République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et
De la Recherche Scientifique
Université d'Ibn Khaldoun – Tiaret
Faculté des Mathématiques et d'Informatique
Département Informatique

Mémoire de fin d'études
Pour l'obtention du diplôme de Master en Informatique

Option : Génie logiciels

Thème

Modélisation d'une Circulation Routière

avec une Approche Basée UML/SMA.

Rédigé par : Goucham Nor El Houda

Dirigé par : N.Hattab

Année universitaire 2015-2016

Résumé

De nos jours, de nombreux travaux de recherche à travers le monde portent sur des problèmes de plus en plus complexes. Parmi ces problèmes, nous nous intéressons dans ce mémoire aux problèmes de la circulation routière. La motivation vient du fait que ce secteur nécessite de nouvelles modèles, méthodes ou applications permettent de contrecarrer l'augmentation des problèmes issus de la circulation routière.

Durant ces dernières années la modélisation à base d'agent est en réelle ébullition, cette ébullition est due au nombre très important de chercheur qui utilisent ce concept.
L'idée de base de cette technique est résumée par le fameux slogan '**tout composant est un agent'**, derrière ce slogan, n'importe quel phénomène est modélisé par un ensemble d'entités (agents) qui le composent et qui interagissent entre eux.

L'objectif de notre travail consiste à modéliser une circulation routière basique avec un langage de modélisation basé sur les agents, en tenant compte des contraintes environnementales du la tache étudié. Nous avons adopté pour ça une approche UML/Netlogo afin de décrire les principales fonctionnalités de notre circulation routière.

Finalement, nous avons implémenté notre approche pour la valider expérimentalement.

Mots-clés : Circulation Routière, Modélisation, Simulation, SMA, UML, Net Logo.

Sommaire

Table des figures

Liste des tableaux

Liste des abréviations

IAV : Intelligent Autonomous Vehicles

PRODYN : Programmation Dynamique

SCATS : Sydney Co-ordinated Adaptative Trafic System

SCOOT : Split Cycle and Offset Optimisation Technique

SMA :Simulation Multi Agent

TRANSYT : TRAFIC Network STUDY TOOL

UML : Unified Modelling Language

Introduction Générale

De nos jours, de nombreux travaux de recherche à travers le monde portent sur des problèmes de plus en plus complexes. Parmi ces problèmes, nous nous intéressons dans cette thèse aux problèmes le transport routier. La motivation vient du fait que ce secteur nécessite de nouvelles méthodes ou applications permettent de contrecarrer l'augmentation des problèmes issus du trafic routier .Partout dans le monde, il a reçu plusieurs milliers de personnes meurent ou deviennent gravement blessé chaque année à la suite d'accidents de la circulation (tableau 1 : statistique d'Algérie 2015)[1]. En effet, l'un des facteurs favorables à cette problématique est dû à l'accroissement du nombre exponentiel de véhicules sur la route.

Désignation	Année 2015
Nombre d'accidents	16245
Nombre de Blessés	19337
Nombre de Décès	809

Tableau 1:Etat des accédants de la circulation routière enregistre durant année 2015 en zone urbaines et Les principale causent des accédants de l'année 2015.

Ces travaux vue de réduire le nombre d'accidents, de diminuer le trafic. Toutefois, la plupart de ces travaux ont presque convergé vers des solutions basées sur l'intelligence artificielle. La simulation est aujourd'hui contribuent à la compréhension des événements dus aux interactions entre les usagers tels que les embouteillages ou les accidents.[**CHAMPION et Al]** décrivent la simulation comme un outil efficace utilisé pour la reproduction et l'analyse d'un large éventail de problèmes complexes, difficiles à étudier par d'autres moyens qui pourraient être trop coûteux ou dangereux. « L'amélioration des infrastructures est très coûte et chaque modification doivent être évaluée avec soin pour son impact sur l'écoulement du trafic. Simulations de trafic informatique forment une méthode rentable pour faire ces évaluations » [**P.EHLERT, et L.ROTHKRANTZ**]

[1] http://www.dgsn.dz

La simulation informatique basée sur l'approche multi-agents considérés a la modélisation du comportement individuel et des interactions. L'approche multi-agents est un moyen prometteur pour déployer des modèles et des applications qui, souvent doivent traiter des problèmes complexes et inter opérer avec d'autres systèmes hétérogènes, en plus est utilisés lorsque le système étudié est de grande taille et repose sur l'interaction de différentes entités.

Problématique

La réalisation d'un modèle ou d'un prototype pour un phénomène quelconque, recommande un procédé scientifique, qui repose principalement sur une démarche avec trois aspects incontournable:

– le phénomène réel (ou virtuel) que l'on souhaite étudier.

– le modèle de ce phénomène.

– la simulation de ce modèle par ordinateur (simulateur).

Dans ce mémoire, nous nous basons sur cette démarche, particulièrement celle liée à l'élaboration des modèles et à l'implémentation de simulateur permettant de les exécuter. De ce fait, et dans cette esprit, nous désirons découvrir par l'expression ‘‘passage UML/ Plateforme SMA’’, la réponse à ‘‘certaines interrogations’’ qui sont continuellement soulevées, mais d'une manière différente, par exemple, à travers un ensemble de méthodologies et plateformes multi agent différentes.

Par l'expression "passage UML/ Plateforme SMA", nous percevons la façon avec laquelle nous approcherons le problème de gestion d'intersections que nous traitons dans ce mémoire à travers une tâche générique prédestiné.

Le problème axial de ce mémoire et le problème de gestion d'intersections, qui est en général vu comme un problème d'optimisation.

Par l'expression "Certaines Interrogation" nous voulons dire que nous n'avons pas l'intention de répondre à toutes les questions de circulation et d'intersection, qui peuvent surgir dans le cadre Trafique routière, parce que les interrogations dans ce champ sont vastes, et intéresse beaucoup d'aspects qui posent beaucoup d'interrogations. Nous nous sommes intéressés dans ce travail à deux problèmes succincts :

-La maitrise de la régularisation des feux rouge dans les carrefours d'une façon autonomique et non pas automatique, et

-La maitrise de la versatilité de passages piétons, aussi bien par la régularisation autonomique de ces moyens de versatilité pour les citoyens.

A notre avis la problématique de circulation des agents mobiles (véhicules) est essentiellement une problématique de régularisation de feux de signalisation et de passage des piétons. Ces deux problèmes peuvent être exprimés en deux questions (Q1 et Q2) : (cahier de charge pour notre modélisation).

Q1: Comment les agents non mobiles (feux rouges virtuelles) peuvent-ils profité de la notion de perception intrinsèque aux agents ? Afin de régularisée les alternances de ces couleurs (rouge et vert dans notre modèle), une régularisation adoptée principalement pour des raisons énergétiques, peaufinée par rapport aux existences réels de véhicules dans chaque voie (en temps réel) et non pas

Par rapport au compteur numérique accordé à chacun de ces feux de circulations, dans le but final de minimiser le nombre total des véhicules à l'arrêt à chaque intersection.

Q2: Comment les agents (passage piétons) peuvent-ils crées un régulateur autonomique adaptable en même temps aux véhicule et aux piétons, afin d'administrer l'interdiction/ l'autorisation de mouvement/mobilité de piétons/véhicule.

L'interrogation ici stipule qu'un agent (passage piétons) ne doit pas interrompre les véhicules alors qu'il n y pas de citoyens dans le trottoir à un instants donné, et vise versa. Sommairement il faut que nos passages piétons soient adaptables à la situation.

Ainsi, dans ce mémoire, nous adoptons un ensemble d'hypothèses, énoncées dans le dernier chapitre de ce mémoire.

Organisation du mémoire

Ce mémoire comporte 2 chapitres incluant une introduction générale et conclusion.

Le premier chapitre nous présentons une étude bibliographique sur les systèmes de gestion d'intersections dans les réseaux de transport. Nous mettons en avant les grandes dimensions selon lesquelles ces systèmes se distinguent en insistant sur les différents types d'algorithmes utilisés.

Le deuxième chapitre nous présentons l'approche multi-agents et le langage/plateforme (multi-agents) Netlogo, et une nouvelle approche UML/SMA.

Enfin, Le troisième chapitre nous proposons une modélisation de notre problème, et présentation de notre simulateur pragmatique nommé **"SimCirc"** basé sur l'approche UML/SMA.

Chapitre1 : Etat de l'art

Introduction

Ce chapitre est consacré à l'étude bibliographique des approches existantes pour les problèmes de gestion du trafic sous l'intersection dans un réseau de transport. Nous notons que ce problème n'a pas été beaucoup étudié dans la littérature.

1.1. Gestion d'intersections dans un réseau de transport :

Le travail, [**Bazzan 2009**] présente un aperçu des méthodes et des approches dans les domaines de l'ingénierie du trafic pour surmonter les problèmes de congestion aux intersections, **Bazzan** choisit une classification historique, à savoir trois grandes classes :

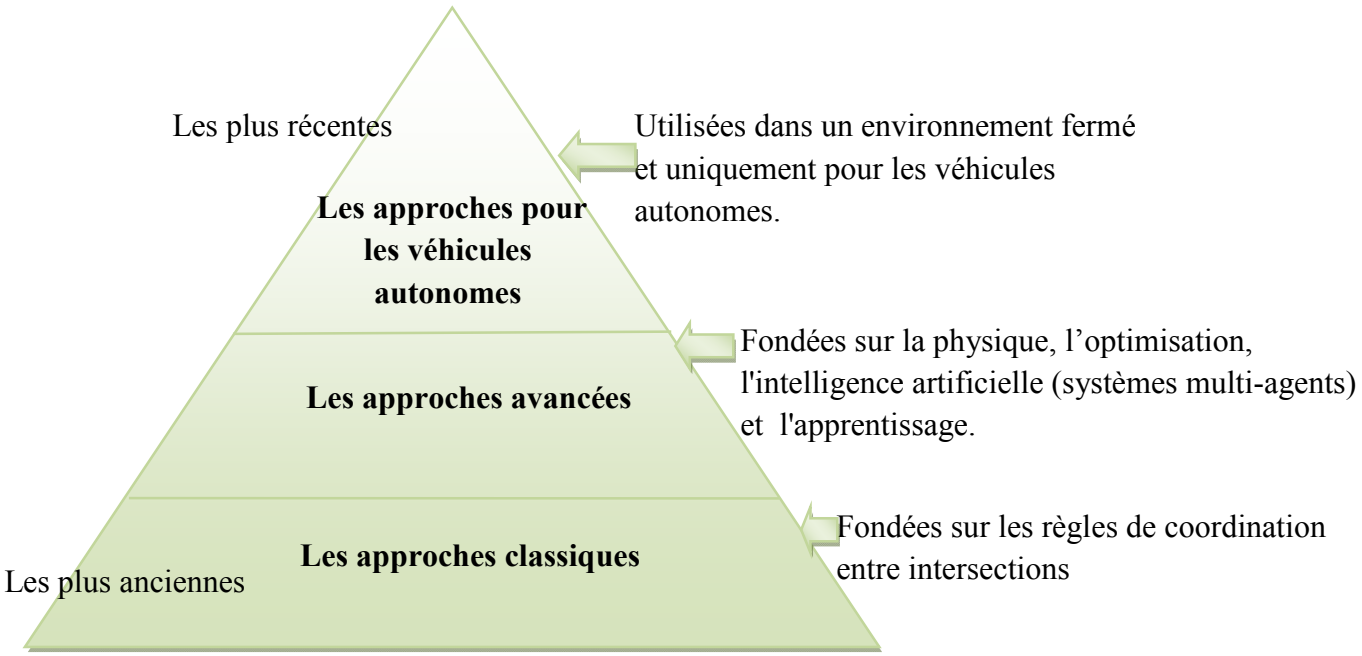

Figure 1:les trois grandes classes présentées par Bazzan.

Cependant, nous avons choisi de présenter les travaux différemment. Il est plus simple et clair de mettre plutôt en avant les grandes dimensions selon lesquelles ces travaux se distinguent puisqu'il existe plusieurs classifications possibles. Nous proposons, ici, de présenter ces grandes dimensions en insistant sur les différents types d'algorithmes utilisés.

Dans cette partie, nous dressons un aperçu des méthodes utilisées par le code de la route pour gérer les intersections et garantir la sécurité des usagers de la route. Ensuite, nous présentons le problème de gestion d'intersections comme un problème de satisfaction des usagers et d'optimisation. Après, nous détaillons quatre approches de référence qui correspondent aux systèmes les plus déployés actuellement. Enfin nous avons présenté les approches spécialement conçues pour les véhicules intelligents autonomes (IAV).

1.1.1. Le code de la route :

Dans cette partie, nous considérons par défaut le code de la route français. Les systèmes de gestion du trafic (feux de signalisation) ne sont pas installés partout, car les autorités ne peuvent pas gérer toutes les intersections par des systèmes coûteux. Ainsi, il existe des intersections gérer par de simples panneaux et parfois des intersections dépourvues de toute signalisation. C'est pourquoi le législateur a défini des solutions simples qui doivent être respectées par les conducteurs dans ces situations.

1.1.1.1. Règles de priorité :

Par exemple le code de la route français, dans son article droit.org [2] r415-5: « Lorsque deux conducteurs abordent une intersection par des routes différentes, le conducteur venant par la gauche est tenu de céder le passage à l'autre conducteur, sauf dispositions différentes prévues ». Ceci implique une règle simple à respecter à chaque intersection sans signalisation souvent appelée « priorité à droite ».

Nous trouvons aussi dans l'article R:415- 10 : « Tout conducteur abordant un carrefour à sens giratoire est tenu, quel que soit le classement de la route qu'il s'apprête à quitter, de céder le passage aux usagers circulant sur la chaussée qui ceinture le carrefour à sens giratoire ». Cet article annule donc l'effet de la première règle de priorité à droite et définit une nouvelle règle de priorité à gauche appliquée uniquement pour les carrefours à sens giratoire souvent appelés « ronds-points ».

Ces règles ne sont pas communes à tous les pays. Elles différent d'un pays à un autre et surtout dans les pays où les véhicules circulent à gauche sur la chaussée. Par exemple, au Royaume-Uni, la priorité aux ronds-points est pour les véhicules qui viennent de droite et il n'existe pas de règle générale pour les intersections, la priorité étant généralement indiquée.

[2] http://codes.droit.org/cod/route.pdf

1.1.1.2. Les panneaux de signalisation :

Il existe des panneaux pour la gestion des intersections. Par exemple, parmi les panneaux les plus répandus nous trouvons les panneaux de signalisation de céder le passage. En application de l'article R:415- 7 du droit.org: « certaines intersections indiquées par une signalisation dite "cédez le passage", tout conducteur doit céder le passage aux véhicules circulant sur l'autre ou les autres routes et ne s'y engager qu'après s'être assuré qu'il peut le faire sans danger ». Il est souvent utilisé pour définir une route principale par rapport à une route secondaire et n'oblige pas les usagers de la route secondaire à l'arrêt.

Il existe aussi un autre panneau très répandu dans presque tous les pays du monde : le panneau « STOP ». Si un conducteur trouve ce panneau implanté à une intersection sur son chemin, il doit obligatoirement marquer l'arrêt avant de continuer. Il indique que chaque véhicule sur cette voie n'est jamais prioritaire quelle que soit la direction qu'il souhaite prendre. Ainsi nous trouvons dans l'article R:415-6 du droit.org : « A certaines intersections indiquées par une signalisation dite stop, tout conducteur doit marquer un temps d'arrêt à la limite de la chaussée abordée. Il doit ensuite céder le passage aux véhicules circulant sur l'autre ou les autres routes et ne s'y engager qu'après s'être assuré qu'il peut le faire sans danger ». Ce panneau est généralement utilisé pour contrôler le trafic aux intersections dangereuses, où le trafic est dense, mais pas suffisamment encombrées pour installer un système de feux de signalisation.

