif qui limiterait les temps d'arrêts, quelques soient les situations rencontrées. Un modèle d'apprentissage sera alors produit et optimisé pour prendre en compte les différentes contraintes du projet, notamment en temps de calcul et en réactivité du système.

[1] Source : SITEPA / Format SECTEN - avril 2019.
[2] CIL4SYS a été créée en juin 2015 par Philippe GICQUEL qui a travaillé pendant 25 ans en R&D automobile chez EDAG et PSA Peugeot-Citroën. Le coeur 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.

I. Environnement de simulation

En l'état des connaissances actuelles dans le domaine, le projet est encore en phase exploratoire. Aussi plutôt que de s'appuyer sur des données réelles difficiles à rassembler (trajectoires individuelles de l'ensemble des véhicules d'une ville de 100 000 habitants, vitesses, temps d'arrêt, etc.), il convient de générer artificiellement l'ensemble des variables nécessaires à travers un simulateur éprouvé. De plus, au regard des algorithmes d'apprentissage possibles pour résoudre ce type d'optimisation, l'apprentissage par renforcement semble le plus adapté. Aussi l'environnement de travail devra aussi être propice à l'apprentissage par renforcement et permettre l'interaction avec le simulateur de manière fluide et optimale.

De plus, Python étant un langage de programmation maîtrisé par tous les membres du groupe, nous l'avons choisi comme langage de programmation, et c'est autour de ce choix que s'est structuré la sélection de l'architecture et de tous les composants du projet.

Afin de s'inscrire dans la continuité du travail du groupe qui a nous a précédé sur ce projet, nous avons décidé pour le moment de poursuivre le travail avec les mêmes outils. Notre modèle repose donc sur un certain nombre d'outils qu'il est nécessaire d'introduire à ce stade du document, c'est ce que nous faisons à présent.

I.1 Outils de simulation : SUMO & TraCi

SUMO (Simulation of Urban MObility)[3] est un simulateur de trafic microscopique libre, multi-plateforme, développé et maintenu par le ministère des transports allemand.

Il s'intègre à Python par le biais d'une API nommée **TraCi**[4] de haute qualité. Ce simulateur permet entre autre de requêter et de modifier l'état d'un réseau routier à un pas de temps prédéfini via TraCi. Ainsi, elle permet de prendre des actions et de calculer les récompenses lors de l'entraînement de notre agent.

Le module **TraCi** de SUMO permet aussi de gérer entre autre l'état des **feux de signalisation**, de requêter les émissions de chaque voiture, gérer les collisions éventuelles, ainsi que la redirection, suppression et l'instanciation de véhicules à un pas de temps défini.

Dans SUMO les routes sont représentées par des arêtes ("**edges**" en anglais), chaque arête pouvant avoir une ou plusieurs voies ("**lanes**" en anglais). Chaque arête et chaque voie est

[3] https://elib.dlr.de/6661/2/dkrajzew_MESM2002.pdf
[4] https://SUMO.dlr.de/docs/TraCI.html

identifiée de manière unique par une chaîne de caractères, ce qui permet d'interagir facilement avec les véhicules de la simulation.

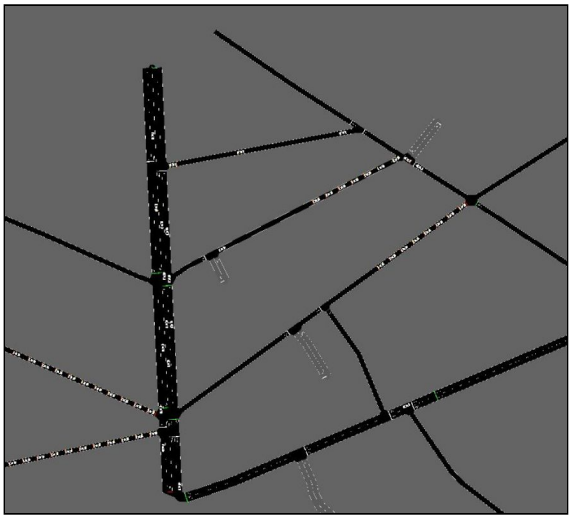


Figure 1: *Représentation du quartier Issy-les moulineaux dans SUMO*

Les feux sont placés à certaines intersections sur la carte. Mais dans SUMO, le nombre de feux peut être supérieur au nombre d'intersections. L'état des feux à une intersection est défini par une chaîne de caractères où chaque caractère représente l'état d'un des feux de l'intersection. Par exemple, l'état des feux d'une intersection gérée par trois feux pourrait être "**GrG**" ce qui voudrait dire que le premier et le dernier feu sont verts et le feu intermédiaire rouge.

Le premier caractère de la chaîne représente l'état du feu le plus au nord de l'intersection gérée, le suivant le feu directement après en allant dans le sens anti-horaire, et ainsi de suite jusqu'à la fin de la chaîne. Enfin, au même titre que les arêtes, SUMO identifie chaque jeu de feux gérant une intersection par une chaîne de caractère unique permettant de requêter le feu via Traci.

```
action_spec = OrderedDict({
    # The main traffic light, in sumo traffic light state strings
    # dictate the state of each traffic light and are ordered counter
    # clockwise.
    "30677963": [
        "GGGGrrrGGGG",  # allow all traffic on the main way w/ U-turns
        "rrrrGGGrrrr",  # allow only traffic from edge 4794817
    ],
    "30763263": [
        "GGGGGGGGGG",  # allow all traffic on main axis
        "rrrrrrrrrr",  # block all traffic on main axis to unclog elsewhere
    ],
    "30677810": [  # the smallest of all controlled traffic lights
        "GGrr",
        "rrGG",
    ],
})
```

Figure : *Exemple d'instanciation d'état des feux de signalisation*

I.2 RLib

Rlib est une librairie Python facilitant la mise en place d'un modèle d'algorithme d'apprentissage par renforcement. Il gère plusieurs types d'algorithmes, dont le **DQN** (Cf. § II.1) et l'entraînement sur des supports de calcul distribués. Elle assure donc un passage à l'échelle sur des ressources de calcul plus importantes.

I.3 FLOW

Afin de simplifier l'interaction entre SUMO & Rlib, un laboratoire de l'UC de Berkeley travaille sur une solution open source. En python, cette librairie s'appelle **FLOW** [1]. Elle permet la gestion à la fois de la simulation du trafic routier sur SUMO par des appels via l'API TraCi, et l'instanciation des algorithmes d'apprentissage par renforcement, notamment les fonctions de l'agent, les fonctions de récompense, etc.

Une simulation FLOW est constituée de 3 éléments principaux :

- Le **réseau** ("**Network**"), qui est l'ensemble des éléments physiques du réseau routier, comprenant les routes, les voitures, les feux de signalisation, etc. FLOW permet la création et la personnalisation de ce réseau. Les réseaux peuvent être soit créés de toute pièce, soit créés à partir de cartes en **Open Street Map** (fichiers .osm). Cette fonctionnalité s'avère particulièrement pratique lorsque l'on souhaite travailler sur des routes réelles. FLOW permet entre autre la gestion des flux, des trajets et des comportements des véhicules. Pour le projet, nous avons décidé du paramétrage optimal de notre réseau sur FLOW, afin qu'il s'approche le plus possible d'une simulation réelle (Cf. III. Objectifs à atteindre).

- **L'environnement** est le deuxième composant d'une simulation FLOW. Cette partie concerne la mise en place et le paramétrage de l'algorithme d'apprentissage par renforcement. Ainsi, il permet la définition des agents et des fonctions de récompenses. Comme pour les réseaux, FLOW propose une multitude de paramètres pour laisser la possibilité à l'utilisateur de personnaliser sa simulation et le comportement de son agent. L'agent peut contrôler la vitesse des véhicule ou gérer les feux de signalisation. Les fonctionnalités proposées par cette API s'avèrent ainsi donc très intéressantes pour les besoins du projet.

- Le troisième élément important d'une simulation FLOW est une **Expérience.** C'est une classe qui définit les paramètres de l'algorithme pour maximiser l'espérance de la récompense lors de d'apprentissage de l'agent

La majeure partie du code se fait donc via la librairie FLOW, c'est par elle que l'utilisateur passe pour gérer son environnement et son algorithme (DQN, PPO, etc. . .). Elle permet aussi de définir le nombre de processus SUMO à instancier et le nombre d'itérations durant l'entraînement.

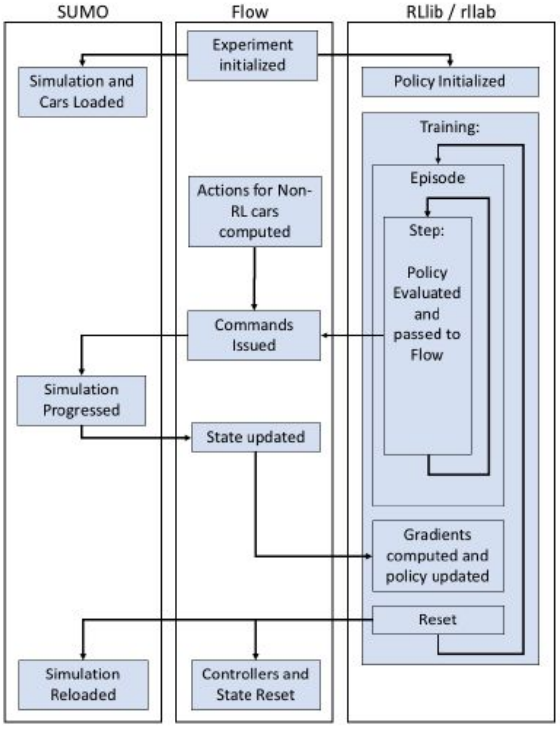


Figure : *Schéma représentant l'architecture globale de l'environnement de simulation*

I.4 Choix de la ville test

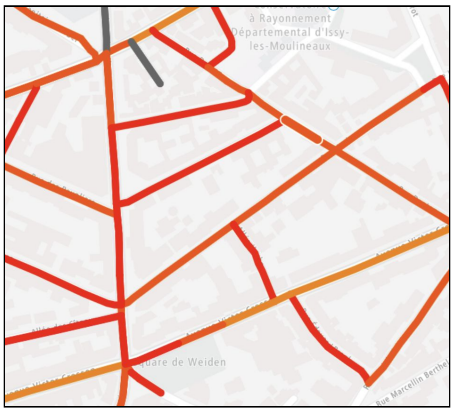


Figure : *Quartier de issy-les-moulineaux*

L'optimisation du trafic routier par apprentissage par renforcement étant relativement gourmand en ressources de calcul, nous avons fait le choix de limiter nos premiers essais à un espace restreint du monde réel. C'est pourquoi nous avons choisi un quartier de la ville d'Issy-les-moulineaux en Île-de-France, où le trafic est relativement normal.

Par ailleurs, il se trouve que ce quartier est très bien représenté sur Open Street Map, avec les feux de signalisation déjà implémentés dans la carte, ce qui nous a fait gagner du temps dans la configuration du réseau routier.

II. Apprentissage par renforcement

Pour résoudre ce type de problème, les méthodes d'apprentissage par renforcement sont les plus adaptées [2], [3]. En effet, ces méthodes sont connues pour être efficaces dans des cas où un agent évolue et interagit dans un environnement par des actions qu'il choisit. Chaque action engendre une réponse de l'environnement qu'on appelle récompense (négative ou positive) qui le mènera dans un nouvel état. Ce type d'approche se distingue essentiellement des approches classiques du machine learning du fait qu'un jeu de données n'est pas disponible et que ces données ne sont pas indépendantes et identiquement distribuées.

Si on appelle politique π la stratégie de notre agent pour choisir une action a lorsqu'il se trouve dans un état s alors le but est de trouver la politique qui maximise la récompense R attendue par l'agent. Pour des transitions d'environnement P et une politique π stochastique, la probabilité d'une trajectoire τ (séquence d'états et d'actions) est :

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$$

La récompense attendue est alors :

$$J(\pi) = \int_{\tau} P(\tau|\pi)R(\tau) = E_{\tau\sim\pi}[R(\tau)]$$

On peut alors exprimer le problème d'optimisation par :

$$\pi^* = \arg\max_{\pi} J(\pi) \quad \text{où } \pi^* \text{ est la politique optimale.}$$

Une notion importante à définir est celle de la fonction valeur qui permet de connaître la valeur d'une paire état-action, c'est-à-dire la récompense attendue si on commence dans un état-action fixé puis on agit selon une politique particulière. On définit les grandeurs suivantes :

1. une fonction valeur sur politique : $V^{\pi}(s) = E_{\tau\sim\pi}[R(\tau)|s_0 = s]$ renvoie la récompense attendue si on commence dans l'état s et qu'on agit toujours selon la politique π

2. une fonction action-valeur sur politique $Q^{\pi}(s) = E_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a]$ renvoie la récompense attendue si on commence dans l'état s, on prend une action a arbitraire et qu'on agit toujours selon la politique π

3. une fonction valeur optimale $V^*(s) = \max_{\pi} V^{\pi}(s)$

4. une fonction action-valeur optimale $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$

On distingue 2 principales familles dans les algorithmes d'apprentissage par renforcement : ceux avec et ceux sans modèle. Le principal avantage d'avoir un modèle est qu'il permet à l'agent de planifier en anticipant, en voyant ce qui se passerait pour une gamme de choix possibles et en décidant explicitement entre ses options. Les agents peuvent ensuite distribuer les résultats de la planification à l'avance dans une politique apprise. Cependant, un modèle de l'environnement n'est généralement pas disponible pour l'agent comme c'est le cas pour notre problème de trafic routier. Sans modèle, l'agent doit apprendre le modèle par expérience, ce qui crée plusieurs défis. Le plus grand défi est que le biais dans le modèle peut être exploité par l'agent, résultant en un agent qui fonctionne bien par rapport au modèle appris, mais se comporte de manière sous-optimale dans l'environnement réel. Dans les méthodes basés sur modèle, deux principales familles d'approche existent :

- **Optimisation de politique :** Les méthodes de cette famille représentent explicitement une politique $\pi_\theta(a|s)$. Elles optimisent les paramètres θ par montée de gradient de la fonction objectif $J(\pi_\theta)$ (ou une approximation). Cette optimisation est réalisée *sur-politique*, chaque mise à jour utilise uniquement les données collectées tout en agissant selon la version la plus récente de la politique.

