UNIVERSITÉ NATIONALE DU VIETNAM À HANOÏ

INSTITUT DE LA FRANCOPHONIE POUR L'INNOVATION

Option : Systèmes Intelligents et Multimédia (SIM)

Promotion : XXII

Rapport de Système Multi-Agent(SMA) et Intelligence Artificielle(IA)

Conception & implémentation du traffic dans un carrefour : cas de la ville de Hanoi

Rédigé par :

OUBDA Raphaël Nicolas Wendyam

Encadrant :

Dr. NGUYEN Manh Hung

Année académique : 2017 - 2018

Table des matières

1 Introduction générale

Un système multi-agents (SMA) est un système composé d'un ensemble d'agents (un processus, un robot, un être humain, etc.), situés dans un certain environnement et interagissant selon certaines relations. On utilise les systèmes multi-agents pour simuler des interactions existant entre agents autonomes. On cherche à déterminer l'évolution de ce système afin de prévoir l'organisation qui en résulte.

Ainsi dans le cadre de notre cours de système multi-agent et intelligence artificielle nous avons comme tâche de concevoir, d'implémenter et de simuler la circulation dans un carrefour de la ville d'Hanoi au Vietnam. Ce travail pratique a été simulé sur la plateforme GAMA 1.6.1.

Le présent rapport consistera à présenter les différents processus de simulation de la circulation. Pour ce faire, notre travail est constitué de tois grandes parties. La première partie constituera la phase de conception du modèle 1, la deuxième celui du modèle 2 et nous terminerons la conception du modèle 3.

2 Travail à réaliser.

2.1 Problématique

La capitale du Vietnam pays de l'asie pacifique, Hanoi, comme de nombreux pays doit faire face à de nombreux embouteillages dans les différents carrefours. De ce fait il est difficile de prédire les embouteillages dans un carrefour. Mais comment mêttre en place un cas de traffic dans un carrefour afin d'observer le comportement et essayer de réduire les embouteillages.

2.2 Objectifs

L'objectif de notre travail pratique est de simuler le traffic routier dans un carrefour à feu tricolore dans la ville d'Hanoi. Le travail sera effectué sur la plateforme de simulation GAMA la version 1.6.1. De ce fait il sagit dabord de choisir un carrefour sur la carte, de télécharger le fichier.osm , ensuite de traiter le fichier obtenu dans le logiciel qgis enfin inclure les fichiers.shp obtenu dans GAMA afin de prococéder à l'implémentation puis à la simulation. Dans cette étape de simulation il sagit de présenter aussi le comportement des agents quand le feu tricolore est rouge, vert ou orange.

3 Conception du modèle 1

3.1 Choix d'un carrefour à Hanoi.

Pour le choix de notre carrefour, nous accédons au site openstreemap.org. Une fois connecter nous sélectionnons la ville Hanoi puis nous avons choisi notre carrefour. La figure ci-dessous présente notre carrefour sélectionné qui est encadré en rouge.