Ces panneaux que nous venons de citer sont très utilisés dans les villes et sont connus pour leur fiabilité. Ils sont toujours présents et ne risquent pas de mal fonctionner. C'est pour cette raison qu'ils peuvent aussi être présents à côté des feux de signalisation afin d'assurer le trafic en cas de défaillance des systèmes électroniques.

1.1.1.3. Les feux de circulation tricolores :

Dans les systèmes de gestion du trafic, le système le plus connu et le plus utilisé dans le monde est le système des feux de signalisation. En consultant l'encyclopédie libre enligne Wikipedia, on peut trouver cette définition des feux de signalisation[3].

« Un feu de circulation routière est un dispositif permettant la régulation du trafic routier entre les usagers de la route, les véhicules et les piétons. Les feux destinés aux véhicules à

[3] https://fr.wikipedia.org/wiki/Feu_de_circulation

moteurs sont généralement de type tricolore, auxquels peuvent s'ajouter des flèches directionnelles. Ceux destinés aux piétons sont bicolores et se distinguent souvent par la reproduction d'une silhouette de piéton. Les feux tricolores pour cyclistes se distinguent par la reproduction d'une bicyclette. »

Généralement, un feu tricolore est composé d'un système électronique commandé. Il est composé de trois couleurs principales. La couleur rouge indique l'obligation d'arrêt aux véhicules. La couleur orange qui ne dure que quelques secondes signale le passage du rouge au vert. La couleur verte indique aux véhicules qu'ils ont la priorité exclusive pour passer. Ces couleurs ont été choisies parce qu'elles ont l'avantage d'être très distinctes.

Le système des feux de signalisation est le système le plus efficace pour la gestion du trafic car il évite tout malentendu entre les différents conducteurs au moment du passage. Cependant, c'est aussi le système qui génère le plus de retard puisqu'il favorise à chaque instant un ou deux flux (qui ne se croisent pas) et oblige systématiquement l'arrêt de tous les autres flux entrants.

1.1.1.3.1. Définition d'un flux, d'un cycle et d'une phase :

Une intersection, un flux de véhicules est l'ensemble des véhicules entrant par une voie donnée et ressortant par une autre. Le trafic dans une intersection est constitué d'un ensemble de flux de véhicules, chacun provenant d'une source. Deux flux sont cohérents s'ils peuvent évacuer l'intersection simultanément. Une intersection gérée par des feux de signalisation est composée de plusieurs feux tricolores, implantés sur les différentes voies entrantes dans l'intersection.

Une phase d'un feu est une période durant laquelle un ou plusieurs flux cohérents sont admis dans le carrefour. Le cycle d'un feu représente la durée qui sépare deux phases identiques de l'intersection. Il est défini par une séquence de phases. La figure 2 présente un cycle de feux composé de quatre phases.

Ainsi, la durée d'un cycle d'un feu sera d'autant plus élevée que le nombre de phases sera grand. Le « temps de dégagement » est un temps perdu qui est interposé entre deux phases consécutives. On l'appelle aussi le rouge intégral. Ce rouge est la durée nécessaire pour que tous les véhicules qui sont dans le carrefour puissent évacuer la zone de conflit avant l'admission d'une autre phase. On remarque donc que l'augmentation de la durée du cycle

permet d'augmenter la capacité du carrefour, ceci est lié à la minimisation de la part du temps de dégagement.

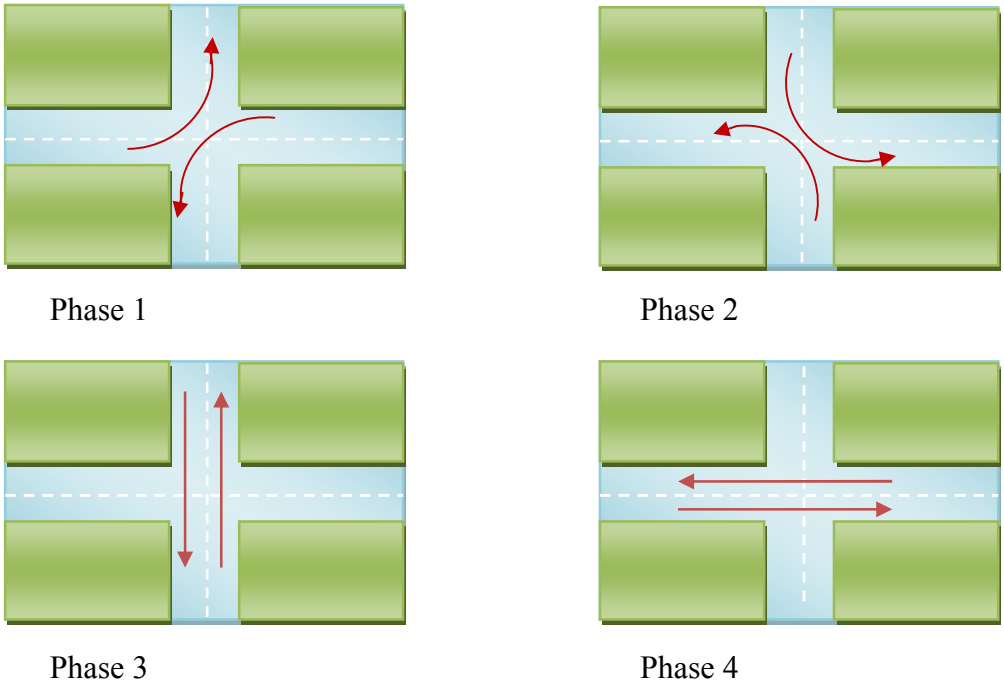

Figure 2:Plan d'un cycle composé de 4 phases.

Néanmoins, une durée d'évacuation trop grande favorise l'apparition de blocages au niveau des carrefours suivants. Un autre concept qui est le déphasage entre les carrefours adjacents est très important dans la modélisation et le contrôle du trafic géré par les feux de signalisation. Ce paramètre permet de contrôler la création de feux verts successifs pour une voie donnée appelés aussi ondes vertes. Il est calculé à partir de la distance entre deux feux consécutifs et d'une vitesse moyenne des véhicules [**Mammar, 2007**].

1.1.2. Quel est le problème ?

La demande croissante de mobilité dans nos sociétés a engendré de nouveaux problèmes. Plus précisément, à partir de la deuxième moitié du siècle dernier, le phénomène de congestion du trafic est apparu, plus connu sous le nom d'embouteillage. Ceci est dû au fait que de nombreux véhicules veulent utiliser les mêmes infrastructures routières au même moment. Réguler le trafic dans un réseau routier est au moins un problème de satisfaction puisqu'il faut satisfaire les besoins des usagers de la route, c'est-à-dire qu'ils puissent atteindre leur destination. Une façon triviale de résoudre le problème de la gestion d'intersections est d'arrêter tous les flux pour n'en laisser passer qu'un sous-ensemble, comme c'est le cas dans la

gestion actuelle des intersections dans les villes (phase d'un feu de signalisation). Or il a été montré que cette stratégie est coûteuse en termes de temps [**Fok et al. 2012**] puisqu'elle oblige l'arrêt systématique des flux. Donc, la régulation du trafic est aussi un problème d'optimisation puisqu'il s'agit de trouver les règles de fonctionnement du réseau (au sens large) qui induisent la meilleure qualité de trafic possible, par exemple le fluidifier et ainsi réduire les temps d'attentes de chaque véhicule. La régulation du trafic est réalisée en définissant les règles de comportement des véhicules (leurs conducteurs) et des systèmes de signalisation placés dans l'environnement (marquages au sol, panneaux stop, panneaux priorité, feux de circulation...). Le code de la route définit ainsi le comportement que doit suivre un conducteur en fonction des actions des autres véhicules et de la signalisation environnante. Ce code, s'il est respecté, assure la sécurité de chacun, mais pas nécessairement la satisfaction de chacun (arrivée à destination), ni l'optimisation du trafic. Pour améliorer le trafic sur un réseau, on peut jouer :

- sur le réseau, en ajoutant des routes et des intersections.

- sur la disposition de la signalisation

- sur le contrôle (les règles de fonctionnement) des signalisations actives/dynamiques (feux de circulation).

-sur le contrôle (le comportement) des véhicules.

Ici, nous supposons fixes les besoins, le réseau, et la disposition de la signalisation. Comme nous allons le voir, la plupart des travaux rencontrés jouent sur le contrôle des signalisations, mais quelques uns s'intéressent aussi au contrôle des véhicules.

1.1.3. Quatre approches de référence :

Les approches de référence correspondent aux systèmes les plus déployés, car ils sont les plus anciens, mais aussi parce qu'ils sont simples et ne nécessitent pas beaucoup d'infrastructure comparés à d'autres. A partir de 1960, plusieurs algorithmes ont été proposés pour améliorer le passage des flux de véhicules. Quatre systèmes différents, TRANSYT, SCOOT, SCATS et PRODYN, sont décrits pour montrer la diversité de ces approches classiques. Ils s'intéressent tous à la gestion du trafic dans les réseaux sans être centralisés (à l'exécution) pour autant. Nous nous sommes appuyés dans cette section sur le livre [**Mammar, 2007**] qui présente une synthèse de ces différents systèmes de gestion de trafic.

1.1.3.1. TRANSYT :

(TRANSYT) TRAFIC Network STUDY TOOL [**Robertson, 1969; Mammar, 2007**] est l'un des premiers systèmes proposés. Il repose sur une optimisation hors-ligne qui génère des plans de coordination optimaux entre les feux de signalisation d'un réseau pour une période donnée. TRANSYT exige beaucoup de paramètres d'entrées, comme par exemple la géométrie des artères des intersections, le débit des véhicules, le taux de véhicules sur chaque voie sortante de chaque intersection (fixé à l'avance), le temps de feu vert minimal, des plans de feux initiaux, et des valeurs initiales pour les durées des cycles et les déphasages. A partir de ces paramètres, un modèle mathématique (calculé à partir de la dispersion des trains de véhicules) est simulé macroscopiquement et ses performances sont évaluées. Enfin vient l'étape de l'optimisation à l'aide d'un algorithme d'optimisation Hill-Climbing, c'est-à-dire améliorant progressivement la solution en modifiant légèrement la durée des feux verts et des décalages entre intersections adjacentes. Les plans récupérés à la fin sont des plans calculés pour une situation statique dans le temps qui dépend des paramètres d'entrée.

Pour résumer, TRANSYT est un système centralisé et n'est pas adaptatif au trafic. Une application réussie de ce modèle nécessite un calibrage approprié des paramètres du modèle. Plus précisément, [**Guebert et Sparks 1990**] ont montré qu'un étalonnage précis des paramètres est essentiel pour le développement de plans de synchronisation efficaces.

1.1.3.2. SCOOT :

SCOOT [**Mammar, 2007**] est un système décentralisé et complètement adaptatif à la situation du trafic. Il optimise :

- les durées de feu vert pour chaque intersection indépendamment,

- les décalages entre intersections voisines, et

- les cycles des feux entre les zones d'intersection.

Il collecte les données (nombre de véhicules par intervalle de temps) à partir des détecteurs installés sur les routes, boucles magnétiques en amont du carrefour (voir Figure 3)

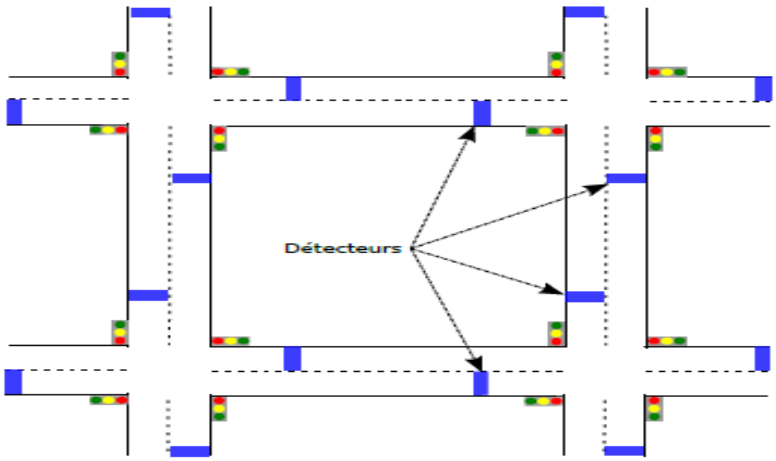

Figure 3:Boucles magnétiques en amont des carrefours (SCOOT).

Très similaire à TRANSYT, mais fonctionnant en temps réel, il compare différentes options (par exemple pour les durées des cycles : reproduire les mêmes plans, ajouter ou retrancher quelques secondes) et choisit celle qui le rapproche le plus de son objectif d'optimisation en utilisant l'algorithme Hill-Climbing. Il modifie graduellement les plans de feux de signalisation en cours d'exécution en fonction des données collectées par les capteurs installés sur les routes.

Ces modifications consistent en de petites variations de la durée des temps de cycle (0, 4 ou 8 secondes), des temps de feux vert (0 ou 4 secondes) et des décalages entre intersections (0 ou 4 secondes).

Ce système est installé dans beaucoup de villes dans le monde et principalement en Grande-Bretagne. Néanmoins, SCOOT reste un système faiblement adaptatif comparé à d'autres, vu ses faibles variations graduelles des phases à chaque cycle.

1.1.3.3. SCATS :

SCATS [**Lowrie, 1982; Mammar, 2007**] est un système partiellement décentralisé et adaptatif à la situation du trafic. SCATS utilise aussi les données des capteurs installés sur les routes.

Pour contrôler le trafic, SCATS s'appuie sur une notion de sous-systèmes qui regroupent les intersections (de 1 à 10 intersections par sous-système) et où une seule intersection est jugée critique (définie par l'utilisateur du système). Chaque sous-système s'optimise indépendamment des autres sous-systèmes. L'approche de SCATS consiste à utiliser des

bibliothèques prédéfinies de décalages et de durées de feu vert avec un algorithme temps réel de reconstruction de plan de feux pour intersection. Les bibliothèques (fournis par l'utilisateur du système) stockent généralement 10 ensembles de décalages et 4 ensembles de durées de feu vert. A partir de ces ensembles, l'algorithme reconstruit les plans des feux une fois par cycle sachant que la variation ne peut excéder les 6 secondes. L'algorithme compare plusieurs solutions et les données des capteurs avant d'appliquer la solution qui minimise la saturation des routes. Par ailleurs, dans SCATS il existe deux types de décalages : à l'intérieur d'un sous-système et entre deux sous-systèmes voisins. Les décalages sont sélectionnés aussi à partir des bibliothèques et en fonction des axes les plus saturés.

Comme SCOOT, SCATS est installé dans beaucoup de villes dans le monde et principalement en Australie (dans les villes contenant plus de 10 millions d'habitants). La différence avec SCOOT est que SCATS est un système hiérarchique qui dépend fortement du choix des bibliothèques des durées de feu vert et des décalages.

1.1.3.4. PRODYN :

PRODYN [**Henry et al. 1984; Mammar, 2007**] est un système décentralisé et adaptatif au trafic développé par le (CERT) Centre d'Etude et de Recherche de Toulouse en France.