- **Q-Learning** : Les méthodes de cette famille apprennent un approximateur $Q_\theta(s,a)$ de la fonction valeur d'action optimale, $Q^*(s,a)$. Elles utilisent généralement une fonction objective basée sur l'équation de Bellman. Cette optimisation est presque toujours effectuée *hors-politique*, chaque mise à jour peut utiliser les données collectées à tout moment pendant l'entraînement, quelle que soit la manière dont l'agent a choisi d'explorer l'environnement lorsque les données ont été obtenues. La politique correspondante est obtenue via la connexion entre Q^* et π^* : les actions prises par l'agent Q-learning sont données par
$$a(s) = \arg\max_a Q_\theta(s,a)$$

Dans ce projet, les algorithmes de Deep Q Network (DQN) repose sur les méthodes de Q-Learning quant au Proximal Politic Optimization (PPO), il repose sur l'optimisation de politique. Ces algorithmes ont été testés via la librairie RLlib.

II.1 Deep Q Network

L'algorithme consiste à trouver une stratégie qui maximise une fonction de récompense. A chaque instant, l'agent obtient une récompense $r_t \in \mathbb{R}$ et cherche à maximiser la récompense future cumulée R_t définit comme suit :

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

avec $\gamma \in]0,1]$ le *discount rate* permettant de donner plus ou moins d'importance aux récompenses futures. On définit souvent la stratégie d'exploration de l'agent avec la stratégie *epsilon-greedy* qui consiste, selon une probabilité ε , à choisir une action aléatoire ou une action bien choisie calculée par Q-learning. En pratique on fait en sorte que cette

probabilité décroît le long de l'entraînement. Le problème principal avec la formulation précédente de Q-learning est qu'elle nécessite de connaître $Q^{\pi}(s', a')$ et donc d'avoir une connaissance complète de son environnement, c'est-à-dire des paires états-actions.
On a souvent un trop grand nombre de combinaisons possibles d'états-actions et il est difficilement possible d'explorer ces états en un temps raisonnable. L'idée du DQN [8] est de prédire les valeurs de Q^{π} en utilisant un réseau de neurones prenant en entrée les états de départ et d'arrivée s et s', les actions a entreprises, et les récompenses respectives r obtenues. Contrairement à un problème de classification, la dernière couche du réseau utilise une fonction d'activation linéaire qui renvoie les valeurs réelles Q^{π} associées aux actions disponibles. Le but de ce réseau de neurones est d'approximer le mieux possible l'équation du Q-learning en minimisant la différence temporelle δ :

$$\theta \in \arg\min_{\theta} \mathcal{L}(||\delta||)$$
$$\text{avec } \delta = Q^{\pi}_{\theta}(s,a) - (r + \gamma \arg\max_{a'} Q^{\pi}_{\theta}(s,a'))$$

Pour entraîner ce réseau de neurones, il est donc nécessaire d'avoir un jeu de données de $Q^{\pi}_{\theta}(s,a)$ qui sera obtenu notamment pendant la phase d'exploration de l'algorithme. Notons que, au début de l'algorithme, le réseau de neurones sera peu entraîné car ayant peu de donnée à disposition et donc prédira mal les valeurs de Q^{π}, ensuite plus l'agent aura explorer son environnement, plus le réseau de neurones aura des données disponibles (garder en mémoire et sélectionnées aléatoirement) pour sa phase d'apprentissage.

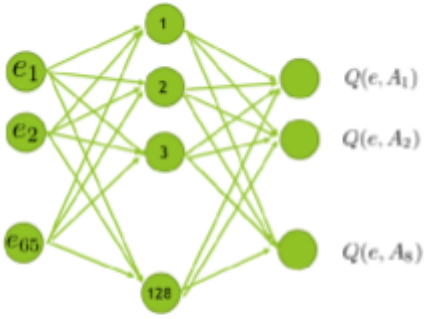


Figure: *Visualisation du réseau de neurone approchant les Q-values*

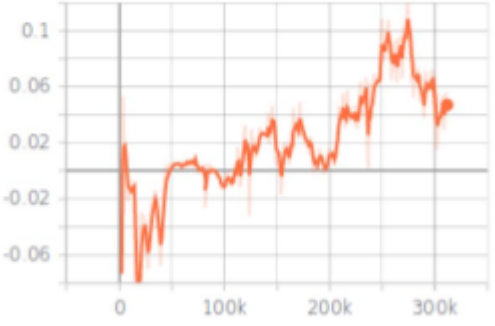


Figure: *Evolution de la différence temporelle*

II.2 Proximal Policy Optimization

Contrairement au Deep Q Network, l'algorithme Proximal Policy Optimization (PPO)[5] est une méthode sur-politique : on cherche à maximiser la récompense attendue $J(\pi_\theta) = E_{\tau \sim \pi_\theta}[R(\tau)]$ en utilisant une montée de gradient :

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta)|_{\theta_k}$$

[5] on discute ici de la version qui "clippe" la fonction objectif et non celle qui la pénalise

Le terme $\nabla_\theta J(\pi_\theta)$ est le *gradient de la politique* qu'on peut définir d'après [7] et le théorème du gradient de politique en utilisant l'astuce du log-dérivée, on obtient :

$$\nabla_\theta J(\pi_\theta) = E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\, R(\tau)]$$

Puis on peut estimer cette espérance par la moyenne empirique d'un échantillon en collectant un ensemble de trajectoires où chaque trajectoire est obtenue en laissant l'agent interagir dans l'environnement en suivant la politique π_θ.

Maintenant, on se demande comment, lors de la montée de gradient, choisir le plus grand pas d'amélioration de la politique en utilisant les données à disposition (sans aller si loin ce qui provoquerait un effondrement des performances). D'abord, on définit la fonction avantage A qui permet de décrire à quel point une action est meilleure que d'autres en moyenne par : $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. PPO-clip[9] met à jour la politique via :

$$\theta_{k+1} = \arg\max_\theta E_{s,a\sim\pi_{\theta_k}}[L(s,a,\theta_k,\theta)]$$

où L est l'avantage surrogé : une mesure du comportement de la stratégie actuelle π_θ par rapport à l'ancienne stratégie π_{θ_k}. Cet avantage surrogé vaut en version simplifié :

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a),\ g(\varepsilon, A^{\pi_{\theta_k}}(s,a))\right)$$

$$\text{avec } g(\varepsilon, A) = (1+\varepsilon)\,A \text{ si } A \geq 0 \text{ et } (1-\varepsilon)\,A \text{ sinon}$$

où ε est un hyper-paramètre indiquant approximativement à quelle distance la nouvelle politique peut s'éloigner de l'ancienne. Si l'avantage est positif alors l'objectif augmentera si l'action devient plus probable c'est-à-dire si $\pi_\theta(a|s)$ augmente. D'un autre côté si l'avantage est négatif, l'objectif augmentera si l'action devient moins probable, c'est-à-dire si $\pi_\theta(a|s)$ diminue. Dans ces deux cas, le minimum dans ce terme pose une limite sur l'augmentation possible de l'objectif ainsi la nouvelle politique ne peut pas s'éloigner trop loin de l'ancienne politique. Ainsi cette astuce dite de "clipping" sert de régularisation en empêchant la politique de changer radicalement.

Enfin, il est important de noter que la politique, ici, est en pratique également modéliser par un réseau de neurones avec en sortie un vecteur de probabilité des différentes actions possibles[6].

[6] Dans le cas d'un espace continu d'action, la sortie du réseau de neurones est la moyenne (et variance) d'une distribution gaussienne et l'action sera par la suite échantillonnée sur cette gaussienne.

II.3 Modélisation

États et Actions

On cherche une représentation du système à un temps t qui encodent suffisamment d'informations pour trouver une solution à notre problème de fluidification du trafic routier. D'abord, l'état de l'ensemble de α feux présents dans notre environnement est :

$$e^f \in \{0,1\}^\alpha$$

Ensuite, on cherche à décrire l'ensemble qui modélise notre ensemble de voitures caractérisées par leur position, orientation, vitesse, accélération, taux d'émission de CO2 et temps passé à l'arrêt. Si on observe β véhicules dans l'environnement alors on a :

$$e^v = (x, y, \theta, v, a, \kappa, t)^\beta$$

A propos des actions, on choisit dans un premier temps, que l'agent peut contrôler l'état de chacun des α feux, une action correspond donc à un vecteur de α booléens (rouge ou vert), et il y a 2^α combinaisons possibles. En fait, pour SUMO, une intersection classique de 2 routes est composé de 16 feux (4*4 trajectoires possibles) et il y aurait donc $2^{16} = 65536$ états possibles. Mais, ce n'est pas le cas dans la réalité, en effet on autorise jamais le fait que le feux E/W soit vert en même temps que le feux N/S. De plus, il est également rare de pouvoir faire demi-tour à une intersection ou encore de mettre tous les feux dans le même état. Ainsi, il est raisonnable de fixer un certains nombre d'états bien choisi permettant de réduire grandement la dimensionnalité du système.

Au regard des premiers résultats, nous avons, par la suite, décidé de changer la modélisation de notre environnement en modifiant les états possibles que notre agent observe en essayant de réduire ses dimensions. En effet, comme conseillé dans [10], on a choisi de modéliser les états du système seulement par le nombre de véhicule q présent sur chaque axe de trafic routier[7] et non pas, comme précédemment, par les composantes de position, vitesse, accélération et temps d'attente des véhicules qui ne permettait que d'observer un petit nombre fixé de véhicule. Ainsi :

$$e^v = (x, y, \theta, v, a, t)^\beta \text{ devient } e^v = (q)^{N_a}$$

avec N_a le nombre d'axes total dans notre réseau de circulation. Cette nouvelle modélisation des états est moins complexe et de moins grandes dimensions mais nécessite de parcourir chaque axe et compter chaque véhicule présent. On présente les résultats obtenus avec cette modélisation après entraînement dans le § V.2.

[7] Mais on garde la partie qui modélise les états des feux

Fonction de récompense

L'objectif de la fonction de récompense est que sa maximisation permet d'approximer le mieux possible le problème de fluidification du trafic sous la contrainte que les voitures doivent parcourir une certaine distance dans un temps fini (empêcher que les voitures restent à l'arrêt). Une fonction de récompense proposée au départ était :

$$\mathcal{R} = \sum_{i=1}^{\beta} C_{1 v_i \geq V_{\min}} + C_{2 \kappa_i \leq \max} - C_{3 t_i \geq T_{\min}} - C_{4 \ddot{x}_i \geq A_{\max}}$$

En outre, au fil de l'évolution du projet, nous avons choisi d'enlever le terme qui permettait de prendre en compte la production du CO2 pour 2 raisons:

- ce facteur est déjà relié à l'accélération des véhicules ;
- sa valeur est obtenue par le biais d'une API fonctionnant un peu sous un modèle "boite-noire", sur lequel nous ne maîtrisons pas le mode de calcul.

La nouvelle forme de récompense choisie a donc été :

$$\mathcal{R} = \sum_{i=1}^{\beta} C_1 \, \mathbb{I}_{v_i \geq v_{\min}} - C_3 \, \mathbb{I}_{t_i \geq \tau} - C_4 \max(0, a_i)$$

Celle-ci a servi de modèle pour les deux modélisations d'environnement du quartier d'Issy-les-Moulineaux.

II.4 Modélisation d'un simple carrefour et fonction de récompense simplifiée

Comme nous le verrons par la suite dans la partie résultats, la modélisation d'un quartier d'Issy Les Moulineaux a permis d'améliorer sensiblement la fluidité du trafic routier avec des vitesses moyennes plus importantes et des temps d'arrêts plus limités. Cependant, ces résultats ont été obtenus au prix de fonctions de récompense complexes et pas aisément généralisables. Aussi, dans les dernières semaines du projet, il a été décidé, en accord avec la société CIL4SYS, de simplifier le problème en ne se concentrant qu'au niveau d'un carrefour en croix avec des feux de circulations comme seul agent de contrôle et une fonction de récompense repensée.

Fonction de récompense

L'objectif de cette fonction de récompense, en plus d'être beaucoup plus simple à interpréter, est de **maximiser la vitesse des véhicules sur les routes**, pour que la vitesse de ces véhicules se rapproche de la vitesse maximale autorisée sur le noeud.