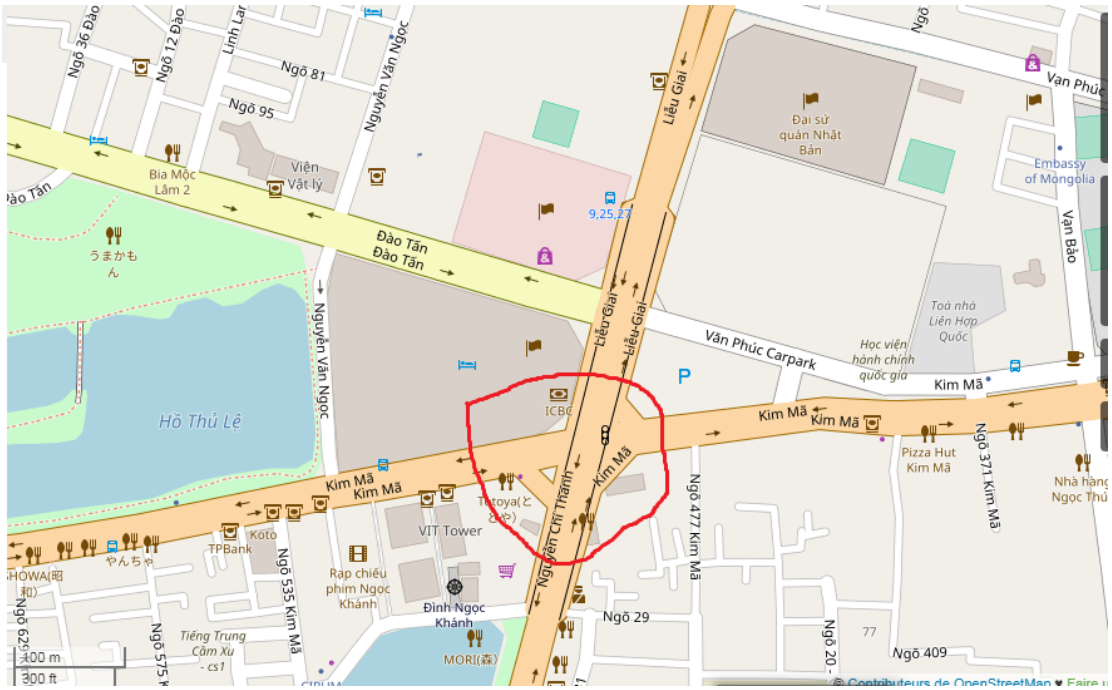

FIGURE *1 – Zone d'étude sur la plateforme openstreemap.*

Nous sélectionnons notre correfour et nous importons le fichier sous extention OSM. Ce fichier doit être traité sous QGIS.

3.2 Traitement du fichier OSM sous QGIS.

Lorsque le fichier est importé, il est sous formant OSM. Il doit être traité sous QGIS pour obtenir les fichiers shp. Pour ce faire nous importons notre fichier.osm dans qgis pour le traitement. La figure ci-dessous présente le carrefour sous qgis.