L'objectif de PRODYN est d'optimiser la circulation en ligne en minimisant les retards aux intersections sur un horizon futur de 80 secondes. La fréquence de modification des phases et des durées des feux est fixée à 5 secondes (pas de discrétisation temporelle). Il utilise la programmation dynamique pour minimiser les retards en s'appuyant sur un modèle d'écoulement du trafic sur tout l'horizon. Ce dernier lui permet d'exprimer l'évolution des files d'attente en fonction des arrivées et des départs sur les tronçons. Pour les départs, comme SCATS et SCOOT, il utilise les données des capteurs installés sur les routes. Cependant, pour les arrivées, il lui faudrait idéalement les données des départs des intersections en amont, eux mêmes en cours d'optimisation, ce qui est impossible.

La solution choisie consiste à utiliser des prédictions établies au pas d'optimisation précédent tout en faisant l'hypothèse que la situation du trafic n'a pas beaucoup évolué. Etant donnée la complexité calculatoire exponentielle en fonction du nombre d'intersections, PRODYN optimise chaque intersection séparément en fonction des données reçues de ses voisines. Il est installé dans beaucoup de villes en France et en Belgique, mais il reste un système relativement complexe en termes de calcul et coûteux en communication.

1.1.4. Limites des approches de référence :

Plusieurs aspects peuvent être perçus comme des limitations de ces systèmes. Nous présentons ici ceux qui nous paraissent les plus importants.

Hypothèses simplicatrices :

Le premier problème est que ces différents systèmes (ces quatre plus d'autres) font des hypothèses simplicatrices (prédiction du trafic pour TRANSYT, variation des durées de feux selon des pas prédéfinis pour SCOOT, bibliothèques prédéfinies de décalages et de durées de feu vert pour SCATS, etc..) qui ne reflètent pas forcément la réalité.

Ces hypothèses sont essentielles pour ces systèmes, sinon les plans des feux de signalisation deviennent complexes à calculer et donc impossibles à exécuter en temps réel.

Approches autonomes pour véhicules non autonomes :

Un autre problème de ces différents systèmes est qu'ils ont été conçus pour les véhicules conduits par les humains. En effet, l'utilisation de véhicules classiques (non autonomes) pose surtout problème parce que, si les feux évoluent (optimisation en ligne), il faut « idéalement » prévenir les véhicules pour qu'ils puissent adapter leurs profils d'accélération au mieux. En plus, nous avons remarqué que les optimisations proposées favorisent le plus souvent les routes principales, généralement les plus congestionnées, et défavorisent les routes secondaires, sachant que dans les grandes villes ces dernières sont devenues de plus en plus importantes dû à la saturation des axes traditionnels.

Enfin, d'autres aspects sont délaissés par ces systèmes. Par exemple, le trafic peut être affecté par des incidents (accidents, blocages des routes, etc..). Il est donc nécessaire d'avoir des approches robustes et flexibles pour une bonne gestion du trafic.

1.1.5. Synthèse des approches :

Dans cette section, nous ne nous limitons pas à la synthèse des quatre approches précédentes, mais détaillons toutes les approches de gestion du trafic [**Bazzan, 2009**]. Nous décrivons, ici, les principales caractéristiques des approches rencontrées, puis nous donnons un tableau récapitulatif.

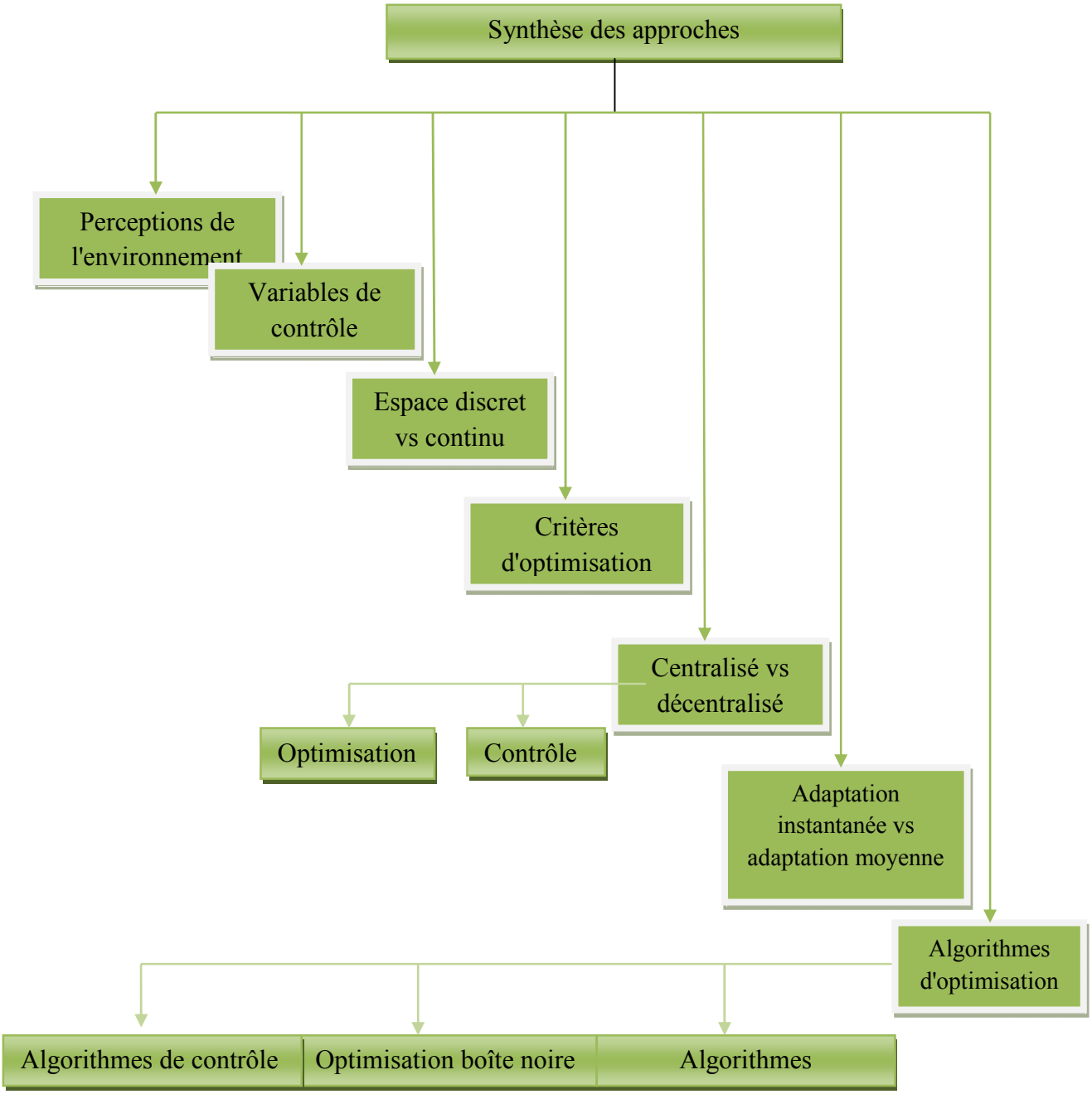

Figure 4: schéma d'organisation de la synthèse des approches.

1.1.5.1. Perceptions de l'environnement :

En fonctionnement, les feux de signalisation les plus simples ne perçoivent rien. Dans ce cas, ils ne peuvent pas s'adapter aux conditions de circulation réelles. Par exemple, nous pouvons citer le système TRANSYT où les auteurs ne peuvent que prérégler leurs cycles respectifs, et éventuellement leurs déphasages en fonction des conditions de trafic prédites. Ces réglages peuvent aussi dépendre du jour de la semaine ou de l'heure de la journée.

Dans d'autres approches plus évoluées, telles que les systèmes SCOOT et SCATS, nous pouvons trouver des feux de signalisation reliés à des capteurs (boucles magnétiques sous les voies, caméras, ...) permettant ainsi :

- d'estimer en continu le trafic, donc le débit, sur les différentes voies [**Rochner et al.2006; Bazzan, 2005**], ou

- de prédire les arrivées des véhicules à court terme, comme c'est le cas pour PRODYN. Il existe aussi des approches où les véhicules eux-mêmes préviennent les contrôleurs des intersections de leurs arrivées. Par exemple, [**Dresner et Stone ,2005**] utilisent un agent contrôleur à chaque intersection du réseau, et chaque véhicule en approche doit communiquer avec ce dernier an de réserver un créneau de passage.

Pour mieux faire encore, les phases d'une intersection peuvent communiquer entre elles pour se prévenir mutuellement de leurs activités, et donc des arrivées prévisibles de véhicules. Dans [**Kosonen, 2003**], l'auteur introduit une nouvelle méthode de contrôle des feux de circulation qui combine la simulation en temps réel, le contrôle multi-agents, et la logique floue. Pour chaque intersection, les phases sont modélisées comme un groupe d'agents. Chaque agent à la possibilité de modifier le cycle du groupe, si les autres agents sont d'accord, ce qui implique un besoin de négociation et d'échange de données locales de trafic.

Dans le même contexte de la communication entre intersections, [**Sánchez et Aguirre ,2007**] comparent deux protocoles fondés sur la négociation et les enchères entre les agents a un de définir les cycles des feux de circulation. En ce qui concerne les véhicules (quand ceux-ci sont contrôlés), ils peuvent éventuellement:

- percevoir, en plus des signalisations routières, les autres véhicules, et

-communiquer avec ceux-ci, comme c'est le cas du contrôleur proposé par [**Naumann et Rasche 1997**] fondé sur les négociations entre véhicules a un de déterminer un ordre de passage à travers l'intersection.

1.1.5.2. Variables de contrôle :

Dans la littérature de la gestion d'intersections, sont contrôlés : les feux de circulation ou les véhicules. En effet, la majorité des travaux considèrent les allumages des feux de signalisation comme variables de contrôle [**Brockfeld et al. 2001; Steingrover et al. 2005;France et Ghorbani, 2003**]. Plus précisément, ils varient l'ordre des phases, la durée des cycles et les déphasages entre intersections. Cependant, quelques travaux considèrent les véhicules eux-mêmes comme variables de contrôle, bien que leurs parcours ne peuvent pas être modifiés. Dans ces cas, les auteurs les font ralentir ou accélérer pour qu'ils passent au bon moment à un certain endroit [**Dresner et Stone, 2004; Naumann et al, 1997**].

1.1.5.3. Espace discret vs continu :

Certaines approches emploient des modèles de l'environnement discrets [**Brockfeld et al.2001; Gershenson, 2004; Wiering, 2000**]. Cela a toutefois le défaut de contraindre les déplacements des véhicules à des arrêts et des accélérations brusques et donc peu réalistes. Cependant, cette solution permet de faciliter la simulation.

Le reste des approches, qui sont plus réalistes à notre sens, soit emploient un modèle continu de l'environnement tels que les solutions de [**Kosonen ,2003**] soit sont indépendantes, c'est-à-dire que l'optimisation fait abstraction de la notion d'espace [**Rochner et al, 2006; Balan et Luke, 2006**].

1.1.5.4. Critères d'optimisation :

Le problème de gestion d'intersections est en général vu comme un problème d'optimisation. Dans les travaux rencontrés, ce n'est pas toujours explicite, mais souvent les auteurs partent du fait qu'ils sont confrontés à un problème dans lequel il faut maximiser ou minimiser une grandeur. Nous nous focalisons, dans cette section, sur les différents critères d'optimisation rencontrés. Nous pouvons distinguer deux critères principaux :

-La plupart des auteurs se sont concentrés sur la minimisation du retard moyen des véhicules. C'est-à-dire qu'ils considèrent le retard de tous les véhicules, à chaque instant, en comparant

leurs temps de parcours à un temps calculé théoriquement [**Mohring et al, 2006; Köhler et al. 2005; Bhouri et al, 2011; Hounsell et Shrestha, 2012**]. A notre connaissance, il n'existe pas de travaux considérant la minimisation d'un autre type de retard tel que le pire retard par exemple.

- D'autres travaux, tels que dans [**Ferreira et Khosla, 2000; Bazzan, 2005**], se sont intéressés à la minimisation des temps d'attentes des véhicules au niveau des intersections. Dans le même contexte, [**Da Silva et al. 2006**] considèrent la minimisation du nombre total des véhicules à l'arrêt à chaque intersection.

Cependant, il existe aussi d'autres critères dans la littérature, mais qui restent des cas particuliers dus à des contraintes de modélisation ou des contraintes algorithmiques. Par exemple nous pouvons citer les travaux de :

- [**Nunes et Oliveira 2004**] qui considèrent la minimisation du taux d'occupation sur les voies (nombre de véhicules par rapport à la taille du réseau), calculées d'une façon centralisée par l'environnement de simulation, ou

- [**Richter et al. 2007**] qui maximisent le débit en maximisant le nombre de véhicules qui entrent dans les intersections, car leur environnement de simulation est discret.

Nous remarquons que ces critères ne sont pas vraiment différents les uns des autres, c'est-à-dire qu'ils tendent tous à fluidifier la circulation des réseaux de transport en laissant quitter les véhicules le plus rapidement possible.

1.1.5.5. Centralisé vs décentralisé :

Les systèmes multi-agents ont fourni un moyen efficace pour décentraliser les approches. Dans cette partie, comme la plupart des auteurs, nous distinguons l'exécution (le contrôle) et l'optimisation.

1.1.5.5.1. Contrôle :

A notre connaissance, la plupart des approches impliquent des contrôleurs complètement décentralisés. En effet, un contrôleur centralisé serait trop complexe et non robuste aux problèmes de communications. Il est généralement suffisant pour un feu de signalisation de connaître des informations liées à son voisinage an d'agir efficacement. Cependant, dans les travaux rencontrés, il en existe que nous qualifions de « centralisés par zones » [**Robertson,**

1969; Mohring et al. 2006; Köhler et al. , 2005], c'est-à-dire que, pour un ensemble de feux de circulation, les auteurs affectent un unique contrôleur en charge de réguler cette zone.

1.1.5.5.2 Optimisation :

L'optimisation aussi se fait rarement de manière centralisée, parce que le problème à résoudre serait trop complexe (trop de paramètres à régler simultanément). Par contre, décentraliser/distribuer l'optimisation implique typiquement de définir un critère d'optimisation local à chaque intersection (mesurable en fonction de ce qui se passe dans le voisinage de l'intersection), ce qui pose les difficultés suivantes :

-Il n'est pas évident de pouvoir définir de tels critères locaux tels que l'optimisation de ceux-ci garantisse l'optimisation du critère global initial. De ce fait, plusieurs auteurs se contentent de l'optimisation du critère local [**Ferreira et Khosla, 2000; Da Silva et al. , 2006**] et n'ont pas de garanties « globales »

- Chaque agent (intersection ou véhicule) poursuit son propre objectif, donc nous sommes dans le cadre d'un problème de théorie des jeux générique, dans lequel les joueurs ne collaborent pas nécessairement. C'est pour cette raison que [**Camponogara et Kraus Jr [2003], Bazzan [2005] et Mandiau et al. [2008**] présentent le problème comme un problème de la théorie des jeux. Dans ce cas, les algorithmes d'optimisation distribués risquent de ne pouvoir trouver un équilibre, voire de ne jamais se stabiliser sur une solution xe (les joueurs essayant perpétuellement d'adapter leurs comportements individuels aux comportements des autres).

1.1.5.6. Adaptation instantanée vs adaptation moyenne :

Nous distinguons dans cette section deux types de contrôleurs : les contrôleurs qui s'adaptent à des conditions moyennes de trafic (par exemple les flux moyens des véhicules) et ceux qui s'adaptent à des situations instantanées (par exemple un état particulier du système).