Ainsi, cette fonction de récompense pénalise l'écart de vitesse entre un véhicule à un instant t, et la vitesse maximale que ce véhicule pourrait avoir sur une route donnée. La fonction de récompense prends donc la forme :

$$reward = \frac{cost}{N_{veh}} \quad \text{où} \quad cost = \Delta_t * \sum_{i}^{N} \frac{Vmax_k - Vi_t}{Vmax_k}$$

Avec :

- Δ_t : Pas de temps entre 2 observations par l'agent
- $Vmax_k$: Vitesse maximale possible sur la route k
- Vi_t : Vitesse du véhicule i à l'instant t
- $N_{veh]=N}$: Nombre de véhicules dans l'environnement

L'idée sous-jacente derrière cette approche est que grâce aux modèles qui contrôlent les véhicules, ces derniers anticipent le changement d'état d'un feu en approchant d'un croisement, et réduisent ainsi leur vitesse. En optimisant le changement d'état des feux de signalisation, l'agent devrait aider les véhicules à garder leur vitesse de croisière le plus longtemps possible.

III. Objectifs à atteindre

III.1 Amélioration de l'environnement de simulation

La première difficulté est la prise en main et le fonctionnement de l'environnement. En effet, si la librairie FLOW est une solution open source extrêmement pratique pour l'apprentissage par renforcement appliqué à l'optimisation de trafic routier, sa maintenance est très difficilement assurée. Il y a relativement peu de contributeurs au projet, et comme nous l'avons vu, elle interagit avec beaucoup d'autres librairies et packages. Ainsi, La compatibilité entre FLOW et toutes ses dépendances ou celles de SUMO n'est donc pas assez bien assurée. A l'heure actuelle déjà FLOW n'est plus compatible avec la dernière version de SUMO, ce qui rend toute simulation très difficile. Il a donc été question pour nous de mettre à jour le code fourni par nos prédécesseurs.

III.2 Contrôle / comportement des véhicules

L'étape suivante a ensuite été d'instancier un comportement réaliste du flux de voitures à travers la carte. Ce comportement dépends de plusieurs paramètres qui sont:

- **Le mode de conduite de la voiture** : ce paramètre modifiable via FLOW permet de fournir une intelligence basique à la voiture qui lui indique le mode de fonctionnement qu'elle doit adopter. Le mode qui se rapprochait le plus de nos besoins est le mode "humain", qui permet à une voiture d'accélérer ou de freiner en fonction des feux de signalisation ou des autres voitures. Pour renforcer le réalisme de certains comportements des véhicules, comme le fait de se rabattre sur la file de droite avant de tourner à droite, en évitant ainsi la formation de bouchons non réalistes faut de pouvoir se rabattre à temps, nous avons dû exploiter toutes les fonctionnalités offertes par SUMO[8] et rentrer dans le détail de ces dernières. Nous avons dû coder les comportements désirés dans les paramètres "`car_following_params`" et "`lane_change_params`" avec notamment le code "`2722`" (chiffre binaire converti en décimal).

```
vehicles = VehicleParams()
vehicles.add("human",
             acceleration_controller=(IDMController, {}),
             car_following_params=SUMOCarFollowingParams(
                 speed_mode="right_of_way"),
             lane_change_params=SUMOLaneChangeParams(
                 lane_change_mode=2722)
             )
vehicles.add("rl",
             acceleration_controller=(IDMController, {}),
      car_following_params=SUMOCarFollowingParams(
```

[8] https://SUMO.dlr.de/docs/TraCI/Change_Vehicle_State.html

```
            speed_mode="right_of_way"),
        lane_change_params=SUMOLaneChangeParams(
            lane_change_mode=2722),
        color="cyan")
```

- **La gestion des trajectoires des voitures** : FLOW exige en effet d'instancier à l'avance les trajets que les véhicules effectueront. L'équipe précédente a ainsi créé ce qu'ils ont baptisé des "**rails**", qui sont des trajets types des véhicules à travers le quartier. Il a donc été question d'implémenter des trajets réels et diversifiés qui peuvent représenter le flux de trafic le plus réaliste possible. Pour cela, la modification de certains trajets plus conformes avec un comportement humain a été élaboré. Tout en conservant l'exigence de faire passer des véhicules sur l'ensemble des noeuds et des rails possibles du quartier d'Issy-les-Moulineaux, l'un des comportements de l'équipe précédente prévoyait de faire faire une boucle à certains véhicules qui favorise la congestion du trafic. Cela a été modifié dans le nouvel environnement avec le code suivant :

```
class IssyOSMNetwork(Network):

    def specify_routes(self, net_params):
        return {
            "-100822066": [ #N
                "-100822066",
                "-352962858#1",
                "-352962858#0",
                "-4786940#1",
                "-4786940#0",
            ],

            "4794817" : [ #Loop
                "4794817",
                "4786972#0",
                "4786972#1",
                "4786972#2",
                "4786965#1",
                "4786965#2",
                "4786965#3",
                "4786965#4",
                "4786965#5",
            ],

            "4783299#0" : [ #Loop bis
                "4783299#0",
                "4783299#1",
                "4783299#2",
                "4783299#3",
                 "4783299#4",
                "4783299#5",
                "4783299#6",
                "4786940#0",
                "4786940#1",
                "352962858#0",
                "4795742#0",
                "4795742#1",
                "4786965#3",
                "4795729",
                "100822066",
            ],

            "155558218": [     #SE
                "155558218",
                "4786940#1",
                "352962858#0",
                "352962858#1",
                "100822066",
            ],
        }
```

- **Le nombre de voitures présentes dans le réseau**, et éventuellement sa variation en fonction du temps (pour avoir un phénomène d'**heures de pointe**). Pour ce point, comme pour les trajectoires, nous avons modifié aussi le flux de véhicules. Après avoir testé plusieurs environnements possibles et bien que l'insertion aléatoire de véhicules puisse davantage correspondre à la réalité, nous avons fini par instancier un flux de véhicules constant grâce au paramètre : `period` dans la fonction `inflow`. Ainsi les résultats sont plus facilement comparables entre ceux obtenus sans entraînement et les résultats obtenus avec entraînement. En supprimant la part d'aléatoire, les résultats peuvent être interprétés de manière uniforme.

```
inflow.add(veh_type="flow", edge="4794817", period=8, depart_speed=7,
depart_lane="best")
inflow.add(veh_type="flow", edge="4783299#0",period=5,
depart_speed=7, depart_lane="best")
inflow.add(veh_type="flow", edge="-100822066", period=3,
depart_speed=7, depart_lane="best")
inflow.add(veh_type="flow", edge="155558218", period=15,
depart_speed=7, depart_lane="best")
inflow.add(veh_type="flow", edge="itineraire_bis", period=10,
depart_speed=7, depart_lane="best")
```

III.3 Ajout de perturbations

Un des objectifs du projet était aussi d'ajouter des perturbations dans le système de manière à pouvoir rendre l'expérience la plus réaliste possible. Malheureusement après de nombreuses recherches dans la documentation de SUMO mais aussi auprès de l'équipe de FLOW Project de l'Université de Berkeley, cette option a dû être abandonnée. SUMO ne permet pas d'instancier des perturbations tierces de type piétons ou autres et le développement *ad hoc* de fonctions de perturbations aurait été au détriment du coeur du projet.

III.4 Augmenter le spectre des agents de contrôle

Pour optimiser le trafic, l'agent peut jouer sur 3 types d'actions :

- Contrôler la vitesse des véhicules
- Contrôler l'état des feux de signalisation
- Contrôler les deux simultanément

Il a été question d'évaluer chaque type d'action et son efficacité, ainsi que sa faisabilité, notamment pour ce qui est du temps d'entraînement et de l'efficacité du modèle.

Cependant les nombreuses difficultés rencontrées au cours du projet pour mettre à jour le code et évaluer différents environnements ainsi que différentes fonctions de récompense ont rendu impossible l'exploration du contrôle de la vitesse des véhicules. Il a été décidé en cours de projet, en accord avec CIL4SYS, de ne se concentrer que sur les feux de

signalisation. Ils représentent, à l'heure actuelle, les seuls capteurs à la main des pouvoirs publiques pour réellement pouvoir influer sur la fluidité du trafic, faute de déploiement de véhicules autonomes en suffisamment grand nombre en 2020.

III.5 Améliorer les fonctions de récompense

Dans un premier temps, il a aussi été question d'améliorer les fonctions de récompense proposées par l'équipe de l'année précédente, notamment pour la prise en compte de la production de CO2 pour chaque véhicule. L'objectif était d'implémenter une meilleure prise en charge de la vitesse des véhicules, mais aussi de la production de CO2 pour chaque véhicule, lors des phases d'accélération/décélération.

Malheureusement la valeur de CO2 produite étant fournie par une API tierce dont on ne maitrisait pas le fonctionnement ni le mode de calcul, nous avons décidé de nous passer cette valeur dans la fonction de récompense. Le CO2 étant lié à l'accélération positives des véhicules, il sera indirectement contrôlé par le biais de cette grandeur, comme le montre ce graphique pris après une simulation sur le trajet d'un véhicule :

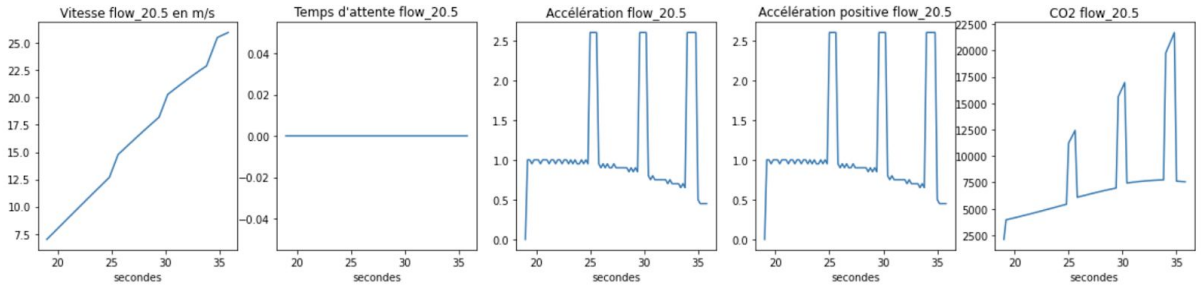


Figure : *Caractéristiques de trajet du véhicule "flow_20.5" dans l'environnement Issy-les-Moulineaux*

On constate sur ces graphiques, en effet, qu'il y a une corrélation forte entre les accélérations positives et le taux d'émissions de CO2 renvoyé par l'API. Pour obtenir ces graphiques, qui nous permettent d'étudier finement le comportement de chaque véhicule intervenant dans la simulation, nous avons dû implémenter des fonctions *ad hoc* que nous développerons par la suite dans la stratégie adoptée pour fiabiliser les résultats.

Lors des premiers entraînements, nous avons cherché à améliorer les résultats en faisant varier essentiellement les coefficients des différentes grandeurs utilisées pour le calcul de la fonction de récompense. Pour rappel, ces valeurs se concentraient autour de la vitesse des véhicules, leur temps d'attente et leur accélération. Après avoir testé près de vingt fonctions de récompense différentes, avec des temps de calculs de plusieurs heures à chaque fois et des résultats tangibles (Cf. V. Résultats et interprétations), notre effort s'est porté sur la définition d'une nouvelle approche dans la dimension des états possibles du système. Le but a été de réduire la dimension de cet espace. En effet, pour être efficaces, les apprentissages par renforcement font intervenir un très grand nombre de variables. Aussi pour réduire l'espace des possibles, nous avons dû développer, toujours à partir de la carte d'Issy-les-Moulineaux, un nouvel environnement (appelé Env2) se basant sur le nombre de

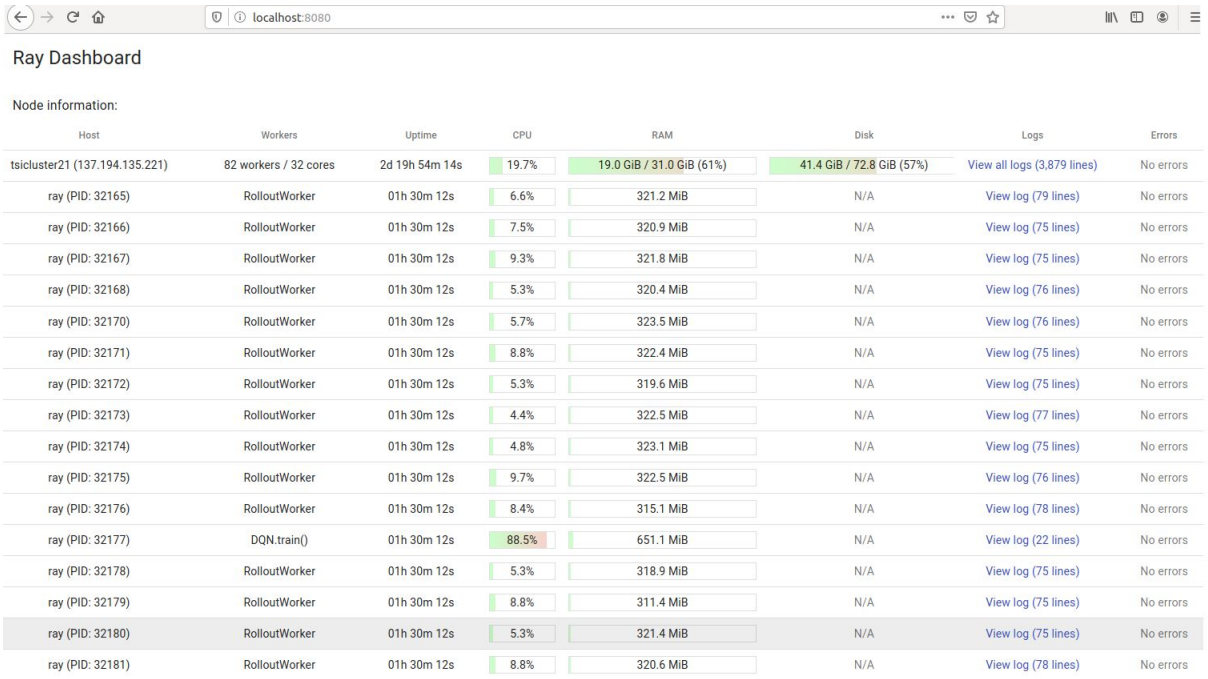
véhicules présents sur l'axe de circulation. Cette fonction récompensait les états du système qui permettaient une fluidification du trafic. Enfin, nous nous sommes concentrés sur l'étude d'un unique carrefour avec une approche différente de la fonction de récompense pour étudier le phénomène à un niveau microscopique. Cette fonction récompensait l'atteinte de la vitesse maximale dans le circuit (Cf. § II.4).