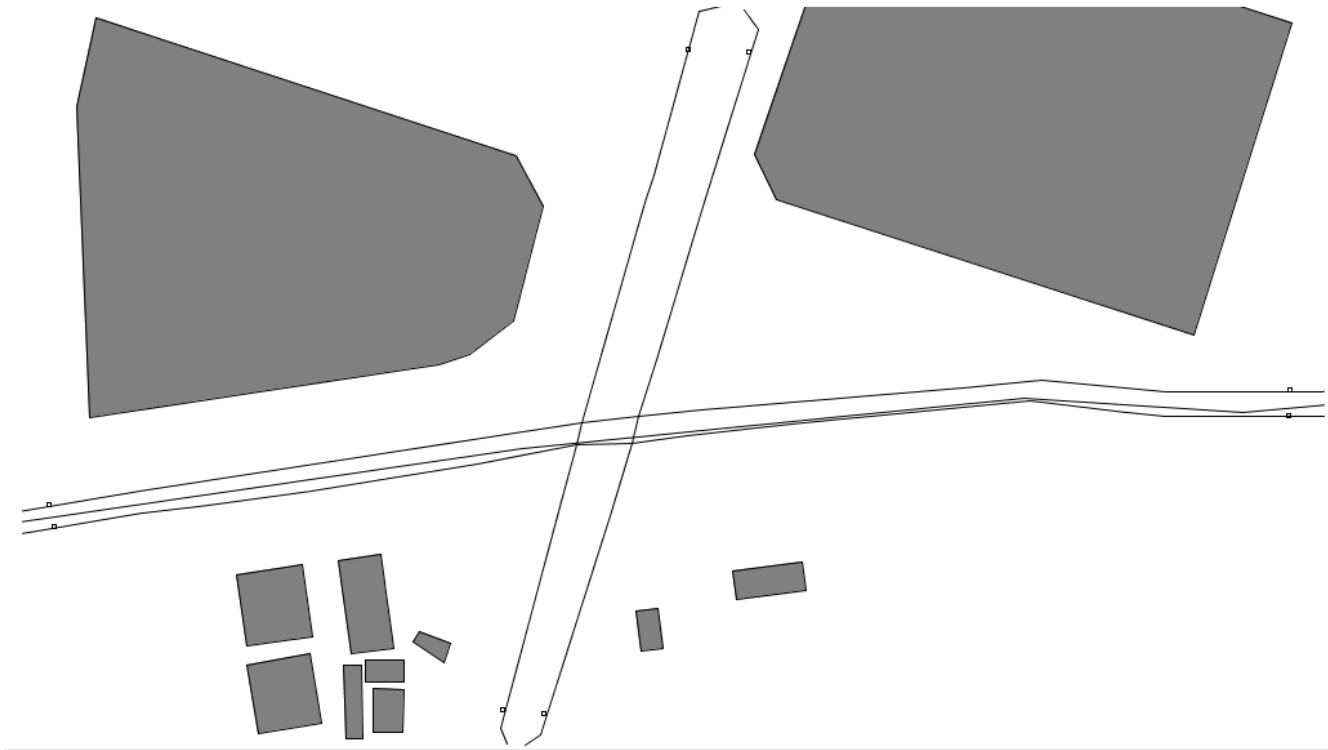

FIGURE *2 – Carrefour sur qgis.*

Après l'importation du fichier OSM. Nous traitons le fichier afin de séparer nos buildings les routes les intersections dans des fichiers shapes différents pour faciliter le travail sur la plateforme gama. De plus nous délimitons notre environement de travail avec un shape file environnement et nous créeons des shapefile feux tricolores. A la fin du traitement nous obtenons des fichiers .shp qui seront inclus dans notre dossier du TP. Nous créons notre projet sur gama puis le modèle. Nous commencons l'implémentation par l'inclusion de nos fichiers shapefile puis à l'implémentation des différents species et méthodes.

3.3 Les species et les méthodes

Les agents que nous présentons dans notre modèle sont ceux que nous avons creé sur gama et par la suite nous les avons affiché. Ces agents sont les suivantes :

- **Les routes.** importé depuis le fichier shapefile que nous nommons routes. Ils sont traçés en noir ;
- **Les immeubles** importé aussi depuis notre fichier shapefile de buildings. Ils sont représentés par des polygone en vert ;
- **Les voitures** ils sont représentés par une image de voiture ;
- **apparution_voitures** importé depuis un fichier shapefile c'est le point de génération, le point de départ de nos agents voitures ;

- **destination_des_voitures** importé depuis un fichier shapefile c'est la destination de nos agents voitures ;

Species du modèle 1		
Agents	Attributs	Méthodes
routes	**- color :** définit la couleur(noir) de nos routes **- nbLane** : définit les routes comme des lignes	**basic** : est une méthode qui permet de tracer nos routes
voiture les routes	**- speed :** la vitesse de la voiture **- target(point) :** destination de la voiture **image :** de type file contient l'image de la voiture	**Deplacement** : permet le déplacement des voitures sur **basic :** représentation de la voiture
immeubles	**color :** définit la couleur	**basic** : est une méthode de nos buildings qui permet la représentation des immeubles
apparution_voitures		**basic** : est une méthode qui permet de tracer les points de départ des voitures
destination_voitures		**basic** : est une méthode qui permet de tracer les points de destination des voitures

3.4 Simulation du modèle 1

Après avoir fini d'écrire notre modèle nous simulons pour voir le résultat comme le montre la figure ci-dessous.