La plupart des travaux rencontrés optimise les phases des intersections en fonction des flux moyens dans les réseaux de transport [**Robertson, 1969; Hunt et al. ,1981; Gershenson, 2004**].

Néanmoins, nous avons rencontrés aussi des algorithmes tels que ceux fondés sur : l'apprentissage par renforcement [**Wiering, 2000; Steingrover et al. 2005**] qui s'adaptent à un état particulier du système, PRODYN [**Henry et al. 1984**] qui replanie en fonction de la

situation courante, ou les négociations [**Naumann et Rasche, 1997**] et les réservations qui se font directement entre véhicules autonomes.

1.1.5.7. Algorithmes d'optimisation :

Nous avons rencontré dans la littérature diverses approches pour optimiser la gestion d'intersections. Nous avons décidé de classer ces différents algorithmes en trois grandes classes :

-les algorithmes heuristiques, souvent fondés sur des règles prédéfinies par ses concepteurs, et qui ne garantissent pas d'optimalité (même locale) ;

- les algorithmes d'optimisation boîte noire, qui considèrent le problème comme une fonction à optimiser sans exploiter le fait qu'il s'agit d'un problème de contrôle ; exemple d'algorithmes : Hill-Climbing, Algorithmes évolutionnaires et Optimisation mathématique.

- les algorithmes de contrôle, qui visent à trouver les contrôleurs qui vont satisfaire certains critères d'optimisation exemple d'algorithmes: Apprentissage par renforcement : Q-learning Planification en ligne et Théorie des jeux etc.

1.1.5.8. Les limites des approches :

Les approches sus-cités (toutes les approches à part les quatre de référence) que nous appellerons les approches avancées, sont toutes très intéressantes et innovatrices. Nous voulons aborder, dans cette section, les problèmes rencontrés par ce type d'approches et donc leurs limites à une implémentation dans les systèmes de gestion de trafic réels.

-Complexité des solutions vs. Simplicité des modèles :

Dans beaucoup de travaux présentés, nous remarquons que la majorité des auteurs testent leurs algorithmes sur des simulateurs où plusieurs aspects du trafic réel ont été négligés an de réduire la complexité calculatoire, comme c'est le cas par exemple pour les approches issues de l'apprentissage par renforcement. En plus, même en simplifiant les modèles, l'exécution des algorithmes restent parfois aussi coûteuses en termes de temps de calcul.

-Manque de coordination entre intersections : Un autre problème de la plupart des solutions réside dans le fait qu'il n'existe pas assez de travaux qui traitent le problème de coordination

entre intersections. Beaucoup de travaux s'intéressent uniquement à l'optimisation locale (au niveau de chaque intersection) afin d'optimiser globalement la circulation des véhicules alors que cette optimisation n'est pas forcément systématique.

1.2. Approche pour véhicules autonomes (IAV) :

Dans cette section, nous considérons les solutions qui s'intéressent au contrôle de véhicules intelligents autonomes (IAV), Nous avons choisi de présenter les travaux spécialement dédiés aux IAV mais aussi les modèles qui, à notre sens, ne peuvent s'appliquer que sur ce type de véhicules.

1.2.1. Négociation véhicule à véhicule :

L'une des premières approches pour véhicules autonomes a été proposée par [**Naumann et Rasche 1997**]. Cette approche repose essentiellement sur la communication et la négociation entre les véhicules an de déterminer l'ordre de passage et de sortie des intersections [**Naumann et al. 1997**]. Dans ce travail, les auteurs modélisent une intersection comme un ensemble de trajectoires parcourues par les véhicules avec des zones critiques où on ne peut avoir qu'un seul véhicule à la fois. Dans la Figure5, les points noirs représentent ces zones critiques.

L'idée de base de cette approche est l'utilisation d'un protocole de communication type Anneau à jetons (Token-ring) entre les véhicules autonomes dans une zone déterminée autour de l'intersection. Ainsi, les informations essentielles sur la position, la vitesse et l'accélération des véhicules sont échangés en vue d'une décision sur l'opportunité ou non de s'engager dans l'intersection. Pour une exécution complètement décentralisée, le véhicule le plus proche de l'intersection et qui est en possession du jeton, prend le rôle de l'agent manager.

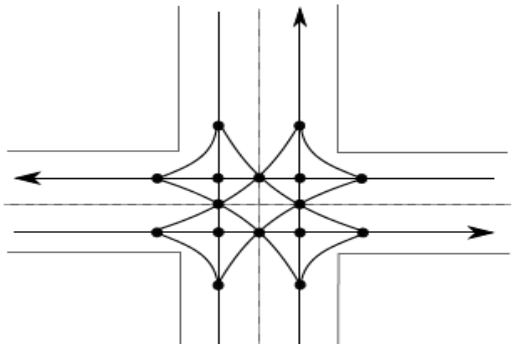

Figure 5: Modélisation de la géométrie des intersections.

Ainsi, il décidera de l'ordre de priorité de sortie de tous les autres véhicules de l'intersection. Sa tâche se termine juste avant de quitter l'intersection quand, à son tour, il doit alors passer le jeton à un autre véhicule. Par ailleurs, l'évitement de collision est assuré au moyen d'algorithmes fondés sur des sémaphores qui permettent à un seul véhicule à la fois de rester dans un segment critique d'intersection.

Cette approche est connue pour ses limites pratiques. Premièrement, elles dépendent clairement du nombre de véhicules voulant négocier leurs passages par l'intersection, car l'augmentation du nombre des véhicules implique une complexité croissante lors des négociations. Deuxièmement, elle suppose que toutes les intersections ont la même géométrie, ce qui n'est pas réaliste dans le contexte actuel (il n'y a pas que des routes avec une seule voie et des angles de 90°). Si nous avions plus de voies, nous aurions plus de calculs et plus de temps de traitement. Enfin, cette approche dépend aussi des capacités de communication : Peut-on assurer la réception de tous les messages envoyés par les véhicules ?

1.2.2. Approche par réservations :

L'approche par réservations a été introduite par [**Dresner et Stone 2004**]. Elle se passe des feux de signalisation conventionnels et à la place elle introduit un agent manager au niveau de chaque carrefour. Chaque véhicule voulant passer doit réserver un temps de passage et une route à prendre, et respecter un certain nombre de règles. Parmi ces règles, il doit garder une vitesse constante fixée à l'avance à l'approche de l'intersection et il ne doit pas changer de route (c'est à dire que les véhicules vont toujours tout droit). La réservation se déroule comme suit : Le véhicule informe l'agent manager en envoyant un message contenant sa vitesse, sa direction, son accélération maximale, sa décélération maximale, le temps qu'il lui reste avant d'arriver à l'intersection (estimé par l'IAV) et d'autres propriétés du véhicule. Chaque agent manager connaît les informations nécessaires sur la géométrie de son intersection. Il discrétise alors la zone de croisement sous forme de grille, comme le montre la Figure 6, an de pourvoir simuler la trajectoire des véhicules à l'intérieur de la zone de conflit.

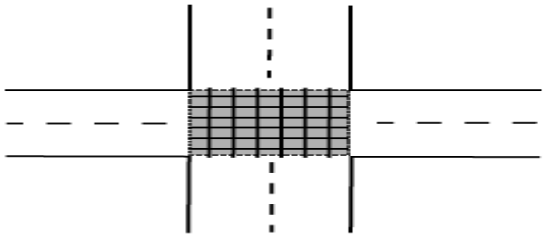

Figure 6:Discrétisation de la zone de croisement par l'agent manager.

Ensuite, l'agent manager simule le trajet du véhicule avec les autres trajets déjà réservés en fonction du temps (l'espace qui va être réservé). Si le trajet en question est en conflit avec d'autres réservations (c'est à dire l'espace discrétisé est réservé au même moment par deux véhicules), alors la réservation est rejetée. Le véhicule doit alors décélérer et demander une autre réservation. S'il arrive à l'intersection sans avoir une réservation, alors il doit s'arrêter. Si la réservation est acceptée alors le véhicule peut passer.

Dans [**Dresner et Stone, 2005**], certaines des hypothèses précédentes ont été assouplies : il n'est plus exigé de maintenir une vitesse constante dans l'intersection et la possibilité de changement de route a été introduite. Concernant la vitesse, dans la deuxième version de l'approche, l'agent manager peut valider une réservation si, en modifiant la vitesse, il ne risque pas d'y avoir de collisions. Le protocole amélioré proposé repose sur des règles que les véhicules sont censés suivre : le véhicule doit essayer de suivre les actions prévues par l'agent manager et il ne peut pas essayer d'améliorer son parcours (l'agent manager ignore une nouvelle demande si une réservation a été déjà validée).

L'avantage de cette approche est que, si on a plusieurs véhicules qui veulent passer et que leurs trajectoires ne se croisent pas, alors on peut satisfaire tout le monde avec un passage simultané. Cependant, dans le cas contraire, on est obligé de donner la priorité à un véhicule par rapport aux autres, ce qui revient à utiliser le principe des feux de signalisation.

1.3. Résumé des solutions citées :

Dans le tableau1, adapté d'après [**Bazzan 2009**], les travaux sont classés d'abord selon que l'approche s'intéresse : (i) à une intersection isolée à la fois, (ii) à un groupe d'intersections, ou (iii) à l'optimisation globale du réseau de transport. On y distingue aussi trois grandes classes d'approches :

-les approches de référence,

- les approches avancées divisées en trois sous-classes :

-les approches fondées sur la physique, les mathématiques ou les automates cellulaires,

-les approches multi-agents (SMA),

-les approches fondées sur l'apprentissage,

-et les approches pour véhicules autonomes (IAV).

Nous pouvons constater à partir du tableau2 qu'il manque beaucoup de travaux de recherche s'intéressant à l'optimisation globale des intersections, et surtout il n'en existe aucune à notre connaissance dédiée aux véhicules intelligents autonomes. C'est pour cette raison que nos travaux, dans les chapitres suivants, vont désormais s'intéresser au développement d'idées pour l'optimisation du réseau de transport d'IAV via les feux de signaux tricolores.

		Intersections Isolées	**Intersections connectées (optimisation. locale)**	**Optimisation. globale**
App. Référence		PRODYN		TRANSYT SCOOT SCATS
App. Avancées	**Physi.**	[Gershenson, 2004]		[Brockfeld et al., 2001] [Köhler et al, 2005] [Mohring et al, 2006]
	SMA	[Bull et al, 2004] [Doniec et al. 2005	[Kosonen, 2003] [Sánchez et Aguirre, 2007] [France et Ghorbani, 2003] [Bhouri et al, 2011] [Hounsell et Shrestha, 2012]	[De Oliveira et Bazzan, 2006] [De Oliveira et Bazzan, 2007]

			[Rochner et al, 2006]	
	Apprent.	[Da Silva et al, 2006]	[Camponogara et Kraus Jr, 2003] [Nunes et Oliveira, 2004] [Steingrover et al., 2005] [Wiering, 2000] [Ferreira et Khosla, 2000] [Richter et al., 2007]	[Bazzan, 2005]
App. IAV		[Dresner et Stone, 2004] [Balan et Luke, 2006] [Naumann et Rasche, 1997]		

Tableau 2: Résumé des solutions citées.

Conclusion :

Dans cette partie a été l'occasion d'étudier les approches déjà existantes pour la gestion d'intersections. Nous avons commencé par présenter les solutions traditionnelles à base de règles de coordination (stop, panneaux, etc..) lesquelles garantissent uniquement la sécurité des usagers de la route. Ensuite, nous avons introduit les solutions avancées pour la gestion du trafic et les approches conçues pour les véhicules autonomes, en mettant en avant les problèmes de réalisation mais aussi les limites posées par ces différents systèmes, Enfin, nous donnons une synthèse des approches dans un tableau récapitulatif.

Chapitre 2 : NetLogo &UML : Aperçu et Initiation

Introduction :

Dans ce chapitre, nous présentons les résultats de l'étude que nous avons faite sur une plateforme multi agent générique, nommé NetLogo. Ce plateforme est dédiée à la modélisation/simulation des phénomènes collectifs, une plateforme qui supporte et illustre la mobilité des agents, profitables et intuitif pour notre modélisation.

Le but de cette étude, est de découvrir ce plateforme générique d'un coté, et d'empoigner de l'autre coté l'architecture générale d'un programme Netlogo, ainsi que le lien fort entre cette plateforme et le langage de modélisation UML, retrouvé et recommandé dans quelque travaux et atelier mentionné dans la suite. Toute en mettant en exergue les différents agents qui composent cette plateforme multi agent, la structure d'un modèle NetLogo typique, quelque primitives utiles dans la plateforme, Quant est-ce qu'on construit un modèle multi agent facile à simuler, comment visualiser, contrôler les simulations de ces modèles établit, à travers quelque primitives.

Le but de ce chapitre est de familiariser le lecteur avec notre réflexion méthodologique exploitée dans le chapitre quatre, en terme de modélisation et d'implémentation afin de concrétiser notre simulateur de coopération ambitionné.

2.1.La Simulation Multi-agents :

La recherche classique tente de développer des programmes informatiques pour simuler le comportement d'un être humain intelligent par exemple, de ce fait, le but était de créer un système artificiel ayant les mêmes capacités qu'une personne intelligente. La simulation Multi agents selon M.Bouzid **[**J. P. Briot, Y. Demazeau**]** facilite l'étude des systèmes complexes constitués de plusieurs entités en interaction.

L'approche multi agents dans **[**Mohamed Rédha BAHRI**]**, ajoute au modèle une composante stochastique avec laquelle chaque agent va réagir différemment des autres, et son comportement sera l'expression de sa perception et ses compétences et par conséquent la vie de l'un des agents sera différente des autres et l'évolution du système global sera plus proche de la réalité.

2.2. Les plateformes de simulation Multi-Agent :

Dans le domaine de la simulation multi-agent **[**B. Chaib-draa, I. Jarras et B. Moulin**]**, les plateformes sont des outils méthodologiques importants, qui aident le chercheur dans le processus de modélisation et le développement des programmes de simulation**.** En d'autre terme, il présente des supports d'aide à la programmation. Elles fournissent une couche d'abstraction permettant de faciliter l'implémentation des concepts orientés agents.

2.3. NetLogo une Platform générique :

Nous nous sommes accentué dans ce chapitre sur une plateforme générique appelé Netlogo, téléchargée du site[4]. Après l'installation, nous avons distingué trois panneaux pour le cadre Netlogo, dont deux indispensables (code, et interface), et une (information) utile mais pas nécessairement capitale.

[4] https://ccl.northwestern.edu/netlogo/index.shtml

Maintenant, on va présenter brièvement ces trois panneaux de NetLogo.

3.1 Interface : une interface d'interaction avec le modèle et de représentation des résultats et des états du « monde ».

3.2 Information : une interface d'information des utilisateurs du modèle.

3.3 Procédures : une interface pour écrire les procédures du modèle (caractéristiques et comportements des agents et du système, leurs interactions,..).

Cette plateforme favorisée pour notre étude, constitue un outil logiciel puissant pour le développement des systèmes multi agents. Elle permet à des milliers d'agents de fonctionner tous indépendamment et en parallèle.

Il s'agit d'un environnement de modélisation et de simulation des phénomènes naturels et sociaux,…etc. Ces modélisation/simulations couvrent de nombreux domaines des sciences naturelles et sociales, y compris la biologie et la médecine, la physique et la chimie, et les mathématiques ainsi que l'économie.