III.7 Déployer l'algorithme d'apprentissage sur des serveurs plus performants

L'apprentissage par renforcement étant extrêmement gourmand en ressources de calcul, nous avons rapidement constaté que nos postes de travail personnels n'étaient pas suffisants pour mener un apprentissage jusqu'au bout (le premier apprentissage avec 1000 itérations, en local, a demandé près de 3 jours de calculs). Nous avons donc déployé notre algorithme sur des infrastructures suffisamment puissantes pour voir les résultats que l'on souhaite assez rapides.

Pour cela, nous avons, dans un premier temps, testé d'installer l'ensemble de notre environnement de travail sur Google Colab Pro. Malheureusement, l'installation de l'ensemble des librairies nécessaires, leur interopérabilité en fonction des versions disponibles dans Colab mais aussi l'impossibilité intrinsèque de Google Colab de fournir l'interface graphique liée à SUMO ont soulevé davantage de difficultés et ont rendu impossible l'utilisation des ressources de calcul de Google Colab Pro.

Nous nous sommes donc concentrés sur le cluster de Telecom Paris, en faisant installer les librairies nécessaires. Bien que l'épidémie a quelque peu retardé le déploiement pleinement opérationnel de notre environnement de travail sur le cluster, nous avons pu commencer l'entraînement de nos algorithmes à partir de mai 2020. Les gains en temps de calculs ont été substantiels. Pour un même nombre d'itérations, il ne fallait plus que 4 heures de calculs contre 3 jours sur nos ordinateurs. Pour ce faire, l'environnement de simulation a été interfacé avec la librairie Ray qui est spécialisé dans la distribution parallèle des expériences pour le reinforcement learning en utilisant les CPUs a disposition puisque le temps de calcul est en majorité dû à la simulation des expériences de trafic routier. La librairie Ray met notamment à disposition un outil qui permet de surveiller l'état des différents CPUs *Workers* qui génèrent des données en lançant des simulations alors qu'un CPU principal fait tourner l'algorithme de renforcement au fur et à mesure que les données sont disponibles



Host	Workers	Uptime	CPU	RAM	Disk	Logs	Errors
tsicluster21 (137.194.135.221)	82 workers / 32 cores	2d 19h 54m 14s	19.7%	19.0 GiB / 31.0 GiB (61%)	41.4 GiB / 72.8 GiB (57%)	View all logs (3,879 lines)	No errors
ray (PID: 32165)	RolloutWorker	01h 30m 12s	6.6%	321.2 MiB	N/A	View log (79 lines)	No errors
ray (PID: 32166)	RolloutWorker	01h 30m 12s	7.5%	320.9 MiB	N/A	View log (75 lines)	No errors
ray (PID: 32167)	RolloutWorker	01h 30m 12s	9.3%	321.8 MiB	N/A	View log (75 lines)	No errors
ray (PID: 32168)	RolloutWorker	01h 30m 12s	5.3%	320.4 MiB	N/A	View log (76 lines)	No errors
ray (PID: 32170)	RolloutWorker	01h 30m 12s	5.7%	323.5 MiB	N/A	View log (76 lines)	No errors
ray (PID: 32171)	RolloutWorker	01h 30m 12s	8.8%	322.4 MiB	N/A	View log (75 lines)	No errors
ray (PID: 32172)	RolloutWorker	01h 30m 12s	5.3%	319.6 MiB	N/A	View log (75 lines)	No errors
ray (PID: 32173)	RolloutWorker	01h 30m 12s	4.4%	322.5 MiB	N/A	View log (77 lines)	No errors
ray (PID: 32174)	RolloutWorker	01h 30m 12s	4.8%	323.1 MiB	N/A	View log (75 lines)	No errors
ray (PID: 32175)	RolloutWorker	01h 30m 12s	9.7%	322.5 MiB	N/A	View log (76 lines)	No errors
ray (PID: 32176)	RolloutWorker	01h 30m 12s	8.4%	315.1 MiB	N/A	View log (78 lines)	No errors
ray (PID: 32177)	DQN.train()	01h 30m 12s	88.5%	651.1 MiB	N/A	View log (22 lines)	No errors
ray (PID: 32178)	RolloutWorker	01h 30m 12s	5.3%	318.9 MiB	N/A	View log (75 lines)	No errors
ray (PID: 32179)	RolloutWorker	01h 30m 12s	8.8%	311.4 MiB	N/A	View log (75 lines)	No errors
ray (PID: 32180)	RolloutWorker	01h 30m 12s	5.3%	321.4 MiB	N/A	View log (75 lines)	No errors
ray (PID: 32181)	RolloutWorker	01h 30m 12s	8.8%	320.6 MiB	N/A	View log (78 lines)	No errors

Figure : Le Ray Dashboard permet de surveiller l'état des CPUs utilisés pendant l'entrainement

IV. Méthodologie et problèmes rencontrés

Comme cela a été exposé plus haut, le projet s'appuie sur les travaux de l'équipe de l'année dernière et il convient ici de mentionner les difficultés rencontrées mais aussi les perspectives de développement et les échéances que nous nous sommes fixés.

IV.1 Problèmes rencontrés

Si les résultats obtenus par l'équipe précédente laissait présager une facilité de prise en main, nous permettant de nous concentrer sur les étapes suivantes pour améliorer l'existant, il s'est rapidement avéré que les informations à notre disposition étaient incomplètes.

Malgré la prévoyance de nos prédécesseurs qui avaient développé un docker dédié pour faire tourner leurs simulations et leur algorithme d'apprentissage, celui-ci n'était plus fonctionnel, faisant appel à des librairies trop anciennes.

Enfin, comme cela a été évoqué dans le § III.1, les différentes librairies nécessaires pour lancer les calculs ne sont pas maintenues à jour avec la même assiduité. Aussi, après avoir installés SUMO, FLOW et RLib en local, selon les tutoriels en ligne, il s'est avéré que le code implémenté disponible n'était plus compatible avec les libraires actuelles. Le code lui-même semblait incomplet ou tout du moins ne pas refléter exactement les résultats obtenus.

Heureusement, après avoir réussi à établir le contact avec l'un des étudiants de l'année dernière, nous avons pu organiser une session de travail en commun qui a été très éclairante. Nous avons notamment pu nous rendre compte que nous étions sur la bonne voie et qu'il nous manquait bien des éléments. En effet, le *repository* auquel nous avions accès depuis le début n'était pas à jour des derniers travaux. Bien que le membre de l'équipe de l'année dernière ait pris la peine de remettre à jour son repo et son docker, ce dernier n'a pas fonctionné. A quelques jours d'intervalle, une mise à jour d'une des librairies a été opérée rendant caduque les efforts de l'ancien étudiant.

Cependant grâce à notre entrevue productive, nous avons eu les pistes nécessaires pour reprendre les tutoriels de chacune des librairies et s'inspirer de leurs travaux et de leur code pour implémenter à nouveau tout un environnement de simulation mais aussi d'apprentissage par renforcement.

Si le projet devait être poursuivi l'année prochaine et malgré nos efforts didactiques pour assurer une prise en main simplifiée, les futurs étudiants ne pourront faire l'économie d'étudier avec attention les 14 tutoriels du projet FLOW pour commencer à maîtriser l'influence de chacun des paramètres sur les environnements de simulation. Fort heureusement, l'équipe de l'UC de Berkeley est réactive et toujours prête à aider.

IV.2 Agenda

Au regard des objectifs que nous nous étions fixés, comme exposés au § III, mais aussi des problèmes rencontrés, nous avons été en mesure d'élaborer une digramme de Gantt de nos travaux jusqu'à la présentation finale de notre projet, en juin 2020.

La première étape de notre projet a été de s'assurer que le modèle élaboré par l'équipe précédente était viable dans la durée. Aussi jusqu'à mi-février, l'équipe a apporté les modifications nécessaires au code pour tenter de le faire tourner, en prenant en compte les mises à jour de toutes les librairies. Le cas échéant, l'équipe aurait été contrainte de repartir du tout début du projet et d'envisager d'autres environnements de simulation et d'autres librairies, plus stables dans le temps. Cette éventualité aurait représenté la tâche "Développer un nouveau modèle de simulation" et aurait duré 90 jours.

Les objectifs décrits au § III sont repris dans la figure suivante :

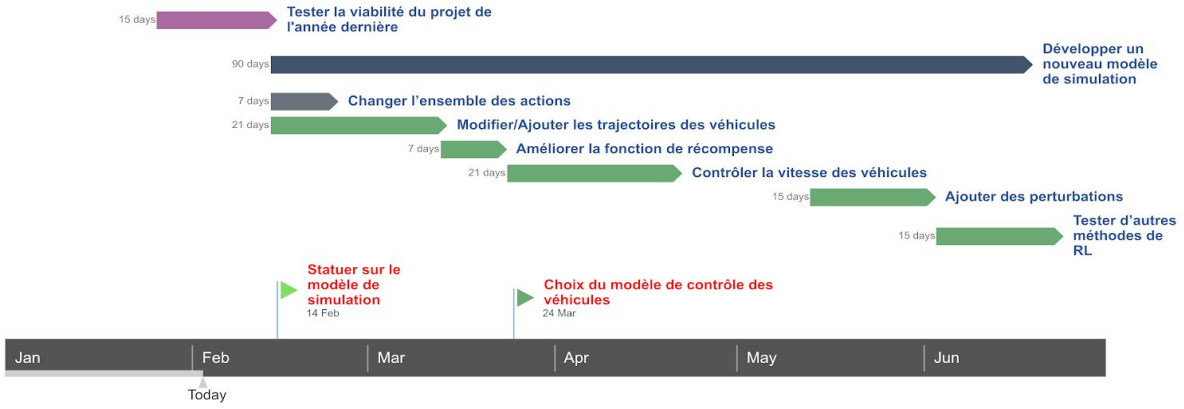


Figure : *Diagramme de Gantt du projet à la fin de la première période.*

A la fin du mois de mars 2020, après avoir réussi à reprendre l'ensemble du code de l'année dernière et apporter les modifications nécessaires pour le rendre opérant, la priorité a été donnée à l'élaboration du modèle de contrôle des véhicules. Il a été décidé, comme évoqué dans le § III.4, de ne se concentrer que sur les feux signalisations et non les véhicules autonomes pour contrôler le comportement des véhicules présents sur le circuit.

Figure : *Diagramme de Gantt du projet actualisé à la fin de la troisième période.*

La grande majorité des tâches relatives à la configuration de l'environnement de simulation ont été effectuées, sous réserve de petits ajustements comme l'abandon des perturbations. En revanche l'élaboration de nouveaux environnements et de nouvelles fonctions de récompense ont été réalisés. Nous nous sommes enfin concentrés sur un carrefour unique dans les dernières semaines du projet.

IV.3 Méthodologie

Quand bien même SUMO permet une visualisation des résultats et permet une comparaison visuelle du changement de comportement du système avant et après entraînement, cette interprétation reste subjective, d'autant plus lorsque le réseau routier est complexe. Savoir si le système a réellement été amélioré peut devenir difficile à mesurer si l'on s'appuie uniquement sur l'interface graphique fournie par SUMO.

Aussi, en plus de l'interprétation graphique, nous avons cherché à apporter des preuves tangibles de nos avancées. Pour cela nous avons défini une méthodologie qui tend à prouver de manière objective la plus value de notre système.

Dans un premier temps, nous nous sommes attachés à définir les caractéristiques des simulations sans entraînement. Cette phase consista à déterminer les valeurs moyennes des véhicules insérés dans le réseau au cours de la simulation :

- vitesse,
- accélération,
- temps d'attente.

Pour se faire, une fonction *ad hoc* a été développée et sera présentée dans le § V.1.

Partant de ces valeurs de référence, nous avons ensuite pu nous fixer une accélération moyenne, une vitesse moyenne et un temps d'attente optimaux à atteindre. Ainsi, si après entraînement, nous avions une vitesse moyenne supérieure mais des accélérations et des

temps d'attente en moyenne inférieurs, nous étions en mesure de conclure que notre système avait non seulement appris mais qu'en plus il avait été amélioré de manière tangible.