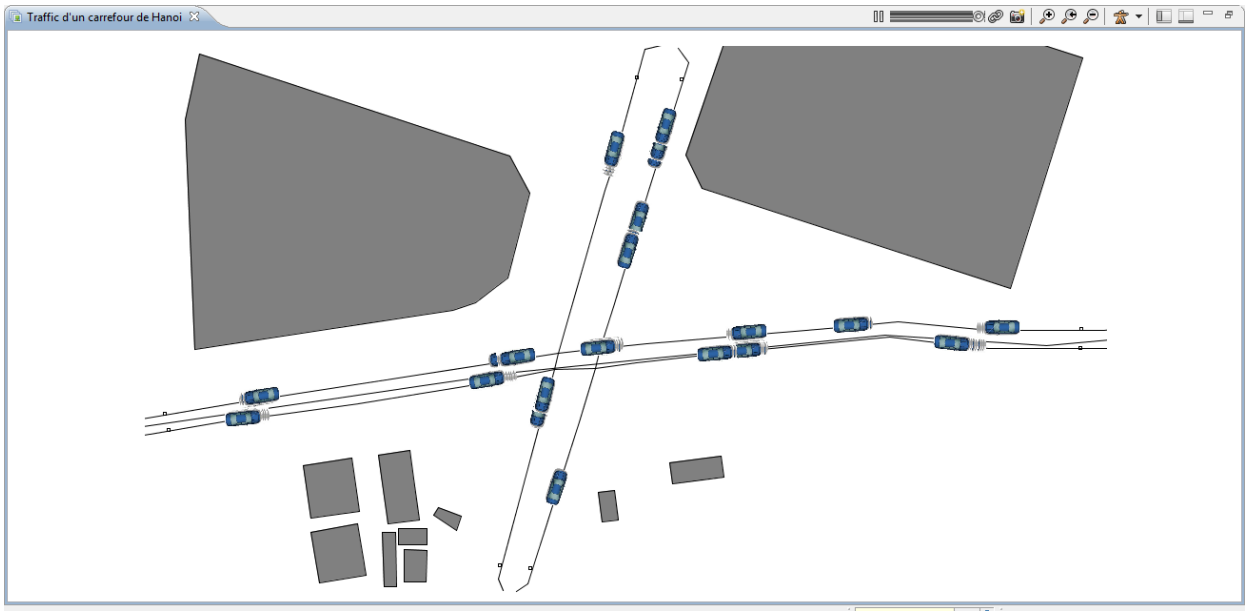

FIGURE *3 – Résultat de la simulation du modèle 1.*

Les paramètres du modèles sont modifiables. En effet ces paramètres sont les suivantes :

Paramètres	**Attributs**	**Catégorie**
Intensité des voitures	intensité	voiture
Taille des voitures	taille_voiture	voiture
Vitesse actuelle de la voiture	vitesse_actuelle	voiture
Vitesse maximale de la voiture	vitesse_max_voiture	voiture

TABLE *2 – Paramètres du modèle 1*

Le point de départ de nos voitures sont les débuts des routes. Une fois la simulation lançée les voitures sont générées automatiquement en fonction de l'intensité et commencent à se déplacer sur les routes. Leur point de destination est le specie "destination_des_voitures". Les paramètres présentés ci-dessus sont modifiables. Ainsi nous pouvons augmenter l'intentsité qui permet d'augmenter le temps de génération des voitures ou la taille des voitures ou la vitesse de nos voitures.

4 Conception du modèle 2 : ajout de feux tricolores

Nous avions copier le modèle1 et nous l'avions nommer modèle2. A ce modèle nous allons ajouter des feux tricolores qui fonctionnneront en fonction d'un temps prédéfini. En effet nous inclusons les fichiers shapefile feux1, feux2, feux3 et feux4 dans notre projet. De plus nous avons ajouté des species et des paramètres afin de gérer les agents quand le feu est rouge.
Le feux1 et feux3 tricolore fonctionnent simultanément ainsi de meême que pour feux2 et feux4

4.1 Les species et les méthodes ajoutées.

Les species que nous avons ajouté sont :

- **feux1 :** cette entité permet de représenter le premier feu tricolore. Sa couleur est initialisée à rouge.
- **feux2 :** cette entité permet de représenter le deuxième feu tricolore. Sa couleur est initialisée à vert.
- **feux3 :** cette entité permet de représenter le toisième feu tricolore. Sa couleur est initialisée à rouge.
- **feux4 :** cette entité permet de représenter le quatrième feu tricolore. Sa couleur est initialisée à vert.