2.4. L'Avantage de NetLogo :

Cette partie est extraite essentiellement du guide NetLogo retrouvé dans site[5]**.**

NetLogo est une plateforme multi-agent de référence, un environnement programmable permettant de modéliser et simuler des phénomènes naturels et sociaux. Il a été créé par Uri Wilenski en 1999, son développement est poursuivi de manière continue par le Center for Connected Learning and Computer-Based Modeling, dont la dernière version 5.0.4, daté mars 2016. En plus, NetLogo est gratuit et peut être téléchargé sur le site[6]**:**

Cette plateforme représente la nouvelle génération d'un ensemble de série de langages, de modélisations multi-agents qui a débuté avec StarLogo[7].

[5] https://ccl.northwestern.edu/netlogo/docs/

[6] https://ccl.northwestern.edu/netlogo/download.shtml

[7] http://ccl.du nord-ouest.edu / StarLogo

NetLogo étant écrit en Java, il peut tourner sur tous les systèmes d'exploitation majeurs (Mac, Windows, Linux,), Il fonctionne en tant qu'application indépendante. Les modèles peuvent même être sauvegardés sous forme d'applets Java et tourner dans tous les navigateurs internet modernes.

NetLogo est accompagné d'une documentation complète et de nombreux tutoriaux.

Elle est livrée avec un vaste tutoriel, et de *Bibliothèque de modèles* qui comprend une grande collection de simulations fonctionnelles pouvant être utilisées telles quelles ou modifiées.

L'intérêt majeur de cette plateforme générique provient de l'ensemble des primitives, et les outils offerts, qui répondent bien à nos besoins symptomatiques, notamment en termes de mobilité, par des agents mobiles préprogrammés nommé 'Turtles' détaillé dans les paragraphes suivant.

Comme il a été discuté ailleurs **[**Franklin, Stan and Graesser**] [**Brooks, R**],** nous avons trouvé NetLogo convenable pour modéliser des systèmes à base d'agents. à l'égard, chaque agent NetLogo:

- Perçoit son environnement et actes sur son environnement,
- Porte son propre fil de contrôle,
- Autonome, il matérialise la définition classique d'agent trouvée dans **[**Labidi, S., Lejouad, W**].**

NetLogo est un outil excellent pour le prototypage initial et rapide, qui expérimente les systèmes multi-agent. Elle convient particulièrement à des systèmes avec des agents situés et qui opèrent dans un espace restreint. Elle représente aussi un excellent outil d'animation du système à modéliser.

La plateforme offre les éléments de base d'un système multi agent, en termes d'agent, d'environnement, et de lien de communication. Elle laisse aux développeurs l'esprit de penser aux stratégies, aux protocoles, méthodes, modèles, comme ils veulent.

Bien évidemment, on ne peut pas trouver des plateformes dédiées prêtes pour tous les problèmes dans tous les domaines.

l'idée clé comme dans notre démarche est de développer ultérieurement derrière cette plateforme, également de reproduire un simulateur dirigé principalement par des concepts de cette plateforme générique, consacrées au problème de simulation inspirés, examinés surtout en termes de coopération ambitionné, toute en profitant au maximum des primitives, des agents, l'environnement exhibé par ce cadre multi agent magique. Est pour effectuer ce but, il faut faire un aperçu sur l'ensemble des agents NetLogo, qui sera l'objectif du paragraphe suivant.

2.5. La classification des agents NetLogo :

L'Univers NetLogo peut être composé de différents types d'agents. Dans **[**Brahim Chaib-draa, Imed Jarras et Bernard Moulin**]** il existe principalement quatre types d'agents : les tortues, le monde (environnement), les patchs, les liens.

2.5.1. Les tortues (Turtles): De notre point de vue, ils représentent les «vrais» agents, qui se déplacent dans l'environnement. Ce sont effectivement nos nœuds mobiles particuliers, ils ont des caractéristiques, par exemple ils naissent, meurent, ils peuvent être de plusieurs types (breeds, consommateurs, firmes).

Chaque tortue parvient avec certains primitifs (l'attribut ou fonction) encastrés. Dans NetLogo les "primitifs" sont des variables (attributs) et des instructions (ordres) qui sont construits dans la langue NetLogo, à partir de la "sémantique" de ce dialecte Netlogo.

Chaque tortue apparait avec beaucoup de variable et des procédures encastrées (fonctions). Pour distinguer les attributs pour une tortue, vous pouvez cliquer à droite de la tortue, et de choisie inspectez.

Ce qu'il faut souligner sur les tortues :

- Les tortues doivent être créées; ils n'existent pas par défaut.
- Les tortues déplacent à travers les pièces (patches) dans l'environnement.
- Une tortue est identifiée par son ID, non pas par ses coordonnées (xcor, ycor).

2.5.2. Le Monde (World) : L'environnement est une grille à deux dimensions et est divisé en patches. Il correspond à la représentation spatiale (en 2D ou 3D) de l'environnement modélisé.

2.5.2.1. Les Patches: Ce sont les composantes spatiales du monde modélisé, les agents se déplacent sur les patches et les patches peuvent stocker des variables, c-a-d chaque patch reflète un agent qui représente un point du monde sur lequel les tortues peuvent se déplacer, ils peuvent être personnalisés par l'utilisateur.

Ce qu'il faut souligner sur les patchs :

- Les patches ne sont pas comme les turtles, i.e doivent être crées. Alors que les patches existent par défaut.
- Les patches sont des agents non mobiles, c.-à-d. il ne déplace pas dans l'environnement.
- Les patches ont une fonction interne nommé **Sprout** permettant de créer des tortues.
- Les patchs sont identifiés que par les coordonnés (xcor, ycor) et non pas par des identifiant (ID) comme les turtles.

2.5.2.2. Les liens (links): un type particulier d'agent qui relie deux agents et qui est représenté comme une ligne dessinée entre ces agents. Ce lien peut être orienté ou non. Les liens servent à faire communiquer physiquement deux agents.
Jusqu'à maintenant nous n'avons présenté que l'ensemble des agents NetLogo, la partie mise en œuvre d'un modèle NetLogo n'est pas dévoilé, la partie suivante présente succinctement la procédure global de mise en œuvre d'un modèle NetLogo.

2.6. Le modèle Net Logo, les étapes de mise en œuvre:

La mise en œuvre d'un modèle multi agent NetLogo se déroule en quatre étapes :

Premièrement : il faut décrire l'interface qui contient tout ce qui concerne la visualisation, ainsi que les paramètres que l'utilisateur pourra modifier.

Deuxièmement : on peut décrire des variables globales, et on peut personnaliser les agents en définissant leurs propriétés.

Troisièmement : il faut décrire ce qui se passe quand on prépare la simulation, initialiser les variables, créer les agents, préparer l'environnement. Généralement cela peut être fait en utilisant le bouton Initialiser(**Setup**).

Finalement : on doit décrire ce qui se passera dans chaque pas de simulation, et décrire ce que la plateforme affichera aux utilisateurs, évidemment cela peut être fait en employant le bouton démarrer (**Go**).

Pour bien allumer ces quartes étapes, un modèle Net Logo typique est composé de plusieurs blocs comme dans **[34]**, ces différents blocs sont éclairais comme suite :

2.6.1. Une interface graphique :

Une interface qui contient les éléments permettant à l'utilisateur de fixer les valeurs des paramètres du modèle et d'observer période par période les sorties.

Par exemple créer l'interface graphique pour la société des fourmis avec au moins : Un champ pour fixer nombre-fourmis (population),Un champ pour fixer chacun des autres paramètres (taux d'évaporation, taux de diffusion),Un bouton « setup/ou initialiser » pour déclencher l'initialisation.

Un bouton « go/ ou démarrer » pour commencer le déroulement de l'histoire.

2.6.2. Déclarations des variables globales et individuelles :

Déclarer des types d'agents et des variables spécifiques de chaque type d'agent, un exemple de déclaration des variables globales est comme suit:

Globals [**Nbr_Robot**].

Les variables fixées dans l'interface graphique sont automatiquement des variables globales. Pour la création des types d'agents, Netlogo offre des primitifs spéciale, comme le primitif **'Breed'** par exemple:

Breed [robots robot] ; permette de crée un ensemble de robots

Création des variables individuelles des agents par le primitifs 'own' Par exemple :

agents-own [energie],

Chaque agent à un niveau d'énergie qui représente une variable individuelle et non pas globale.

2.6.3 Le Procédure setup :

Préparer le modèle pour l'exécution par procédure (en même temp un bouton 'setup').

Exemple :

```
To setup ; execute par le bouton « setup »
        (Initialisations, instructions)
End
```

- **Initialisation** des variables globales (lecture automatique au début de chaque exécution à partir de l'interface graphique ou fixation de leur valeur) ;
- **Création** des populations de chaque type d'agents et initialisation de leurs variables individuelles ;
- **Initialisation** du compteur de périodes et des sorties diverses graphiques.

2.6.4 Le procédure (Go) :

La procédure "**Go**" où démarrer, regroupe toutes les opérations qui ont lieu, pendant une « période » d'exécution du modèle et qui incrémente le compteur de périodes (ticks).

Ce qu'il faut repérer dans les modèles NetLogo :

- On utilise des commentaires pour documenter le modèle : '**;**' ceci est un commentaire, toutes les expressions écrites après les commentaires seront ignorés par le programme NetLogo, ces expressions seront seulement là pour décrire ce que les instructions sont destinés à faire.
- Dans les modèles NetLogo, ce qui concerne le "**SETUP** " et "**GO** " sont appelé des procédures principales. en effet, il y des procédures auxiliaires appelé des procédures complémentaires, où un ensemble d'instruction est regroupé sous un nom désigne l'activité de la procédure complémentaire.

2.7. Le Langage NetLogo et langage UML :

Ces dernières années, les méthodologies multiagents, et notamment les plateformes derrière eux, beaucoup de langages de développement orienté agent, ont profité du langage de spécification oriente UML. Dans cette partie, est de citer quelque travaux étudiés, présente un rapporte et un lien entre le langage UML et le langage NetLogo. On va expliquer deux ateliers **[**David.Sheeren**]** **[**F.Amblard, N.Marileau**]** retrouvé dans la littérature multiagent, afin d'éclairer les relations entre ces deux langages orienté objet (UML) et orienté agent (NetLogo).

2.7.1. Atelier 1 [David.Sheeren] :

Dans Atelier 1 **David.Sheeren** à présenter et expliquer une modélisation UML pour un phénomène collective proie-prédateur **[**Wilensky, U. & Reisman, K. (2006).**]** (loup-moutons), Sachant que ce modèle et déjà formalisable sur Netlogo (dans la bibliothèque NetLogo). Après la description du phénomène comme suit:

«Dans une *prairie*, se trouve des *moutons* et des *loups*: les moutons et les loups se *déplacent*, si un mouton rencontre de l'*herbe*, il la *mange.* Si un loup rencontre un mouton, il le *mange*. Dés que le loup ou le mouton à dépassé *son seuil* de production, il se *reproduit*. Notons que le herbe se *régénère* comme il annonce D.Sheeren ».

Le chef d'atelier D.Sheeren, et après une analyse du phénomène proie-prédateur, une analyse mentionné par des mots en italique dans notre description précédente, les mots en italique représente soient des agents, des objets, des attributs/méthodes pour les agents.

Pour D.Sheeren, est pour modéliser le composant spatial prairie il préconise la représentation de la **Figure 7 (c),** qui préserve les instances, seul les états qui change, contrairement au **Figure 7 (a)** où il n'y pas de représentation intra-praire (avec et sans herbe), aussi bien la **Figure 7 (b)** où la dynamique est représenté explicitement, i.e création/destruction d'instance en continue.

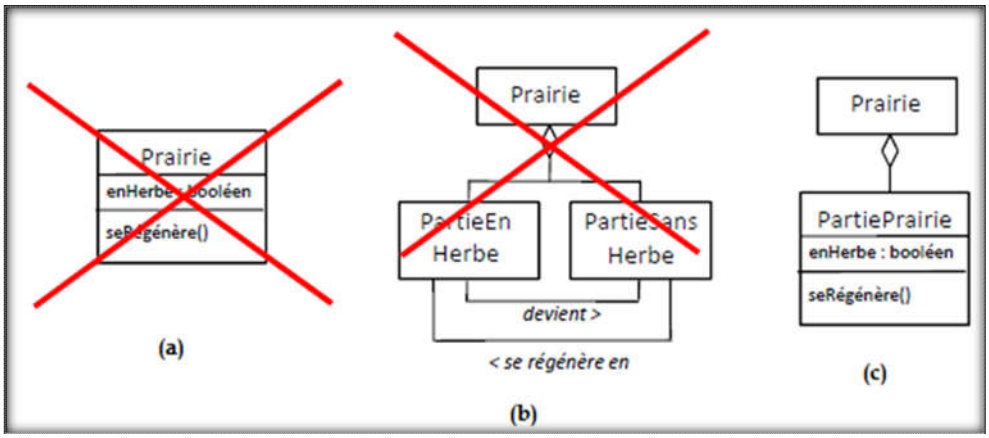

Figure 7:Différentes solutions pour modéliser la composante spatiale prairie.

Ensuite, D.Sheeren est pour modéliser l'ensemble des agents mobile de ce phénomène, il préconise non pas rassembler les agents mobiles dans un seul classe comme dans la **Figure 8 (a)**, mais il recommande d'éclater deux classe différentes comme dans la **Figure 8 (b)**, par ce que il ya deux comportement différents (les loups et les moutons). Il y a trop de différences entre les loups et les moutons, il faut les éclater en deux classes différentes, une pour les moutons et l'autre pour les loups, malgré que les attributs et les méthodes sont similaires.

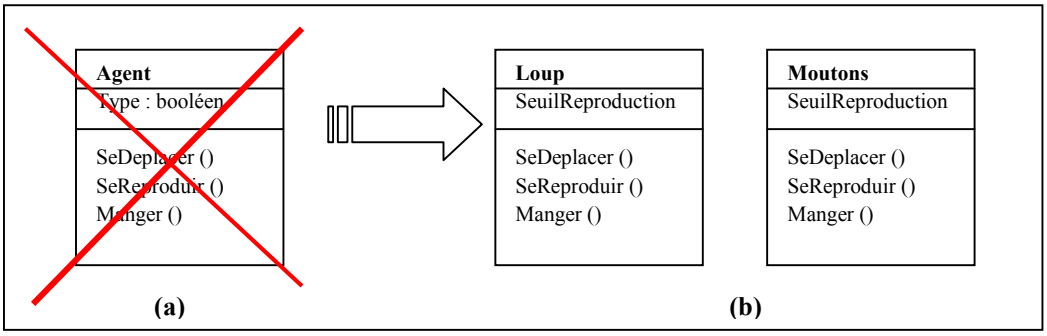

Figure 8: l'éclatement de deux composants, selon leur espèce.