Après chaque entraînement, les résultats obtenus étaient comparés entre eux avec les valeurs de référence. Les coefficients de la fonction de récompense étaient ensuite mis à jour afin de tâcher d'obtenir l'optimum du système. La mise à jour des coefficients consistait à donner davantage d'importance à l'une ou l'autre des grandeurs (vitesse, accélérations ou temps d'attente) pour obtenir le comportement désiré. Cette étape a représenté la phase I de notre projet, après la phase préliminaire d'appropriation du code de l'équipe précédente. Avec cette méthode, en testant de nombreuses combinaisons de coefficients différents, on a réussi à améliorer légèrement la circulation comparée à un contrôle sans agent. Cependant, le résultat est loin d'être parfait puisque la solution retenue par l'agent est de bloquer longtemps l'un des axes secondaires et de laisser entièrement circuler les voitures sur l'autre perpendiculaire.

A la suite de ces premiers résultats, nous avons repensé notre approche du problème et avons cherché à en réduire ses dimensions. Comme exposé au § II.3, le nombre d'états possibles du système a été diminué en ne se concentrant à présent plus que sur le nombre de véhicules présents sur les axes de circulation. En revanche, le modèle de fonction de récompense utilisée en phase I a été conservé. Même si le temps a manqué pour explorer plus avant cette approche, les premiers résultats ont été encourageant. Les axes n'étaient plus bloqués par l'agent apprenant. Cela a représenté la phase II de notre projet.

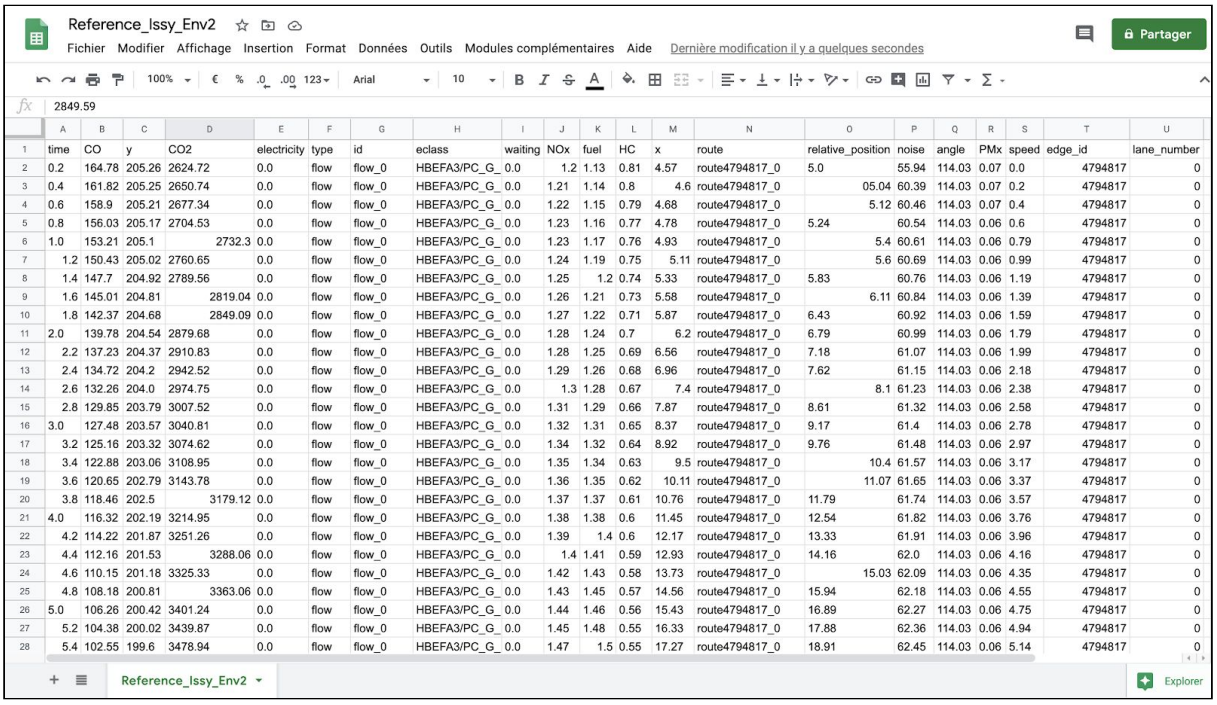
Pour finir, en phase III, nous nous sommes concentrés sur un unique carrefour avec la fonction de récompense du nouveau paradigme. Là aussi, faute de temps, nous avons juste eu le temps de mettre en évidence la réelle pertinence d'une telle approche avec des gains substantiels par rapport à toutes les grandeurs de référence (Cf. § V.4).

V. Résultats et interprétations

V.1 Développement des outils d'interprétation

Afin de pouvoir étudier de manière objective les résultats de nos différentes simulations, il a fallu, par rapport à l'équipe précédente, développer des fonctions permettant de calculer, à partir des fichiers .csv, produits lors des simulations, les accélérations de chaque véhicule et leur temps d'attente à chaque pas de la simulation.

SUMO permet en effet de récupérer, sous forme de fichiers .csv, pour peu que l'on ait rentré les bons paramètres, les caractéristiques de tous les véhicules passant dans le réseau routier au cours de la simulation. Les fichiers .csv, de plusieurs dizaines de milliers de lignes pour seulement quelques dizaines de secondes de simulation se présentent sous la forme suivante :



Reference_Issy_Env2

Fichier Modifier Affichage Insertion Format Données Outils Modules complémentaires Aide Dernière modification il y a quelques secondes

Partager

2849.59

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	time	CO	y	CO2	electricity	type	id	eclass	waiting	NOx	fuel	HC	x	route	relative_position	noise	angle	PMx	speed	edge_id	lane_number
2	0.2	164.78	205.26	2624.72	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.2	1.13	0.81	4.57	route4794817_0	5.0	55.94	114.03	0.07	0.0	4794817	0
3	0.4	161.82	205.25	2650.74	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.21	1.14	0.8	4.6	route4794817_0	05.04	60.39	114.03	0.07	0.2	4794817	0
4	0.6	158.9	205.21	2677.34	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.22	1.15	0.79	4.68	route4794817_0	5.12	60.46	114.03	0.07	0.4	4794817	0
5	0.8	156.03	205.17	2704.53	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.23	1.16	0.77	4.78	route4794817_0	5.24	60.54	114.03	0.06	0.6	4794817	0
6	1.0	153.21	205.1	2732.3	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.23	1.17	0.76	4.93	route4794817_0	5.4	60.61	114.03	0.06	0.79	4794817	0
7	1.2	150.43	205.02	2760.65	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.24	1.19	0.75	5.11	route4794817_0	5.6	60.69	114.03	0.06	0.99	4794817	0
8	1.4	147.7	204.92	2789.56	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.25	1.2	0.74	5.33	route4794817_0	5.83	60.76	114.03	0.06	1.19	4794817	0
9	1.6	145.01	204.81	2819.04	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.26	1.21	0.73	5.58	route4794817_0	6.11	60.84	114.03	0.06	1.39	4794817	0
10	1.8	142.37	204.68	2849.09	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.27	1.22	0.71	5.87	route4794817_0	6.43	60.92	114.03	0.06	1.59	4794817	0
11	2.0	139.78	204.54	2879.68	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.28	1.24	0.7	6.2	route4794817_0	6.79	60.99	114.03	0.06	1.79	4794817	0
12	2.2	137.23	204.37	2910.83	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.28	1.25	0.69	6.56	route4794817_0	7.18	61.07	114.03	0.06	1.99	4794817	0
13	2.4	134.72	204.2	2942.52	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.29	1.26	0.68	6.96	route4794817_0	7.62	61.15	114.03	0.06	2.18	4794817	0
14	2.6	132.26	204.0	2974.75	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.3	1.28	0.67	7.4	route4794817_0	8.1	61.23	114.03	0.06	2.38	4794817	0
15	2.8	129.85	203.79	3007.52	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.31	1.29	0.66	7.87	route4794817_0	8.61	61.32	114.03	0.06	2.58	4794817	0
16	3.0	127.48	203.57	3040.81	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.32	1.31	0.65	8.37	route4794817_0	9.17	61.4	114.03	0.06	2.78	4794817	0
17	3.2	125.16	203.32	3074.62	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.34	1.32	0.64	8.92	route4794817_0	9.76	61.48	114.03	0.06	2.97	4794817	0
18	3.4	122.88	203.06	3108.95	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.35	1.34	0.63	9.5	route4794817_0	10.4	61.57	114.03	0.06	3.17	4794817	0
19	3.6	120.65	202.79	3143.78	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.36	1.35	0.62	10.11	route4794817_0	11.07	61.65	114.03	0.06	3.37	4794817	0
20	3.8	118.46	202.5	3179.12	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.37	1.37	0.61	10.76	route4794817_0	11.79	61.74	114.03	0.06	3.57	4794817	0
21	4.0	116.32	202.19	3214.95	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.38	1.38	0.6	11.45	route4794817_0	12.54	61.82	114.03	0.06	3.76	4794817	0
22	4.2	114.22	201.87	3251.26	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.39	1.4	0.6	12.17	route4794817_0	13.33	61.91	114.03	0.06	3.96	4794817	0
23	4.4	112.16	201.53	3288.06	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.4	1.41	0.59	12.93	route4794817_0	14.16	62.0	114.03	0.06	4.16	4794817	0
24	4.6	110.15	201.18	3325.33	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.42	1.43	0.58	13.73	route4794817_0	15.03	62.09	114.03	0.06	4.35	4794817	0
25	4.8	108.18	200.81	3363.06	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.43	1.45	0.57	14.56	route4794817_0	15.94	62.18	114.03	0.06	4.55	4794817	0
26	5.0	106.26	200.42	3401.24	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.44	1.46	0.56	15.43	route4794817_0	16.89	62.27	114.03	0.06	4.75	4794817	0
27	5.2	104.38	200.02	3439.87	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.45	1.48	0.55	16.33	route4794817_0	17.88	62.36	114.03	0.06	4.94	4794817	0
28	5.4	102.55	199.6	3478.94	0.0	flow	flow_0	HBEFA3/PC_G_	0.0	1.47	1.5	0.55	17.27	route4794817_0	18.91	62.45	114.03	0.06	5.14	4794817	0

Reference_Issy_Env2

Explorer

Chaque véhicule est identifiable à travers son "id" et renvoie un certains nombre de valeurs, comme son émission de CO2, de CO, sa consommation électrique, sa position, sa vitesse, etc. Ces valeurs sont enregistrées à chaque "step" de la simulation, soit dans notre cas, toutes les 0.2 seconde. On constate que ni l'accélération ni les temps d'attente ne sont enregistrées par SUMO. Nous avons donc implémentés la fonction suivante pour récupérer ces valeurs :

```
# fonction calculant l'accélération des véhicules à chaque step
# et leur temps d'attente à chaque step
def get_acc_wait(df):
 df_acc = len(df) * [0]
 df_wait = [0.2 if v == 0 else 0 for v in df.speed]

 for i in range(1, len(df)):

   if (df.loc[i, 'id'] == df.loc[i-1, 'id']):

     d_v = df.loc[i, 'speed'] - df.loc[i-1, 'speed']
     d_t = df.loc[i, 'time'] - df.loc[i-1, 'time']
     acc = d_v / d_t

     df_acc[i] = acc
 df['acc'] = df_acc
 df['wait'] = df_wait
 df['acc_pos'] = df['acc'].apply(lambda x : x if x >= 0 else 0)
 return df
```

A partir ce ces données, l'enjeux a été de pouvoir en déduire les valeurs de référence mentionnées dans le § IV.3. Pour cela, la fonction suivante a été codée :

```
def get_properties(dict):
 i=0
 size = len(dict)

columns=['name','num_vhl','avg_speed','std_speed','avg_acc','std_acc'
,'avg_acc_pos','std_acc_pos', 'integer_acc_pos',
'avg_wait','std_wait']
 zero_data = np.zeros(shape=(size,len(columns)))
 df_results = pd.DataFrame(zero_data, columns=columns)

 for key, value in dict.items():
   df_results['name'][i] = key
   df_results['num_vhl'][i] = len(value.id.unique())
   df_results['avg_speed'][i] = np.mean(value.speed)
```

```
    df_results['std_speed'][i] = np.std(value.speed)
    df_results['avg_acc'][i] = np.mean(value.acc)
    df_results['std_acc'][i] = np.std(value.acc)
    df_results['avg_acc_pos'][i] = np.mean(value.acc_pos)
    df_results['std_acc_pos'][i] = np.std(value.acc_pos)
    df_results['integer_acc_pos'][i] = np.sum(value.acc_pos) * 0.2

    # Calcul du temps moyen d'attente par vhl
    list_vhl_id  = value.id.unique()
    list_wait_vhl = []
    for vhl in list_vhl_id:
      list_wait_vhl.append(np.sum(value[(value['id']== vhl)].wait))

    df_results['avg_wait'][i] = np.mean(list_wait_vhl)
    df_results['std_wait'][i] = np.std(list_wait_vhl)
    i+=1

  return df_results
```

Enfin, nous avons développé un ensemble de fonctions permettant d'étudier dans le détail le comportement de chaque véhicule après chaque entraînement afin de mieux se rendre compte des phénomènes observés au niveau macroscopique (Cf. fichier "Etude_csv.ipynb" en annexe). Ces fonctions rendent notamment possible les observations suivantes, déjà présentées au § III.5 :

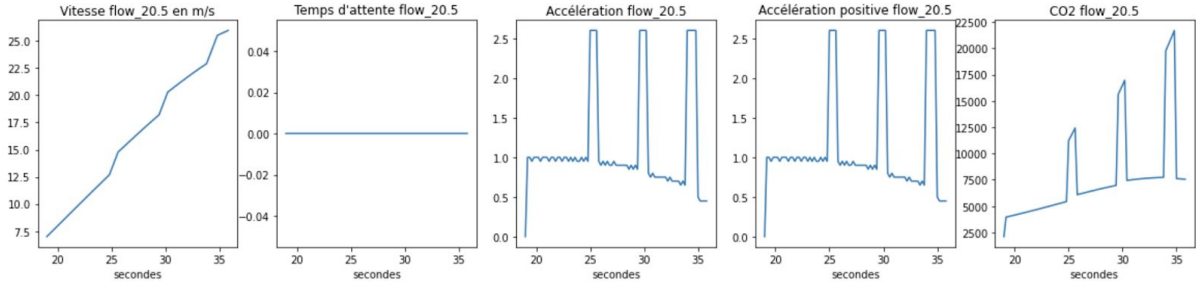


Figure : *Caractéristiques de trajet du véhicule "flow_20.5" dans l'environnement Issy-les-Moulineaux*

❖ Tensorboard :

Il est aussi possible de suivre l'entraînement des modèles en temps réel par le biais d'une interface **tensorboard**. Elle permet d'observer des métriques propres à l'entraînement du modèle (émissions de CO2, accélérations, vitesse, récompense, etc.).