Le tableau ci -dessous montre le bilan récapitulatif des species et des méthodes ajoutées :

Species du modèle 2		
Agents	Attributs	Méthodes
feux1	**color :** de type rgb permet de donner la couleur du feu N1	**- feux1_rouge :** est une action qui permet de changer le feu N1 au rouge **- feux1_jaune :** est une action qui permet de changer le feu N1 au jaune. **- feux1_vert :** est une action qui permet de changer le feu N1 au vert. **- changement_etat_feu :** est une méthode type state qui permet de changer l'état de feu. en fonction du temps à travers les actions définies. Pour cela elle contient des conditions.
feux2	**color :** de type rgb permet de donner la couleur du feu N2	**- feux2_rouge :** est une action qui permet de changer le feu N2 au rouge **- feux2_jaune :** est une action qui permet de changer le feu N2 au jaune. **- feux2_vert :** est une action qui permet de changer le feu N2 au vert. **- changement_etat_feu :** est une méthode type state qui permet de changer l'état de feu en fonction du temps à travers les

... suite page suivante...

... suite de la page précédente...

Agents	Attributs	Méthodes
		actions définies. Pour cela elle contient des conditions.
feux3	**color :** de type rgb permet de donner la couleur du feu N3	**- feux3_rouge :** est une action qui permet de changer le feu N3 au rouge **- feux3_jaune :** est une action qui permet de changer le feu N3 au jaune. **- feux3_vert :** est une action qui permet de changer le feu N3 au vert. **- changement_etat_feu :** est une méthode type state qui permet de changer l'état de feu. en fonction du temps à travers les actions définies. Pour cela elle contient des conditions.
feux4	**color :** de type rgb permet de donner la couleur du feu N4	**- feux4_rouge :** est une action qui de changer le feu N4 au rouge **- feux4_jaune :** est une action qui permet de changer le feu N4 au jaune. **- feux4_vert :** est une action qui permet de changer le feu N4 au vert. **- changement_etat_feu :** est une méthode type state qui permet de changer l'état de feu. en fonction du temps à travers les actions définies. Pour cela elle contient des conditions.
voiture **voiture**	**- feu1_est_rouge** est un booleen qui permet de vérifier l'état du feu 1 **- feu2_est_rouge** est un booleen	**- feu_rouge_imobillisation_voiture :** est une méthode qui permet de stopper les voitures en passant leur vitesse à 0 lorsque le feu est rouge ou jaune. **- depart_feu_vert :** est une méthode

... suite page suivante...

... suite de la page précédente...

Agents	Attributs	Méthodes
	qui permet de vérifier l'état du feu 2 **- feu3_est_rouge** est un booleen qui permet de vérifier l'état du feu 3. **- feu4_est_rouge** est un booleen. qui permet de vérifier l'état du feu 4.	qui permet aux voitures stoppés au feu rouge de redémarrer lorsque le feu passe au vert.

Au modèle nous avons ajouté des paramètres qui sont dans le tableau suivant :

Paramètres	**Attributs**	**Catégorie**
Durée du feu rouge	intensité	feux1
Durée du feu jaune	taille_voiture	feux1
Durée du feu vert la voiture	vitesse_actuelle	feux1

TABLE *4 – Paramètres(ajoutés) du modèle 2*

Fonctionnement : Ce modèle permet de gérer le comportement des voitures face aux différents feux tricolores. La méthode init initialise nos voitures et nos feux tricolores. La méthode changement_etat_feu des species(feux1,feux2, feux3 et feux4) permet le changement de nos feux, elle s'occupe de la gestion des feux tricolores.En effet lorsque les voitures arrivent au feu tricolore s'il est de couleur rouge ou jaune, la méthode **feu_rouge_imobillisation_voiture** du specie **voiture** permet de modifier la vitesse des voitures à 0, elles s'imobilisent. Si le feu qui était au rouge passe au vert la méthode **depart_feu_vert** du specie **voiture** permet de redémarrer les voitures qui étaient stoppées au feu rouge.
En somme dans ce modèle, les agents voitures s'arrêtent au feu lorsque celui-ci est rouge ou jaune et passent lorsqu'il est vert. De plus la durée des feux sont modifiables à travers les paramètres.