Pour modéliser le phénomène proie –prédateur, en terme de dépendances entre les classes, David à appuyé sur un diagramme de classe UML présenté dans (**Figure 9**), quant à exhiber la relation entre les trois composants du phénomène. À la suite il propose un diagramme de classe générique (**Figure 10**), pour rapprocher le modèle d'implémentation NetLogo. Se qu'il souligné sur ce juxtaposition de ce premier atelier **[**David.Sheeren**]**, il reste en terme abstractif seulement, et non pas en terme d'étapes organisé comme dans le deuxième atelier **[**F.Amblard, N.Marileau**]**

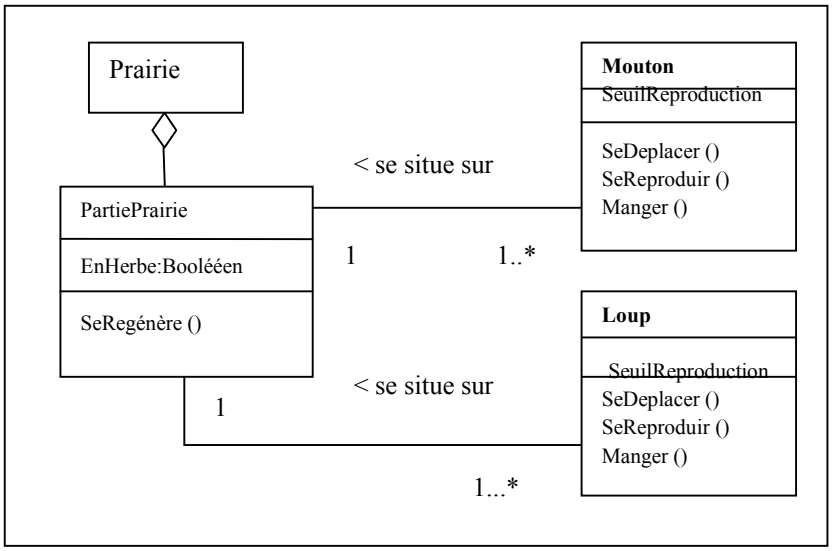

Figure 9: Le diagramme de classe du phénomène proie –prédateur.

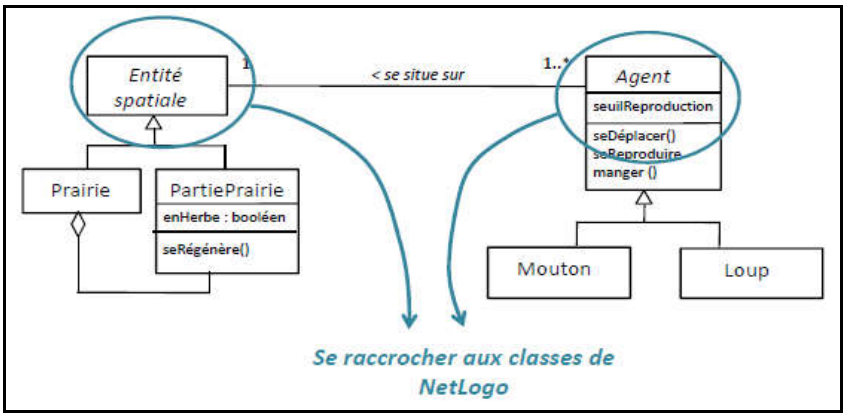

Figure 10: Diagramme de classe générique.

Pour modéliser la dynamicité du phénomène dans ce premier atelier, le chef d'atelier recommande fortement l'utilisation du diagramme d'activité, pour de montrer non seulement l'ensemble d'activité réaliser par l'ensemble des agents mobiles (loups, moutons), Mais il va aller jusqu'à l'abstraction du squelette du programme Netlogo principale, comme dans la figure (**Figure 11**) suivante.

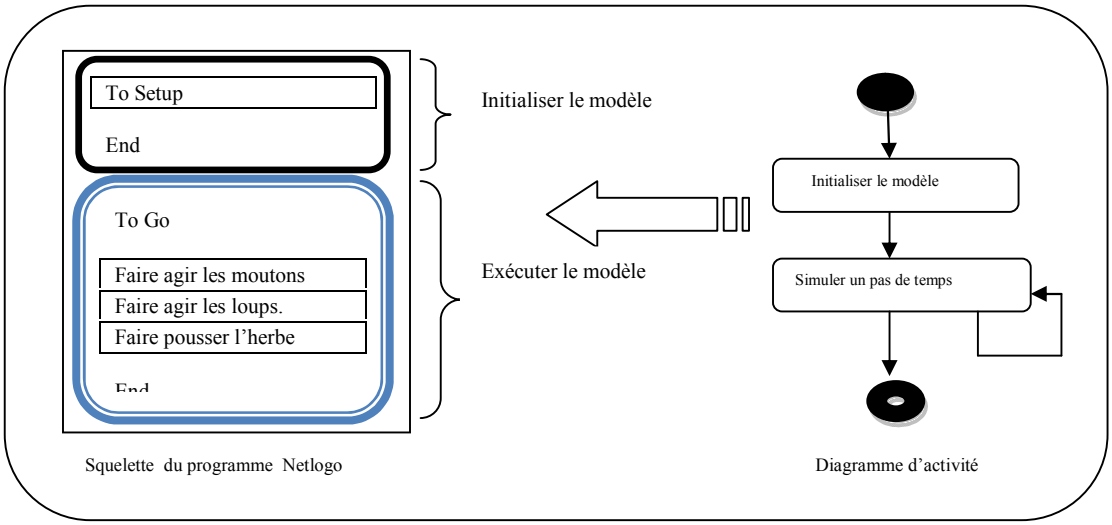

Figure 11: Concordance de diagramme activité & programme Netlogo.

Selon David **[**David.Sheeren**]**, le dynamisme de comportement des loups/moutons, peut être représenté par les diagrammes d'activité de la **Figure 12**.

Ce même diagramme est appliqué aux loups, mais avec le changement au niveau de condition (présence des moutons), sachant que même ces sous activités (manger, reproduire…etc.) sont modélisés par des diagrammes d'activité d'UML pour les décortiquer comme dans la **Figure 13**.

Dans ce premier atelier **[**David.Sheeren**],** ce qu'on peut dire comme premier constat, c'est que la modélisation UML du phénomène en terme de diagramme de classe et d'activité, produit un outil d'annotation puissant pour nous, afin d'éclairer un petit peu la structure de phénomène, les comportement (théoriquement), la dynamicité et la mobilité des agents...etc.

Hélas ces différentes modélisations UML, ne s'appuient pas sur une théorie formelle, ou un outil pragmatique pour valider ces modèles, il reste seulement des digrammes explicatifs et illustratifs.

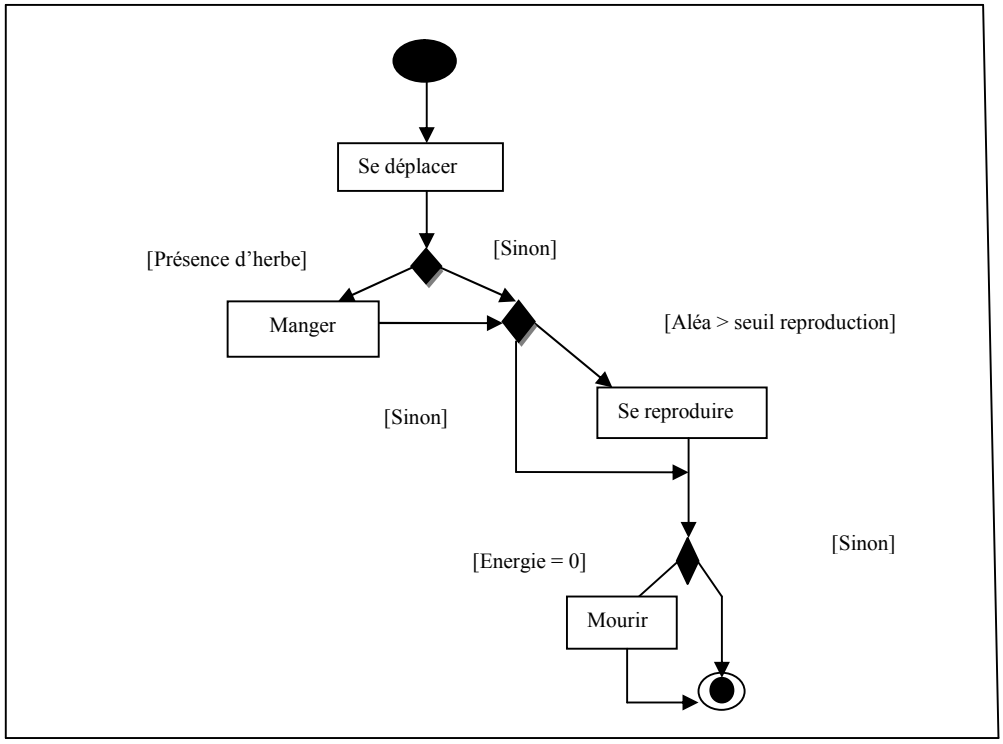

Figure 12: Diagramme d'activité pour les moutons.

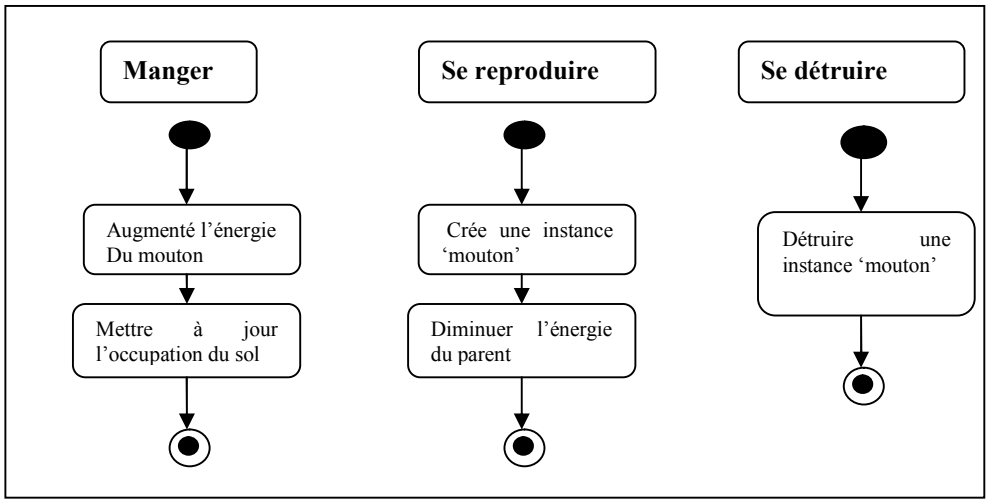

Figure 13: Diagramme d'activité élémentaire pour les moutons.

2.5.1. Atelier 2 [F.Amblard, N.Marileau] :

Passant maintenant au deuxième atelier, à ce stade Amblard et Marilleau ne s'arrêtent pas à une analyse, conception UML, mais il étend jusqu' à une proposition d'une démarche de passage d'un langage UML vers la plateforme NetLogo.

Au début de cet atelier, et comme nous l'avons souligné dans le chapitre quatre de ce mémoire, l'immense décalage existent entre la représentation conceptuel de la tâche et la représentation d'implémentation des systèmes modélisé. Ce décalage à poussé Amblard et

Marilleau d'exprimer et de s'articuler le phénomène souhaite modélisé sur des modèle conceptuel particulier, un modèle qui doit être proche du modèle opératoire (d'implémentation) en terme d'abstraction, de simplification, de sélection des concepts pertinents.

Avant d'énumérer les étapes de la démarche de passage d'UML/NetLogo d'Amblard, il est utile d'exhiber l'ossature netlogo (**Figure 14**) comme il le présente dans cet atelier.

Cette ossature Netlogo présentée par Amblard, illustrer par un diagramme de classe UML, nous permet d'agencer mentalement, tout ce qui à été dit au début de ce chapitre, édifier particulièrement les différents composants NetLogo.

Dans ce qui suit, nous présentons les étapes de la démarche de F.Amblard, N.Marileau, qu'on peut situer comme une bonne charte pour notre approche proposée dans le chapitre six suivant, et qui à notre avis, constitue une remarquable assiette, à partir duquel nous pouvons implémenter notre simulateur de coopération. Cependant ce qu'il faut noter ici, c'est que le chois d'explication et de spécification du problème qu'on souhaite modéliser, est directement influencé par les contraintes techniques imposées par NetLogo, comme le mentionne Amblard.

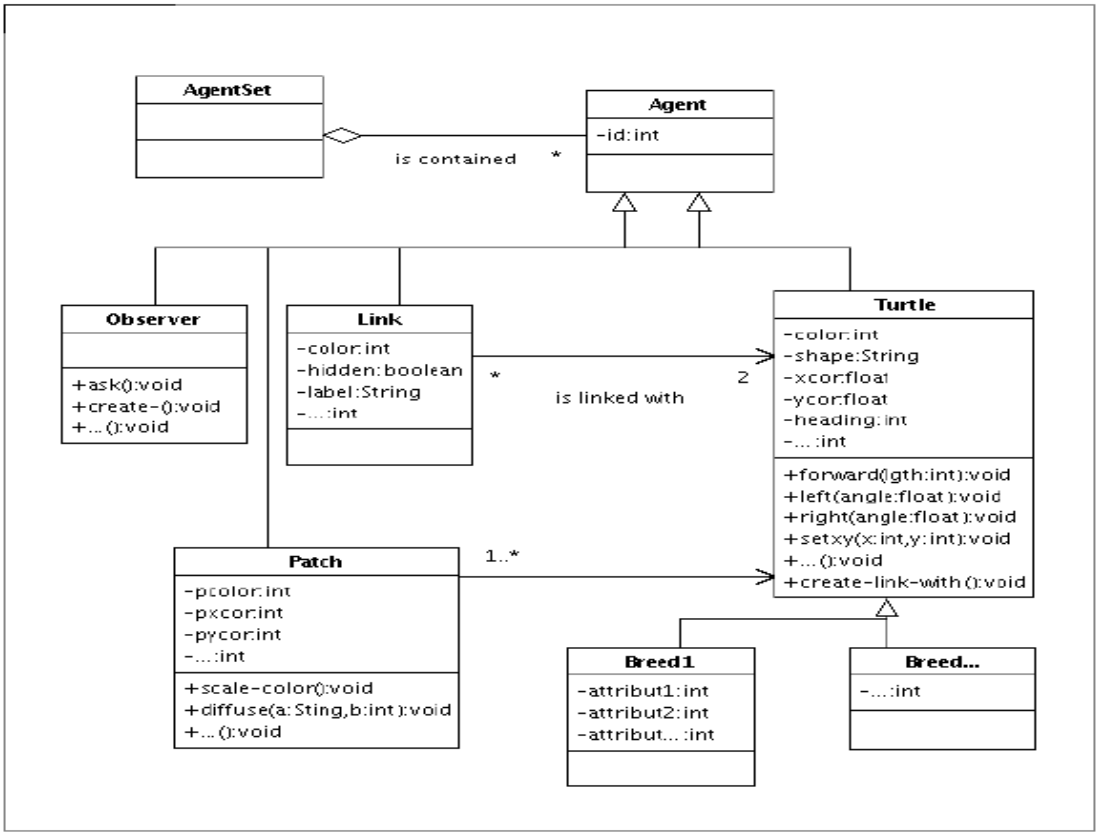

Figure 14: Une Ossature Netlogo.

Les étapes de la démarche [F.Amblard, N.Marileau] :

1. Identifier les variables globales: À partir des diagrammes de classe et d'activité accomplie. Essaye d'extraire les variables globales, qui peuvent être partagé par L'ensemble de composant du phénomène souhaite modéliser, par exemple le variable temps pour notre modélisation, ce variable globale indique le temps consommé.

2. Identifier les propriétés des objets à modéliser : cette identification doit être en termes de structuration/organisation, ou en termes de mobilité, par exemple :

Si la structure à modéliser est un réseau, il est fortement recommandé de modéliser le système par des composants de type 'LINK', alors que dans le cas où le système est assimilé à une grille régulière, comme celle de notre cas, il est recommandé donc d'utiliser Les 'PATCH', pour modéliser l'environnement.