Cet outil a notamment permis de mettre en évidence un gain de temps et une évolution substantielle des performances d'apprentissage du modèle sur les clusters de calcul de l'école, où l'on peut observer la croissance de la fonction de de récompense.

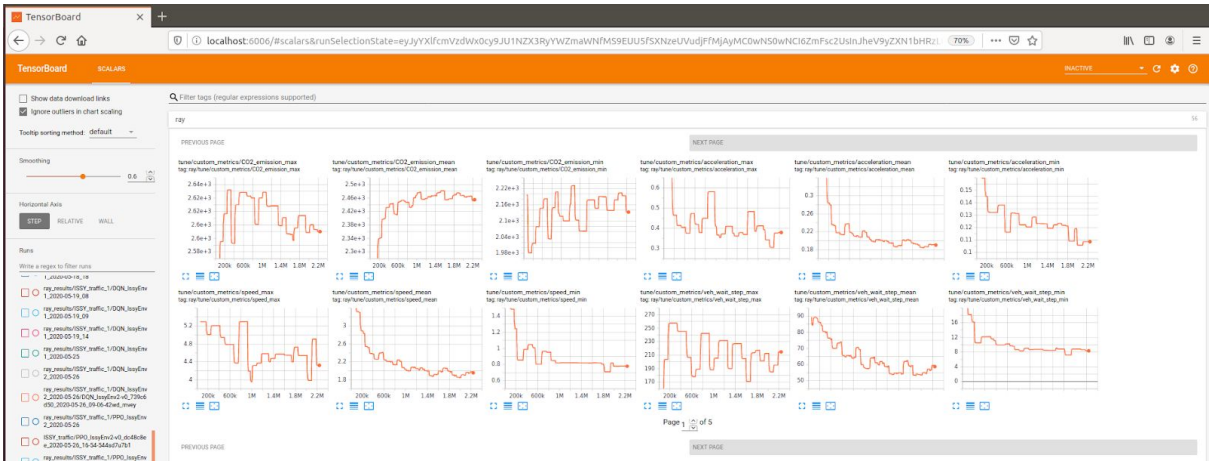


Figure: Exemple de l'interface tensorboard

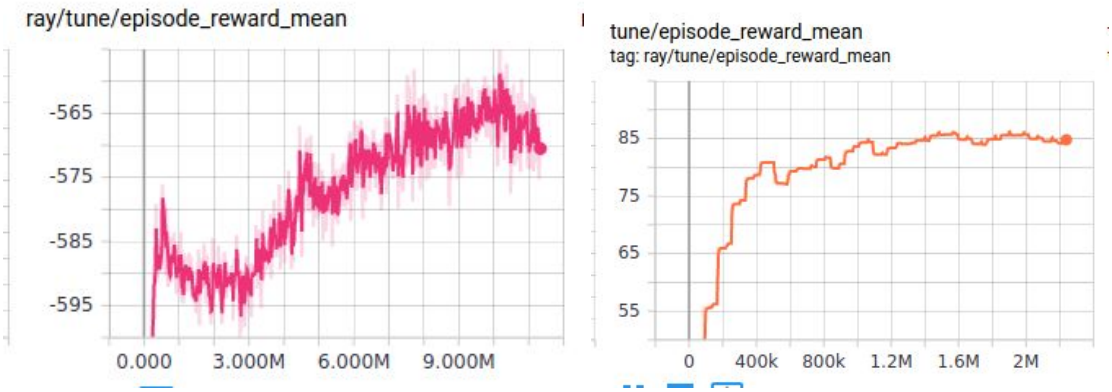


Figure: Exemple de visualisation de l'évolution de la fonction de récompense

V.2 Résultats sur le quartier d'Issy-les-Moulineaux

Dans un premier temps, nous nous attacherons à présenter les résultats obtenus après plusieurs entraînements de modèles lors de la phase I du projet. Puis, nous présenterons ceux obtenus en Phase II.

Afin de permettre une visualisation pertinente des résultats obtenus sous forme d'images animées, nous mettons volontairement ces deux images sur la même page.

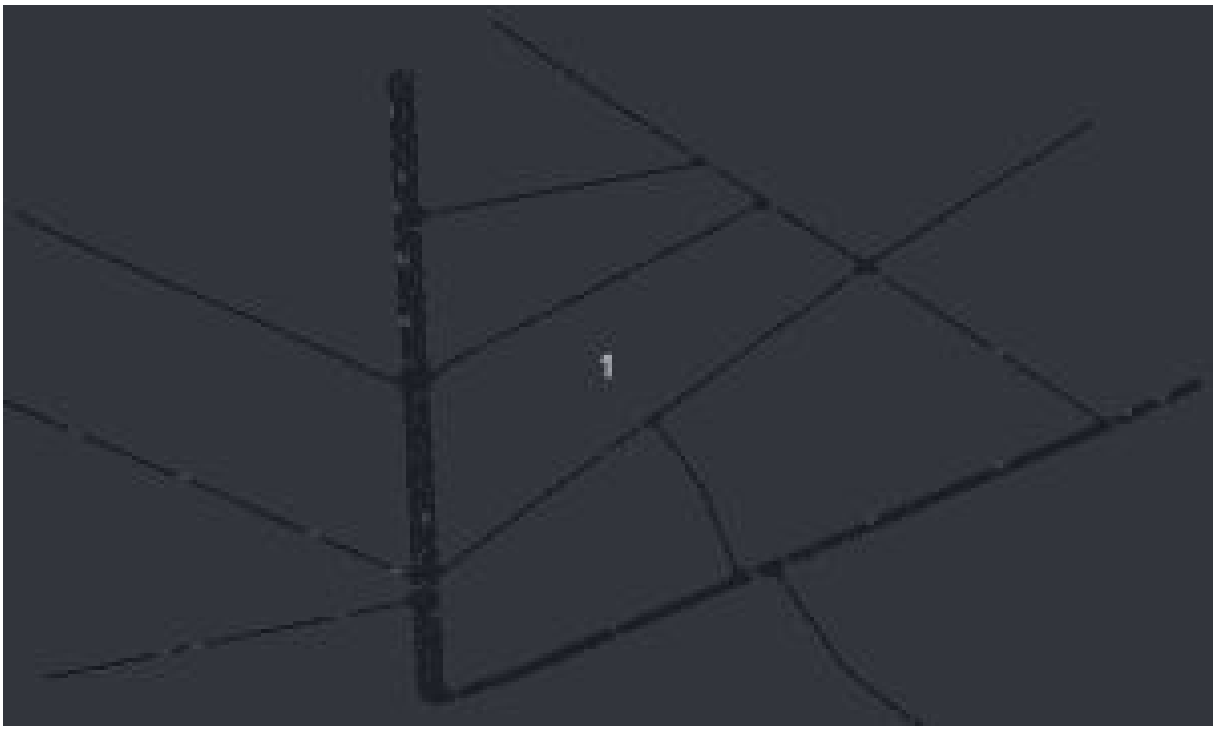


Figure : *Visualisation graphique d'une simulation sans entraînement de l'environnement d'Issy-les-Moulineaux.*

Figure : *Visualisation graphique d'une simulation avec entraînement de l'environnement d'Issy-les-Moulineaux.*

Grâce aux fonctions implémentées et présentées au § V.1, nous avons été en mesure, pour la première phase de notre projet, comme expliqué dans le § IV.3, d'obtenir les résultats suivants :

	name	num_vhl	avg_speed	std_speed	avg_acc	std_acc	avg_acc_pos	std_acc_pos	integer_acc_pos	avg_wait	std_wait
0	Reference_ISSY_traffic_20200526-emission.csv	113.0	4.355959	4.860218	0.035767	1.173870	0.209399	0.456113	2334.63	30.768142	25.684768
1	emission_04_06_2020_IssyEnv2_PPO.csv	199.0	4.920693	6.032023	0.081567	1.235075	0.194678	0.757241	3221.80	42.101508	74.377105
2	emission_04_06_2020_IssyEnv1_DQN.csv	198.0	4.775912	5.949407	0.082179	1.054991	0.191639	0.583288	3182.43	41.237374	72.051985
3	emission_2020-05-27_PPO_Env2.csv	170.0	3.350837	4.303498	-0.029221	0.947444	0.157592	0.415024	2733.68	40.301176	30.002590
4	emission_2020-05-27_PPO_Env1.csv	177.0	4.710275	4.891135	0.006330	1.064542	0.224132	0.472840	3239.06	24.047458	28.431152
5	emission_2020-05-27_DQN_Env2.csv	172.0	4.852349	5.998469	0.087951	0.891933	0.196068	0.473308	2536.96	38.272093	62.704604
6	emission_2020-05-26_01.csv	173.0	4.894853	5.876061	0.083804	1.008399	0.196394	0.519449	2585.45	35.350289	59.632703
7	emission_2020-05-26.csv	177.0	5.010020	4.981883	0.018152	1.221236	0.243055	0.748918	3429.31	20.389831	25.776074
8	emission_2020-05-17.csv	124.0	6.157330	5.440625	0.092947	1.691829	0.325385	0.787538	2942.20	13.945161	22.078268
9	emission_2020-05-16.csv	120.0	2.963572	4.764241	-0.025346	1.088207	0.126698	0.534290	1618.06	63.326667	54.244883
10	emission_2020-05-18_08.csv	118.0	6.972740	6.300048	0.136412	1.711268	0.293788	0.813897	2330.97	21.596610	54.555001
11	emission_2020-05-18_17.csv	130.0	7.146200	5.557183	0.103944	1.501048	0.309401	0.912367	2917.96	15.196923	40.146424
12	emission_2020-05-18_18-05.csv	105.0	2.110771	4.186081	-0.042394	0.753931	0.070730	0.270666	829.64	81.440000	59.926523
13	emission_2020-05-18_18.csv	109.0	6.894426	5.958092	0.107325	1.563322	0.272124	0.898148	2187.39	21.242202	54.228136
14	emission_2020-05-19_08.csv	98.0	6.661893	5.457078	0.078276	1.540464	0.280863	0.644876	2459.80	20.218367	35.417453
15	emission_2020-05-19_09.csv	112.0	2.537466	4.347874	-0.038118	0.792621	0.096597	0.328123	1186.00	70.587500	61.284706
16	emission_2020-05-19_14.csv	114.0	2.542347	4.355909	-0.040148	0.849877	0.100811	0.472848	1206.24	66.629825	56.832874
17	emission_2020-05-19_14_checkpoint220.csv	97.0	2.462371	4.294976	-0.038128	0.835206	0.089528	0.497388	951.57	72.993814	59.128129

Figure : *Tableau des résultats des différentes simulations de la phase I.*

La première ligne du tableau représente nos données de référence. La légende des colonnes est la suivante :

- "`name`" correspond au nom de l'entraînement avec une certaine fonction de récompense, à une date précise ;
- "`num_vhl`" correspond au nombre total de véhicules insérés dans le réseau le temps de la simulation ;
- "`avg_speed`" représente la vitesse moyenne de l'ensemble des véhicules lors de la simulation ;
- "`std_speed`" est l'écart type des vitesses ;
- "`avg_acc`" et "`std_acc`", de même, correspondent aux valeurs moyennes et à l'écart type des accélérations ;
- "`avg_acc_pos`" et "`std_acc_pos`" correspondent aux valeurs moyennes et à l'écart-type des accélérations uniquement positives ;
- "`integer_acc_pos`" est l'intégrale de l'ensemble des accélérations positives sur l'ensemble de la simulation. Cette valeur est la principale mesure corrélative des émissions de CO2 ;
- "`avg_wait`" correspond au temps d'attente moyen de l'ensemble des véhicules et "`std_wait`" l'écart-type.