4.2 Simulation du modèle2

Après avoir fini l'impémentation de notre modèle sur gama, nous lancons la simulation (Voir image ci-dessous)

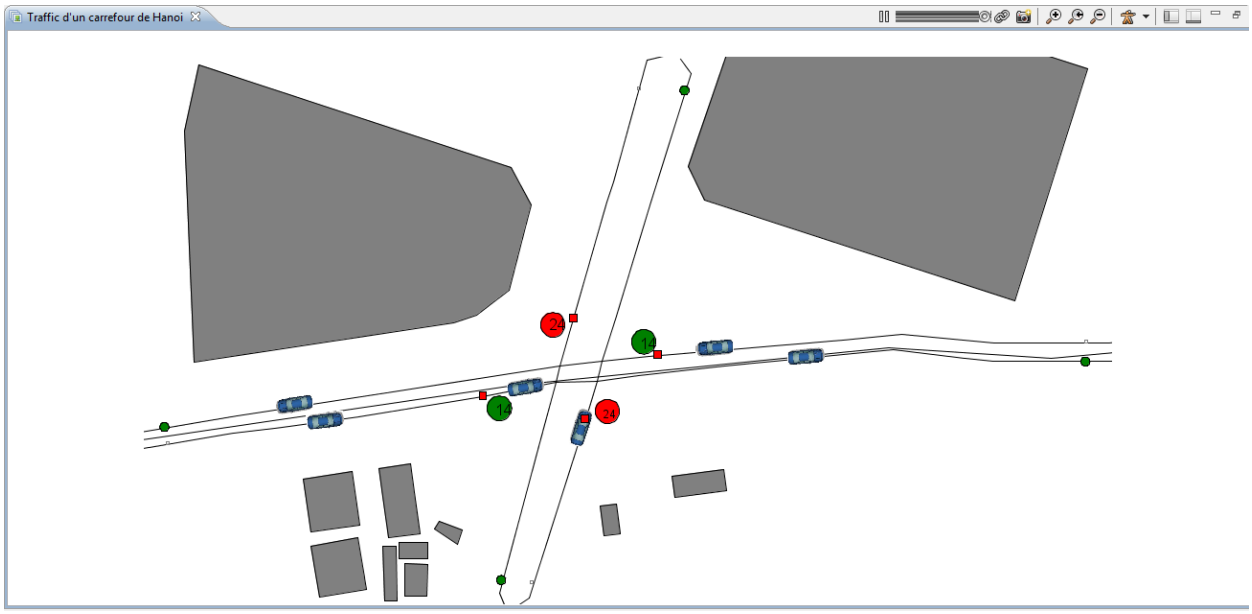

FIGURE *4 – Résultat de la simulation du modèle 2.*

Analyse : Nous constatons que le modèle 2 fonctionne parfaitement. En effet, lorsque le feu est rouge ou jaune les agents s'arrêtent et circulent lorsqu'il est vert donc le but du modèle a été atteint.

5 Conception du modèle3 : Nombre d'agents passant dans le carrefour.

5.1 Conception du modèle 3

Dans ce modèle nous avons copié le modèle 2 et nous avons renommé modèle3. Nous ajoutons ainsi les fonctionsqui permettentde compter dans le nombre d'agents qui passe au différent direction lorsque le feu est vert. Pour cela nous avons initialisé des variables globales booléens (feu_est_rouge) pour chaque feu. De ce fait si le feu est rouge le booléen passe à true, nous ne comptons pas les voitures et si le feu est vert nous dénombrons les voitures qui y passent dans chaque carrefour. Les différentes instructions ont été ajoutées dans le output(Voir figure ci-dessous).

```
display nombre_agent{
    chart "nombre_agent"
    type: 'series'{
        data "Nombre d'agents passé au premier carrefour"
            value: list(voiture) count(!each.feu1_est_rouge)
            color:rgb('red');
        data "Nombre d'agents passé au deuxième carrefour"
            value:list(voiture) count(!each.feu2_est_rouge)
            color:rgb('yellow');
        data "Nombre d'agents passé au troisième carrefour"
            value:list(voiture) count(!each.feu3_est_rouge)
            color:rgb('green');
        data "Nombre d'agents passé au quatrième carrefour"
            value:list(voiture) count(!each.feu4_est_rouge)
            color:rgb('black');
    }
}
```

5.2 Simulation du modèle3

Après l'impémentations du modèle 3, nous simulons et nous obtenons sous forme de graphe le nombre de voitures qui passent dans chaque carrefour(voir figure ci-dessous).