Si on parle de mouvement des agents alors, on peut modéliser les agents comme des turtles, c'est le cas où ces agents sont mobiles. Sinon, il suffit de le modéliser sous forme de patch seulement, s'ils sont immobiles. Où même des turtles à l'exclusion des primitives de mobilité (**Forward**…etc.).

3. Organiser les entités entres elles : c'est l'étape clé de ce passage inspiré, respectent les deux contraintes suivant

– Un seul cran/degré d'héritage possible à partir de Turtle

– Pas de comportements attribués à une liste d'agents (AgentSet), ce sera à faire dans le programme principal.

Exemple:

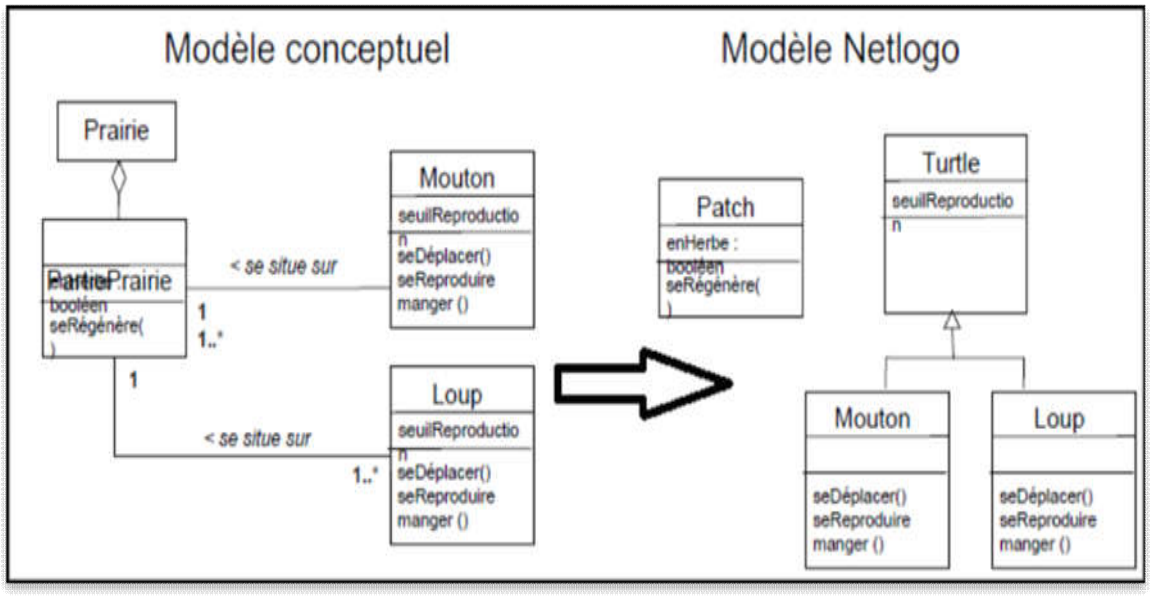

Figure 15: Exemple de passage UML / NetLogo.

Avant d'achever ce chapitre trois, et après la présentation du langage/plateforme Netlogo et les deux ateliers multi agent[F.Amblard, N.Marileau] [David.Sheeren]**,** on peut dire que :

NetLogo présente un langage et une plateforme pour développer des applications multi agents, ceci est découle essentiellement de ces différentes primitives, offertes par cette plateforme, en terme de mobilité surtout de ces agents incarné par des turtles, et en terme de présentation de l'environnement, …etc.

Certain travaux/ateliers récents tentent de créer une correspondance entre cette plateforme/langage et le langage UML, comme nous avons distinguait dans les deux ateliers [F.Amblard, N.Marileau] [David.Sheeren]**.** A travers ces deux ateliers nous avons identifiés une démarche théorique pour passer d'un langage UML (diagramme de classe, et d'activité) vers langage Netlogo. Cette démarche sera adoptée amplement dans notre approche pour développer notre simulateur **SimCirc**.

Conclusion :

Dans ce chapitre, nous avons exposé une étude et un apprentissage que nous avons accompli sur la plateforme générique de modélisation/simulation multi-agents Netlogo.
Le but de cette expérimentation est de comprendre, animer, et saisir les différents types d'agents qui constituent cette plateforme, ainsi que la structure générale d'un modèle multi agent Netlogo.

Dans ce chapitre nous avons sommairement expliqué deux ateliers [F.Amblard, N.Marileau] [David.Sheeren] retrouvé dans la littérature multi agent. A travers ces deux ateliers nous avons d'un coté montré le lien entre UML et Netlogo, et de l'autre coté, envisagé une démarche théorique quant à décortiquer la problématique d'optimisation des feux de signal à travers la plateforme Netlogo.
Le chapitre suivant est consacrée à la démarche usitée pour illustrer notre modèle multi agent, ainsi qu'à l'approche où la méthode employée pour illustrer la simulation de notre la circulation.

Chapitre3 : Démarche & implémentation

Introduction :

Dans ce chapitre présent la modélisation du notre stratégie suivant une démarche UML/Netlogo. La stratégie est fondée sur l'optimisation de temps dans une intersection accompagné par un capteur d'état de la route. Enfin présentation du simulateur **''SimCirc''** et les résultats obtenue.

3.1. Cycle de Vie d'un Logiciel Multiagent :

Selon [Gauthier picard]**,** Les principales phases d'un processus de développement sont : l'analyse des besoins, la conception de l'architecture, la conception détaillée, le développement (ou implémentation) et le déploiement.

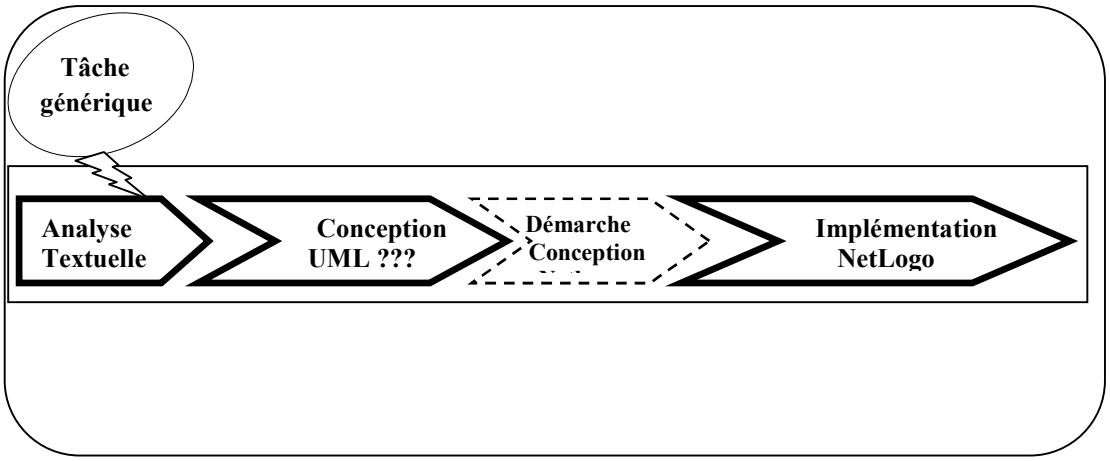

Figure 16: Processus de développement de notre modèle.

Somme tout, notre processus suivi, est illustré par la Figure précédente (**Figure 19**).
Préalablement et avant d'expliquer les quatre étapes de notre processus de développement inspiré, il est important d'éclairé ce que signifié les points d'interrogation mentionné dans la partie conception de ce processus. En effet il désigne pour nous le non confirmation du modèle spécifié (au sens formelle), particulièrement en termes de diagramme de classe, et d'activité indiquent la tâche générique souhaitent modélisé.

3.1.1. Tâche générique :

Les tâches génériques fournir des tâches de référence “benchmark”, pour évaluer les architectures de contrôle proposées dans la littérature. . Alors que dans notre cas, et pour un contexte modélisation et développement multi agent, ces tâches génériques sont censées comme animateur de l’ensemble des étapes d’un processus de développement (d'un simulateur orienté agents). i.e, manipuler les quatre étapes de notre processus, surtout la première étape d’analyse, afin d’inciter le processus. C’est on peut montrer une épithète opérationnel, Les tâches génériques indiquent l’essence pour n’importe quel type de modélisation multi agent.

3.1.2. Description de la tache générique :

Il s’agit de réécrire notre tâche (circulation routière) destiné dans notre modélisation attendu, décrit comme suite [Juan C. Burguillorial et al]:

« Dans un environnement borné se trouvent une intersection, des routes (gauche, droite, haut, bas), feux de signal (quatre), des voitures, des passages piétons (huit), des piétons et des quartiers. Le feu de signale déclenche soit automatiquement (changement des feux en deux couleur seulement rouge-vert) selon un temps donné, sinon intelligemment (notre vision astucieuse) si elles détectent un encombrement (un nombre de voitures est assez supérieur au niveau de la vois stoppé par le signale rouge)

Le passage piétons déclenche soit automatiquement (changement de couleur seulement blanc-rouge) selon un temps donné, le blanc pour le traverse autorisé des piétons et interdit pour voitures, et vise vers ca pour la couleur rouge, sinon intelligemment (notre vision astucieuse)

Les voitures circulent aléatoirement (avec des paramètres d'accélération et de décélérations). si elles détectent une intersection, elles doivent le respecter, sinon si elles détectent un passage elles doivent le respecter, sinon elles continuent la circulation,

En plus chaque voiture peu avancé (doubler) autre aux conditions de vitesse et d’encombrement. Les piétons circule dans les quartiers en cherchent des passages piétons, une fois le passage trouvé, ils doivent le respecter »

3.2. Analyse de la tâche :

il s'agit d'une de relecture de la circulation ambitionné. Une relecture attentive de notre tâche, nous permet d'éclater:

- Quatre thèmes (Voitures, Feux de signal, Piétons, Passages piétons).
- Des attributs pour les différents thèmes.
- Des relations entre les thèmes.
- Des Méthodes pour les thèmes (voiture, feux de signal, …. etc.).

1. Voiture: Un agent mobile peaufiné par la suite par un Turtle NetLogo. Ces voitures se déplacent d'une manière aléatoire, avec une vitesse maximum (vitesse-limite), chaque voiture "**N**" contrôle sa vitesse selon son emplacement (devant feux, passage, autre voiture, etc..):

Les comportements possibles de nos Voitures :

A- Les Voitures devant une autre voiture :

- la voiture "**N**" va accélérer et dépasser la voiture "**N+1**" lorsque la vitesse de "**N+1**" inferieur a "**N**", sinon il respecte sa vitesse et ne fait aucun dépassement.
- **N** Accélère lorsque la vitesse de "**N+1**" est supérieur a celle de "**N**".
- **"N"** décélère sinon arrêté, lorsque la vitesse de "**N+1**" Diminue (des fois vitesse devient nul) a celle "**N**".

B- Les Voiture devant les passages piétons :

- Travers avec la couleur rouge (passage), et stop avec la couleur blanc.

C- Les Voitures devant les feux de signal :

- Avance avec la couleur verte (panneau), et stop avec la couleur rouge.
- Il se peut que nos voiture change la direction vers la droite.

2- Feux de signal :

Connaître les signaux optiques circulation dans la plupart des pays du monde avec deux couleurs dans notre étude (sans orange) et non pas avec trois couleur comme dans notre vie quotidienne, et la couleur vert signifie traverser la voiture.

Dans notre étude (la vision autonomique), un capteur est accordé a nos feux de signal a fin que ces feux soient capable de gérer la situation de nos routes en temps réel, a ce stade si le feu de signal avec un couleur rouge détecte un encombrement au niveau de la route bloqué, il va volontairement déclencher la stratégie (rendre la couleur verte) est vice-versa.

3- Passages piétons :

ce sont des agent immobile pour notre modélisation, Ils fonctionne selon deux couleurs: le blanc pour le traverse autorisé des piétons et interdit pour voitures, et vise vers ca pour la couleur rouge .ils déclenche automatiquement ou intelligemment. Le déclenchement automatique avec changement de couleur selon un temps donné, intelligemment (notre vision astucieuse), chaque passage a des capteurs une pour les voitures, l'autre pour les piétons, changement de couleur selon les capteurs.

4- piétons :

Ils circulent librement (dans toutes les directions), et qui cherchent au fur et à mesure des passages piétons pour traverser les routes, avec une vitesse et un temps de circulation limitée.

Si un piéton détecte un passage de piétons dans un premier point au voisinage de environnement de mouvement (quartier) il va le traversé s'ils avec la couleur blanc, et stop avec la couleur rouge.

Thèmes	**Attributs**	**Méthodes**
Voitures	Vitesse	move-voitures
	V-max	verifier_vitesse
	Tourné?	peu-tourné-droit?
	Att_péitons	verifier_passage_péiton
	Att-inter?	courbe-attd-voitures-intersection
	Stop?	

Feux	CFHG CFHD CFVA CFVD Position_Feux	control-traffic-automatique control-traffic-autonomique Changé_couleur vision_voiture_feux
Passage Piétons	Capteur_piétons_passage Capteur_voitures_passage Adresse	select_vision_passage_p capteur_voiture_piéton_passage_p changé_coleur_passage_autonomique changé_coleur_passage_automatique changé_coleur_passage_p
Piéton	Vitesse_p Temps_mouv_p Pass Att_passage_p?	travercer_route marcher move-péitons courbe_attend_pietons
Environnement **(Patches, Globale)**	Role role_passage_p Temps	

Tableau 3: Les différents thèmes pour analyser la tâche générique.

3. 3. La conception:

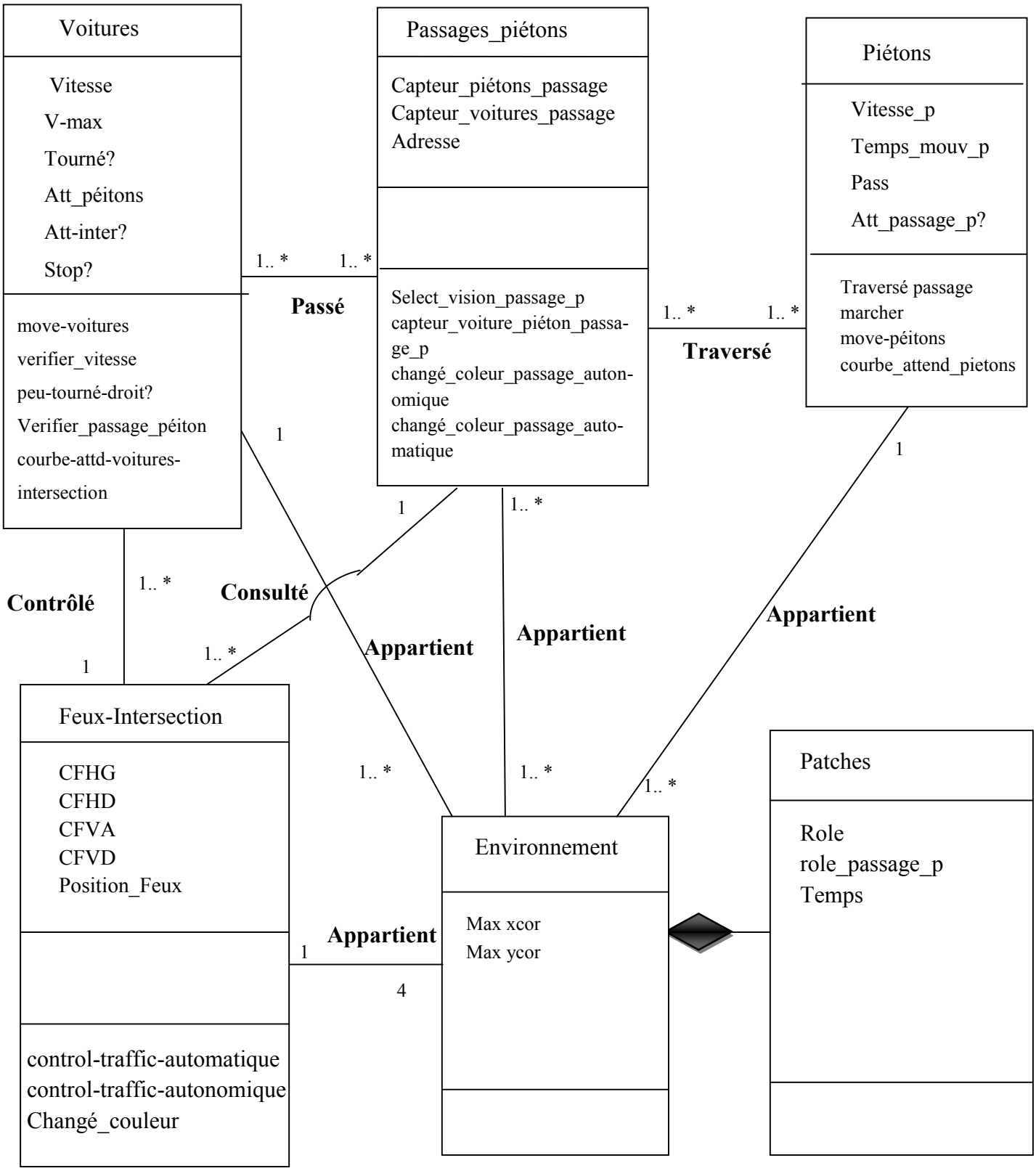

Figure 17:Diagramme de classe de la tâche générique.