Au regard des objectifs fixés, qui sont une vitesse moyenne supérieure, des temps d'attente inférieurs mais aussi une intégrale des accélérations positives inférieure, par rapport aux valeurs de référence, on peut constater que le modèle obtenu à partir de la fonction de récompense du 18 mai 2020 (Cf. "emission_2020-05-18_08.csv") est le meilleur. De manière plus visuelle, on peut s'intéresser au graphique ci-dessous qui présente les résultats normalisés et classés selon leur vitesse moyenne du tableau suivant :

	name	avg_speed	avg_acc	avg_acc_pos	integer_acc_pos	avg_wait
0	Reference_ISSY_traffic_20200526-emission.csv	0.445878	0.437123	0.544535	0.578916	0.249248
1	emission_2020-05-18_17.csv	1.000000	0.818418	0.937231	0.803302	0.018546
2	emission_2020-05-18_08.csv	0.965552	1.000000	0.875919	0.577508	0.113363
3	emission_2020-05-18_18.csv	0.949999	0.837325	0.790850	0.522278	0.108113
4	emission_2020-05-19_08.csv	0.903820	0.674866	0.825167	0.627064	0.092943
5	emission_2020-05-17.csv	0.803617	0.756912	1.000000	0.812626	0.000000
6	emission_2020-05-26.csv	0.575770	0.338609	0.676698	1.000000	0.095484
7	emission_04_06_2020_IssyEnv2_PPO.csv	0.558030	0.693271	0.486726	0.920178	0.417163
8	emission_2020-05-26_01.csv	0.552899	0.705778	0.493466	0.675397	0.317137
9	emission_2020-05-27_DQN_Env2.csv	0.544458	0.728971	0.492185	0.656745	0.360427
10	emission_04_06_2020_IssyEnv1_DQN.csv	0.529278	0.696692	0.474793	0.905034	0.404360
11	emission_2020-05-27_PPO_Env1.csv	0.516243	0.272493	0.602388	0.926818	0.149675
12	emission_2020-05-27_PPO_Env2.csv	0.246268	0.073671	0.341094	0.732416	0.390489
13	emission_2020-05-16.csv	0.169360	0.095339	0.219778	0.303277	0.731634
14	emission_2020-05-19_14.csv	0.085708	0.012557	0.118121	0.144865	0.780573
15	emission_2020-05-19_09.csv	0.084739	0.023914	0.101575	0.137079	0.839210
16	emission_2020-05-19_14_checkpoint220.csv	0.069825	0.023856	0.073814	0.046902	0.874862
17	emission_2020-05-18_18-05.csv	0.000000	0.000000	0.000000	0.000000	1.000000

Figure : *Tableau des résultats normalisés et classés selon leur vitesse moyenne des différentes simulations de la phase I (les résultats de référence ont volontairement été laissés à la ligne 0 pour faciliter les comparaisons)*

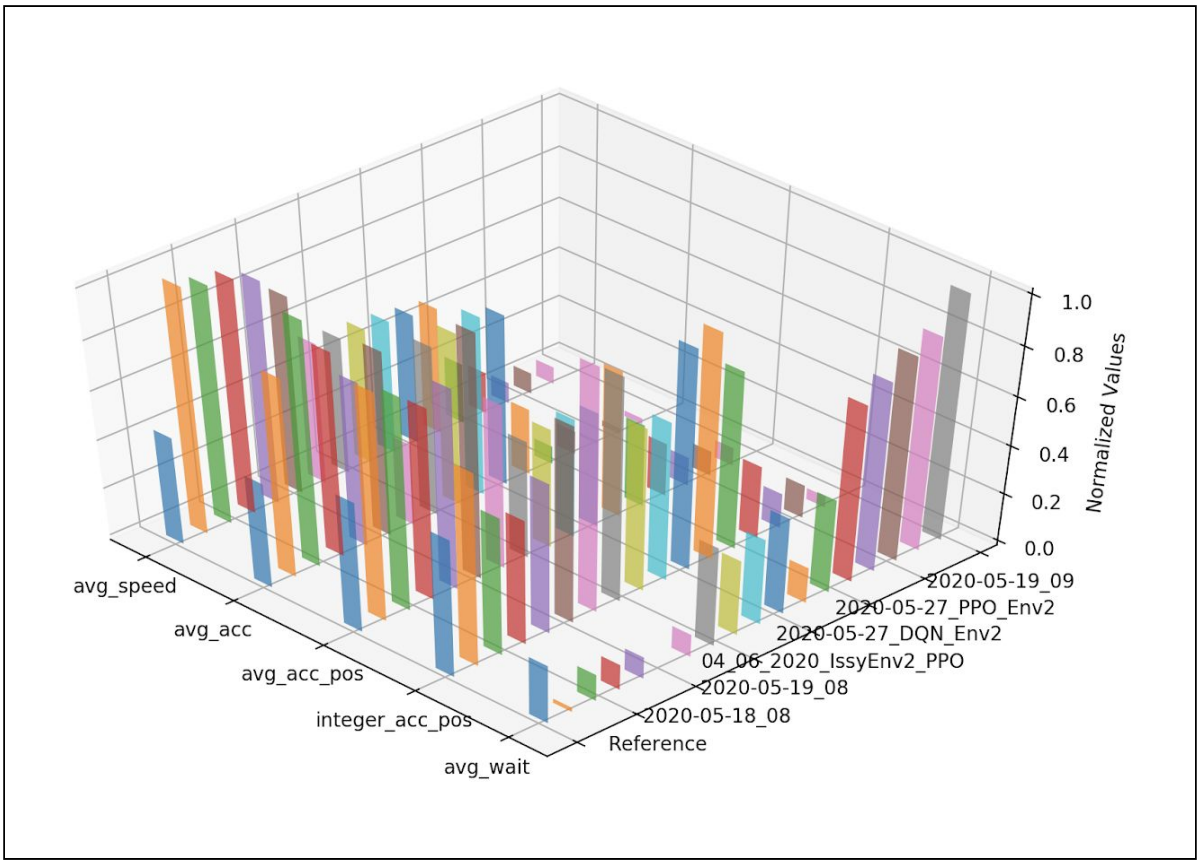


Figure : *Visualisation graphique des résultats normalisés et classés selon leur vitesse moyenne des différentes simulations de la phase I (les résultats de référence ont volontairement été laissés au premier plan pour faciliter les comparaisons)*

On constate 2 enseignements majeurs. Le premier est que la simulation correspondant au fichier “emission_2020-05-18_08.csv” a sensiblement amélioré l’état du système (en rouge sur le graphique en 4e position). On constate aussi que l’intégrale des accélérations positives est difficilement optimisable dans les conditions de la simulation. En effet, on a pu constater qu’en mettant de forts coefficients sur ce paramètre dans notre fonction de récompense, le système finissait par mettre les véhicules à l’arrêt, ce que l’on retrouve dans ce graphique avec les colonnes des derniers plans. On verra par la suite comment nous pouvons interpréter ces résultats.

Pour la phase II, phase où nous avons changé de paradigme pour l’espace des états possibles, nous avons obtenu les résultats suivants :

	name	avg_speed	avg_acc	avg_acc_pos	integer_acc_pos	avg_wait
0	Reference_Issy_Env2.csv	0.687858	0.458145	0.654680	0.613917	0.866181
1	emission_2020-06-17_PPO_Env2_no_constraint.csv	0.725845	0.888877	0.755907	0.789371	0.499731

Figure : *Tableau des résultats des simulations avant et après entraînement.*

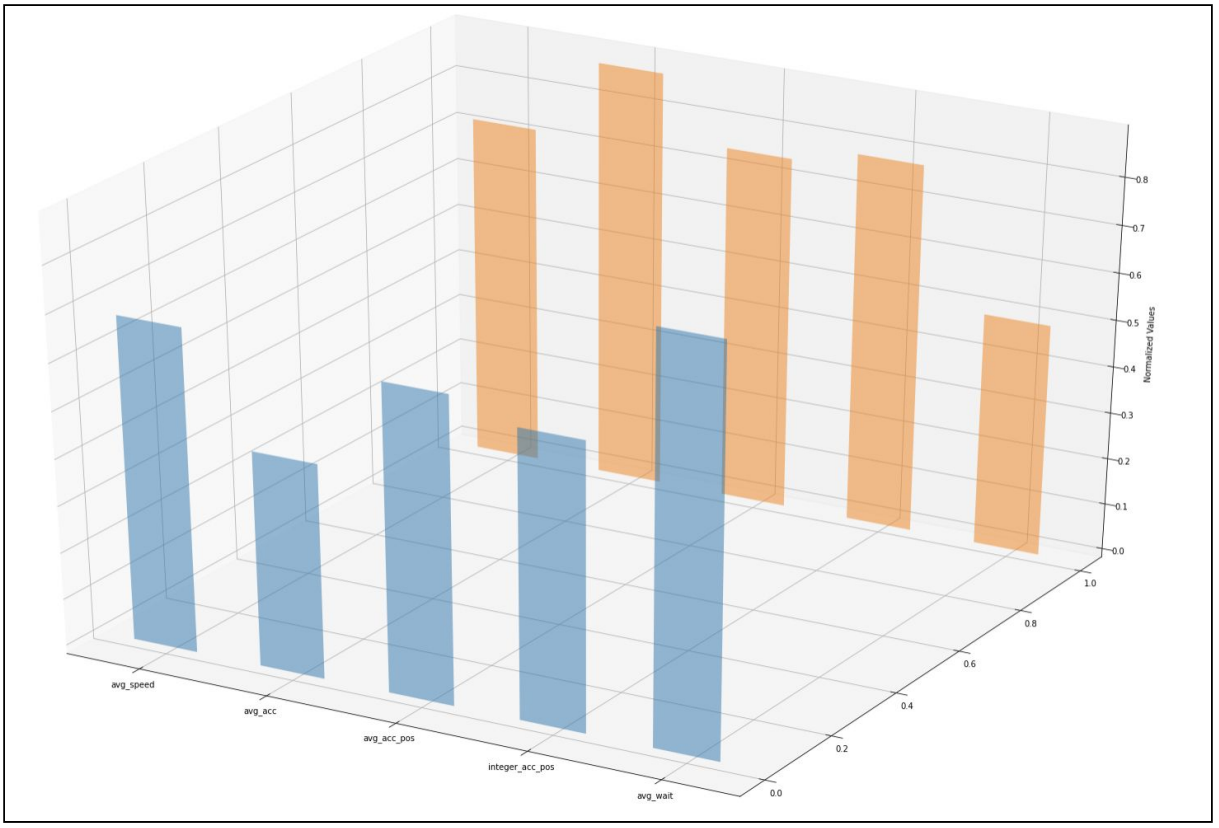


Figure : *Visualisation graphique des résultats normalisés de la simulation de la phase II (les résultats de référence ont volontairement été laissés au premier plan pour faciliter les comparaisons)*

Si le temps a manqué pour explorer plus avant cette voie, il apparaît que les temps d'attente ont été sensiblement améliorés, la vitesse moyenne légèrement et, en revanche, on retrouve la même problématique concernant l'intégrale des accélérations positives. Les résultats sont encourageants et mériteraient d'être approfondis à l'avenir.

V.3 Interprétations

Les résultats obtenus précédemment sont plus qu'encourageant et démontrent qu'avec seulement des feux de signalisation autonomes, capables d'apprendre, il est possible, au moins de manière théorique, de fluidifier significativement le trafic routier d'une ville de taille moyenne. Pour autant, ce qui surprend est l'absence de résultats probants en termes d'émissions de CO2 qui ont plutôt eu tendance à augmenter malgré le gain de vitesse enregistré.

Pour l'expliquer, il faut rappeler ici les principes de la modélisation sous SUMO. Tout d'abord, le comportement des véhicules a été conçu, dans nos simulations, de manière à refléter un comportement humain. Ensuite, les contraintes imposées à notre système comme l'intervalle de temps entre les changements d'état des feux de signalisation ou encore la vitesse limite de chaque axe a été imposé de manière arbitraire, en cherchant à

représenter le plus fidèlement possible la réalité. Enfin, on l'a vu, SUMO ne gère pas directement les accélérations des véhicules mais seulement leur vitesse. Le fait de devoir passer par une fonction tierce n'est pas vraisemblablement optimal pour agir sur ce paramètre pourtant important pour les émissions de CO2. La principale conclusion à laquelle nous sommes arrivés est que vraisemblablement, après chaque passage au vert des feux de signalisation, les véhicules au comportement humain ont tâché d'atteindre, en un minimum de temps, la vitesse maximale autorisée sur chaque axe. Cela a engendré des accélérations d'autant plus brutales que le trafic a été fluidifié. De plus, avec la multiplication des noeuds routiers et les états possibles du système liés, des phénomènes difficilement appréhendables, à première vue, peuvent expliquer cette situation.

Aussi, avec des agents de contrôle pouvant influencer sur les accélérations, il est vraisemblable que l'on pourrait aussi améliorer la valeur de l'intégrale des accélérations positives et donc réduire les émissions de CO2. Pour cela, la généralisation de véhicules autonomes capables d'appliquer des accélérations continues modérées permettrait d'atteindre cet objectif. Faute de temps, nous n'avons pas été en mesure d'explorer cette voie qui n'est, de toute manière, pas réaliste à court termes.

V.4 Résultats au niveau du carrefour

Ici, nous nous attacherons à présenter les résultats obtenus en phase III du projet, en se concentrant sur la modélisation d'un carrefour unique et reprenant la fonction de récompense élaborée en phase II de notre projet.

Pour ce carrefour, il est important de signaler ici que l'environnement créé représente une intersection de nuit avec un flux de véhicules moins dense qu'en journée. Alors que dans les simulations précédentes, les flux de véhicules étaient relativement denses, ici, la densité de véhicule a volontairement été réduite.

Afin de permettre une visualisation pertinente des résultats obtenus sous forme d'images animées, nous mettons volontairement ces deux images sur la même page.