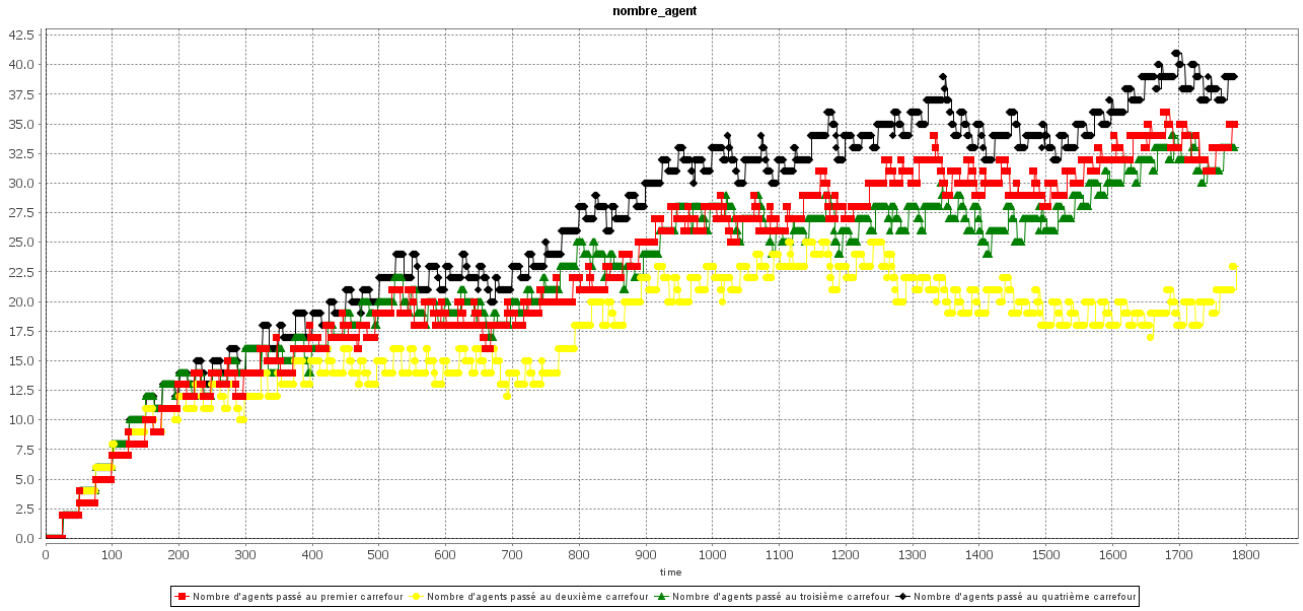

FIGURE *5 – Résultat de simulation du modèle 3.*

La courbe rouge montre le nombre de voiture passé au premier feu, la courbe jaune le nombre de voiture passé au deuxième feu, la courbe verte au troisième feu et la courbe noire au quatrième feu. De ce fait le but du modèle 3 est atteint.

6 Conclusion

Au cours de ce travail nous avions procédé à une conception et une implémentation du traffic de la circulation dans un carrefour d'Hanoi. Dans le TP nous avons choisi un carrefour sur openstreemap et nous avons traité sur QGIS , ensuite nous avons importélles fichiers shapefile obtenus dans le logiciel de simulation GAMA 1.6. Ensuite Nou savons crée les agents voitures et le comportement des différents agents face aux feux tricolores. Ainsi quand le feu est rouge les agents s'arrêtent et passent au vert. Nous avons représenter sur un diagramme le nombre de voitures qui passent dans les différents carrefours. La réalisation du TP était complexe pour des novices comme nous mais nous avons su cerner le foctionnement du logiciel de simulation GAMA et du logiciel QGIS. Ce programme n'est pas fini mais pourait etre utilisé dans le cadre des perspectives pour une amélioration. Ainsi il pourait etre profitable pour la prise de décision lors des embouteillages.

Références

[1] http ://gama-platform.org/tutorialsRoadTrafficModel$_step1/$
[2]$https://www.youtube.com/watch?v=-ApmqwJQVME$
[3]$http://sigea.educagri.fr/tutoriels-de-logiciels-sig/qgis/$