Figure : *Visualisation graphique d'une simulation sans entraînement d'un carrefour.*

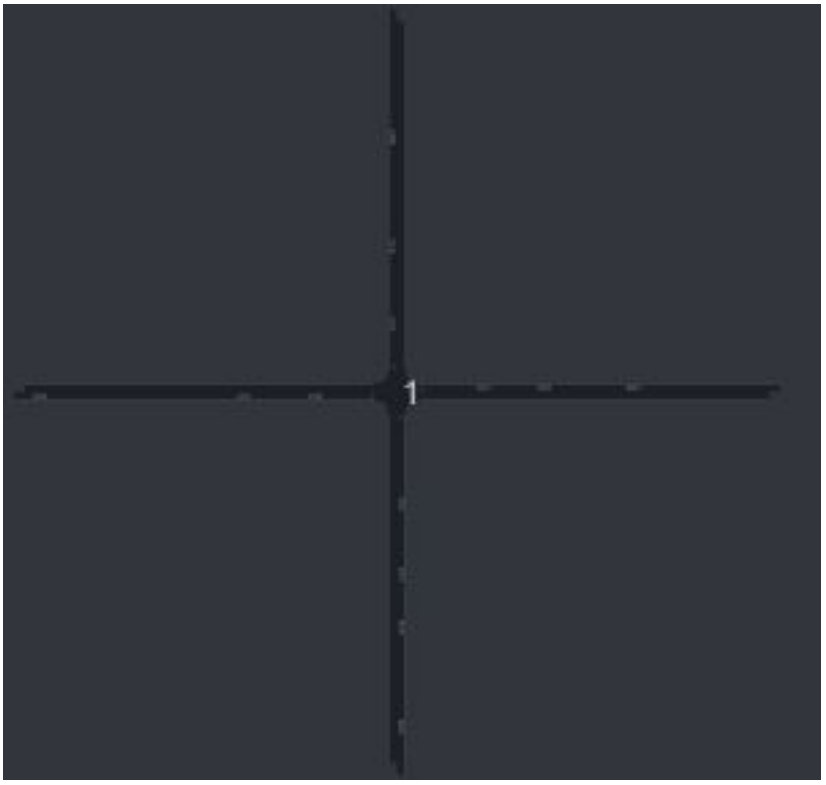


Figure : *Visualisation graphique d'une simulation avec entraînement d'un carrefour.*

Avec cette approche, nous avons obtenu les résultats suivants :

	name	num_vhl	avg_speed	std_speed	avg_acc	std_acc	avg_acc_pos	std_acc_pos	integer_acc_pos	avg_wait	std_wait
0	Reference_Carrefour.csv	47.0	3.486193	3.162719	0.043568	0.466346	0.092471	0.366107	622.22	58.906383	80.104356
1	traffic_light_grid_2020-18-06.csv	56.0	5.441500	2.232415	0.068073	0.578443	0.139984	0.468918	372.33	4.028571	7.534248

Figure : *Tableau des résultats des simulations avec et sans entraînement au niveau du carrefour.*

Non seulement tous les objectifs ont été atteints comme le prouve le tableau ci-dessus (une vitesse moyenne supérieure, mais aussi une intégrale des accélérations positives et un temps d'attente moyen sensiblement inférieurs) mais en plus les temps de calculs ont été beaucoup plus rapide pour obtenir un optimum. Cela s'explique par une complexité du système largement simplifiée.

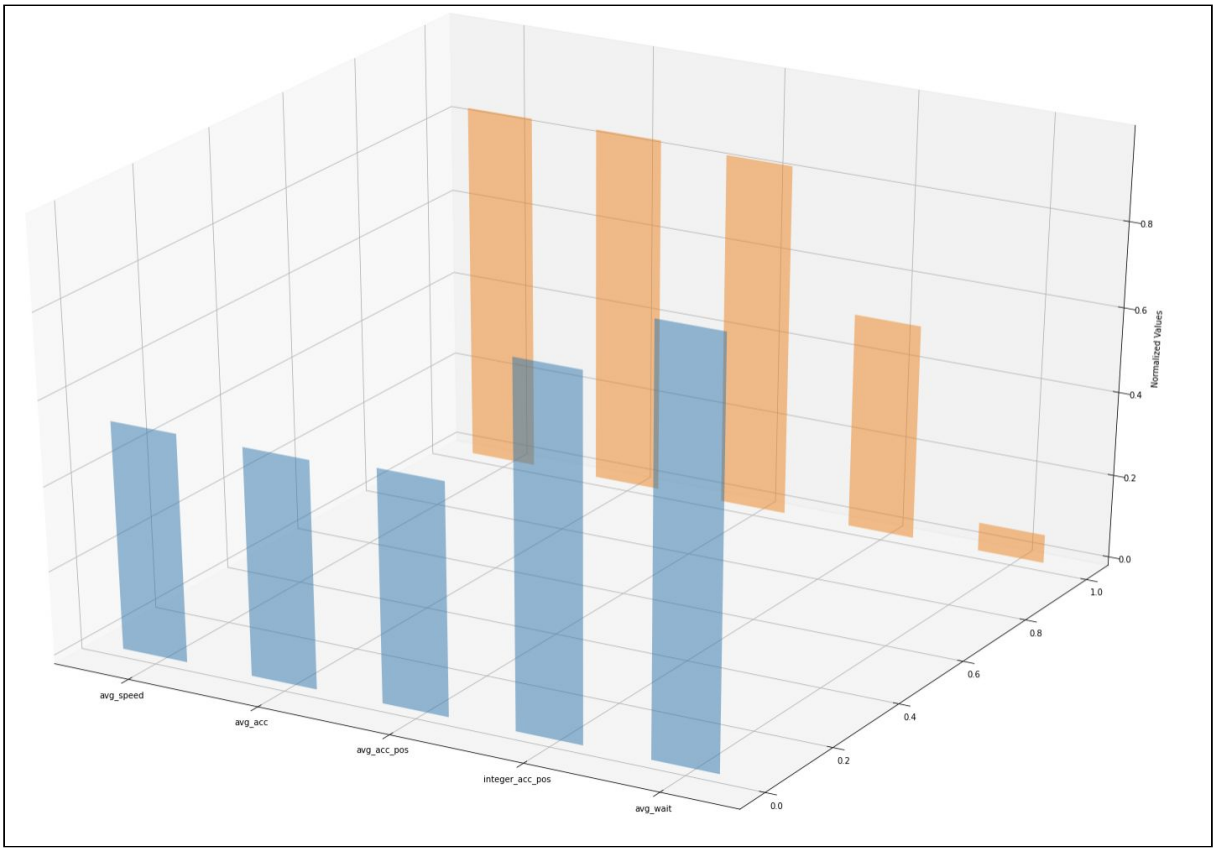


Figure : *Visualisation graphique des résultats normalisés de la simulation de la phase III (les résultats de référence ont volontairement été laissés au premier plan pour faciliter les comparaisons)*

De manière visuelle, on constate une amélioration nette du système.

V.5 Interprétations

La simplification du système ainsi que de la fonction de récompense, déjà initiée lors de la phase II, ont permis d'obtenir des résultats nettement meilleurs. Cela laisse présager un passage à l'échelle plus simple. Il conviendra néanmoins d'étudier le système de manière macroscopique en augmentant sensiblement le nombre de noeuds routiers et étudiant plusieurs densités de véhicules possibles sur chaque axe pour en valider sa pertinence.

Concernant l'amélioration au niveau des accélérations des véhicules dans ce cas, on peut avancer qu'en réduisant drastiquement le nombre de véhicules mis à l'arrêt la très grande majorité est en mesure d'avoir une vitesse constante ce qui réduit d'autant les accélérations intempestives. Cela est rendu possible par le fait que le système n'est conditionné par aucun autre noeud routier, contrairement à ce que nous avons pu expérimenter avec le quartier d'Issy-les-Moulineaux.

Conclusion

Le projet fil rouge CIL4SYS avec de l'apprentissage par renforcement pour l'optimisation du trafic routier a eu pour objectif de le fluidifier tout en réduisant les émissions de CO_2, grâce à un agent contrôlé par une intelligence artificielle.

Pour réaliser ce projet nous sommes partis du travail d'une équipe de l'année précédente et nous avons choisi de nous inscrire dans la continuité des travaux de cette équipe, notamment dans le choix de la technique d'apprentissage par renforcement utilisée et dans le choix des librairies et des outils, dont FLOW et SUMO.

Après avoir maîtrisé le fonctionnement et le paramétrage des environnements et des agents de l'apprentissage par renforcement, il a été question pour nous de les entraîner dans des simulations réalistes et de comparer les résultats obtenus à travers diverses méthodes.

L'un des enseignements majeur, en termes académiques, pour l'équipe de cette année, est que l'apprentissage par renforcement exige la génération d'énormément de données et de maîtriser un grand nombre de paramètres (au niveau des algorithmes comme des environnements). Si nous avons tâché, tout au long du projet, d'isoler chacun d'entre eux pour en comprendre leur rôle et leur portée dans l'optimisation du système, il convient de signaler que notre étude n'a sans doute pas encore pu explorer toutes les potentialités offertes.

Surtout, en termes plus opérationnels, les objectifs atteints à la fin du projet sont très prometteurs et permettent, *a minima*, de démontrer qu'avec seulement des feux de signalisations intelligents il est possible de fluidifier significativement l'état du trafic routier en permettant une vitesse moyenne supérieure et un temps d'attente inférieur. En revanche, malgré plusieurs modélisations pour la fonction de récompense, il a été très difficile de réduire les émissions de CO2 pour des environnements complexes. Il est probable que cela est dû à un comportement trop humain des véhicules. En ayant des accélérations intempestives de manière à atteindre au plus vite les limitations de vitesse du réseau routier, ils génèrent davantage de CO2. Cela pourrait être corrigé à termes avec la généralisation de véhicules autonomes qui auraient été paramétrés pour avoir des accélérations plus douces et éco-responsables. Faute de temps, cela n'a pas pu être exploré par l'équipe actuelle.

En revanche, une nouvelle étude, simplifiée, au niveau d'un unique carrefour a permis d'étudier plus finement le comportement du système et d'obtenir des résultats probants. Il conviendra de s'en inspirer pour les études futures et les passages à l'échelle pour une ville de taille modérée.

Enfin, au regard des difficultés éprouvées lors du projet concernant l'interopérabilité de toutes les librairies nécessaires à maîtriser, il conviendra de garder à l'esprit qu'il pourra être nécessaire de trouver un autre outil de simulation ou une autre librairie pour gérer

l’apprentissage par renforcement, car FLOW ne semble plus prendre en charge les dernières versions de SUMO.

Bibliographie

1. WU, Cathy, KREIDIEH Abdul Rahman, PARVATE Kanaad, VINITSKY Eugene. "*Flow: A Modular Learning Framework for Autonomy in Traffic*", UC Berkeley. **2019.**

2. WU, Cathy. *Simulating the Future of Traffic with RL*. UC Berkeley. Avril **2020**.

3. BAYEN, Alexandre. *The Future of Mixed-Autonomy Traffic.* UC Berkeley. September **2019**.

4. STEVENS, M. YEH, C. *Reinforcement Learning for Traffic Optimization.* Stanford. **2016.**

5. AULT TEXAS James, SHARON Guni, Josiah P. Hanna. *Learning an Interpretable Traffic Signal Control Policy.* A&M University & University of Edinburgh. Feb **2020**.

6. Song Sang Koh, *Reinforcement Learning, SUMO, and Complex Urban Traffic Management.* Jan **2018** .

7. SUTTON R. *Policy Gradient Methods for Reinforcement Learning with Function Approximation.* AT&T Labs-Research. **2000.**

8. MNIH V*. Playing Atari with Deep Reinforcement Learning.* DeepMind Technologies. **2013**

9. SCHULMAN, J. *Proximal Policy Optimization Algorithms*. OpenAI. ***2017***

10. ZHENG, G. *Diagnosing Reinforcement Learning for Traffic Signal Control.* Pennsylvania State University **2019**

11. *« Control Strategies for a Stochastic Planner » [archive], sur www.aaai.org (consulté le 20 Juin 2020)*

12. *Benchmarks for reinforcement learning in mixed-autonomy traffic Eugene Vinitsky, Aboudy Kreidieh, Luc Le Flem, Nishant Kheterpal, Kathy Jang, Cathy Wu, Fangyu Wu, Richard Liaw, Eric Liang, Alexandre Bayen.* <*http://proceedings.mlr.press/v87/vinitsky18a/vinitsky18a.pdf*> *Conference on Robot Learning, 2018* [page consultée le 18 Juin 2020]

13. [Projet] "Smart-Traffic-Signals-in-India-using-Deep-Reinforcement-Learning-and-Advanced-Computer-Vision" <https://github.com/Ujwal2910/Smart-Traffic-Signals-in-India-using-Deep-Reinforcement-Learning-and-Advanced-Computer-Vision> [page consultée le 18 Juin 2020]

14. [Projet] “DDQN-for-traffic-control” <https://github.com/sebxwolf/DDQN-for-traffic-control> [page consultée le 18 Juin 2020]

15. Guanjie Zheng, Xinshi Zang, Nan Xu, Hua Wei Zhengyao Yu,, Vikash Gayah, Kai Xu, Zhenhui Li. *Diagnosing Reinforcement Learning for Traffic Signal Control.* Pennsylvania State University, Shanghai Jiao Tong Univerisity,Shanghai Tianrang Intelligent Technology Co., Ltd .May 2019