Diagramme d'activité :

1. Voitures:

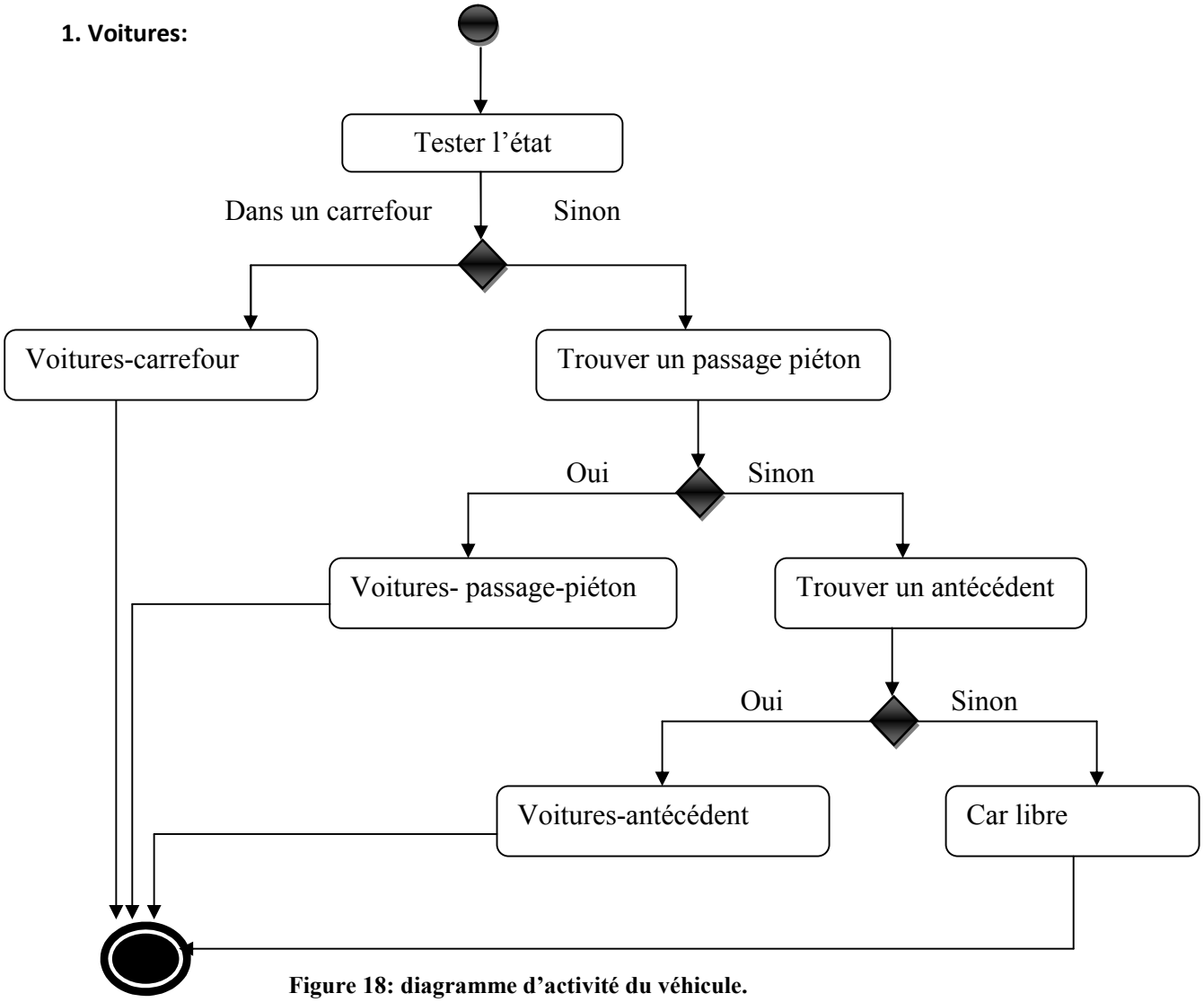

Figure 18: diagramme d'activité du véhicule.

a)- Voitures-carrefour:

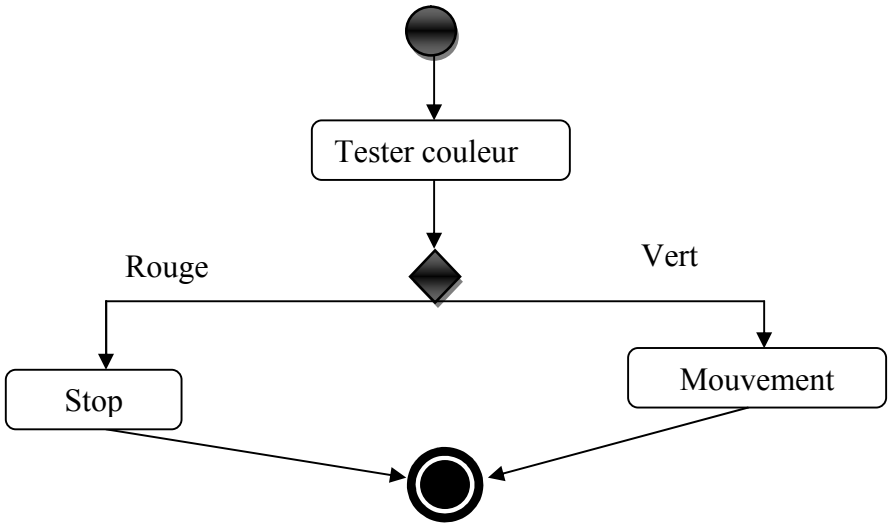

Figure 19: diagramme d'activité de la tâche Voitures-carrefour.

b)-Voitures-passage-piéton :

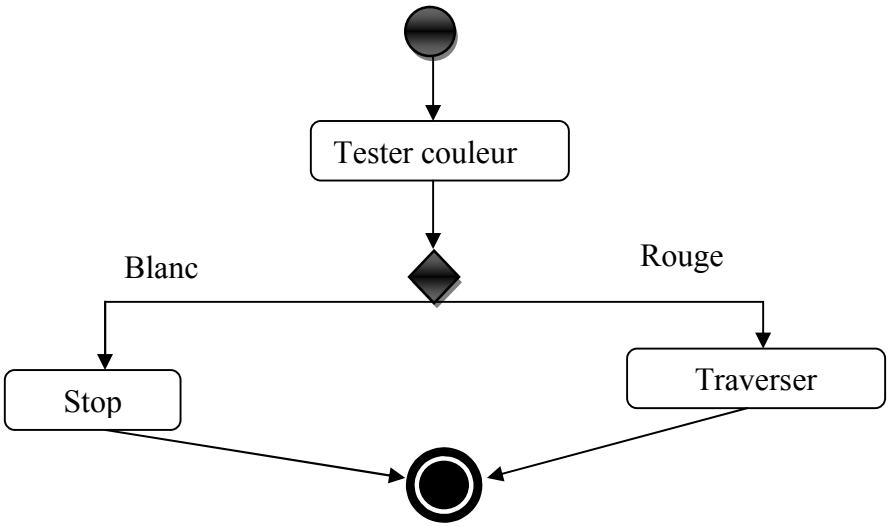

Figure 20: diagramme d'activité de la tâche Voitures-passage-piéton.

c)- Voitures-antécédente :

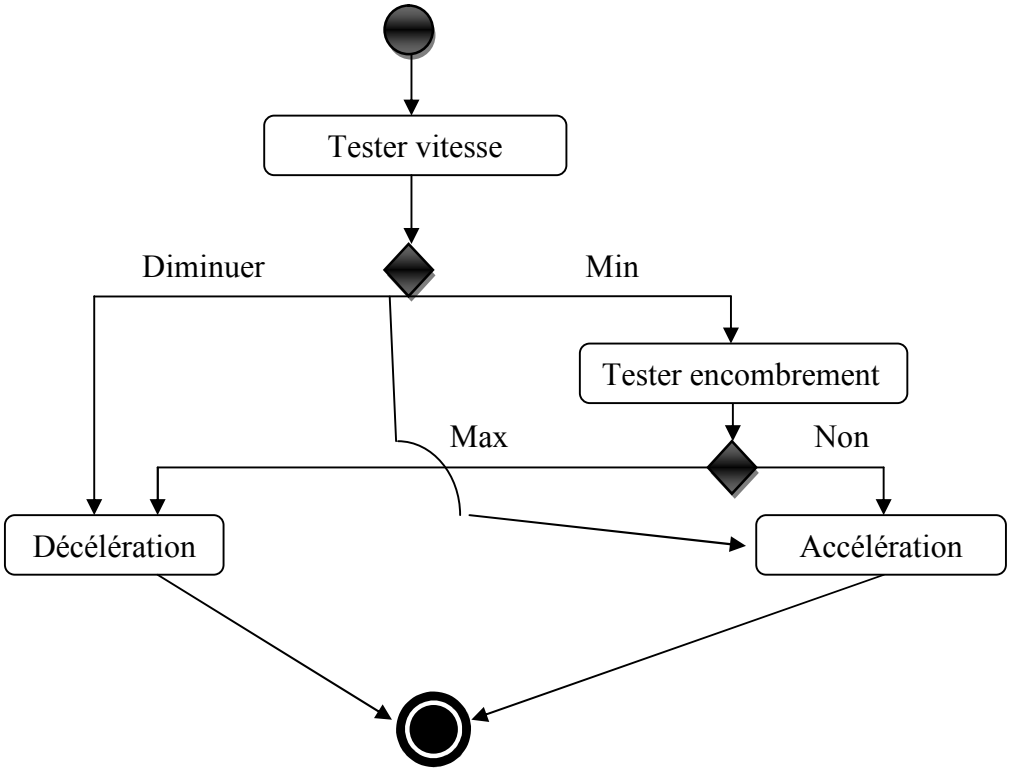

Figure 21: diagramme d'activité de la tâche Voitures-antécédent.

2. Feux de signal:

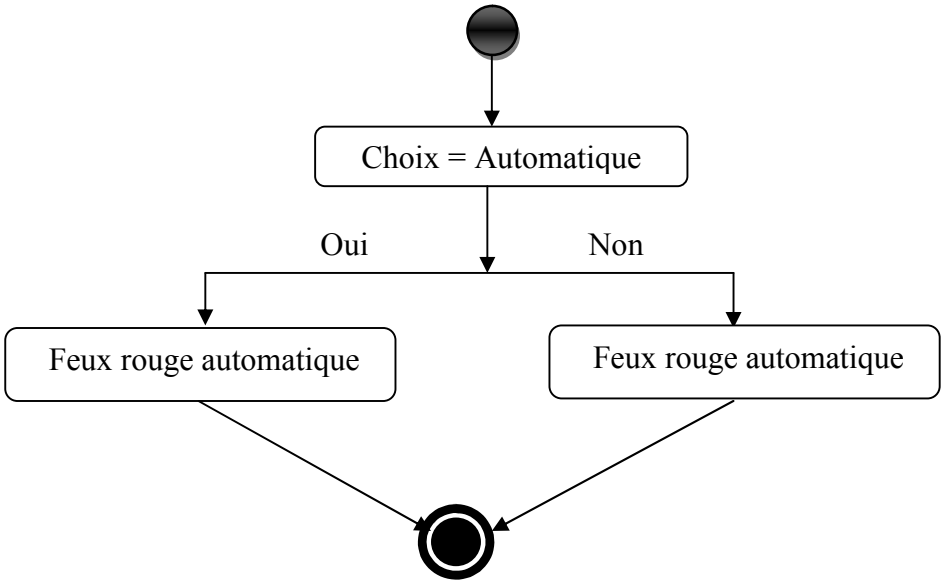

Figure 22: diagramme d'activité de feu de signal.

Feux rouge automatique : selon un temps donné va être le changement des feux

Feux rouge autonomique : chaque feu capté des voitures dans son vision et le changement de couleur lié à la comparaison des capteurs avec un seuil.

3. Passage piétons

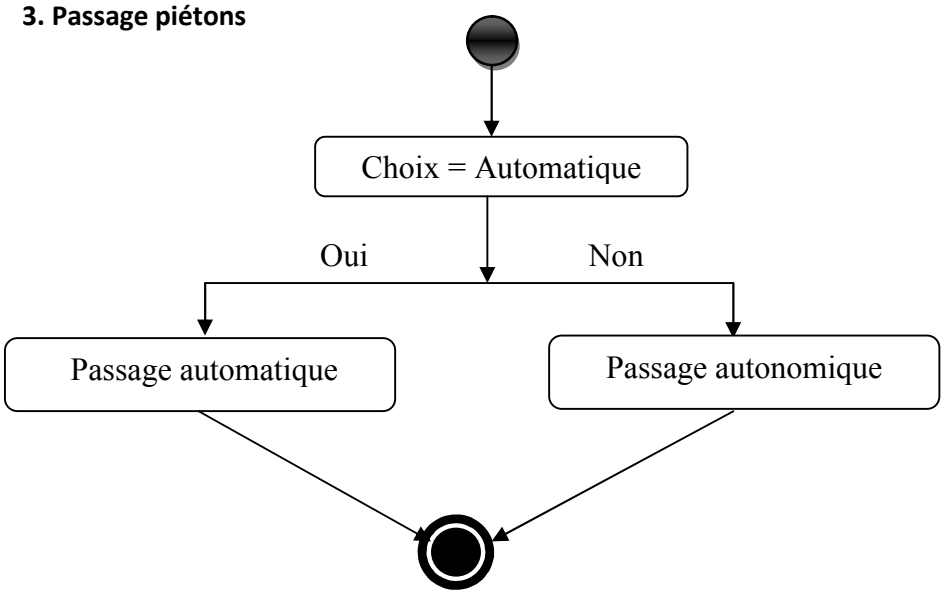

Figure 23: diagramme d'activité de passage piéton.

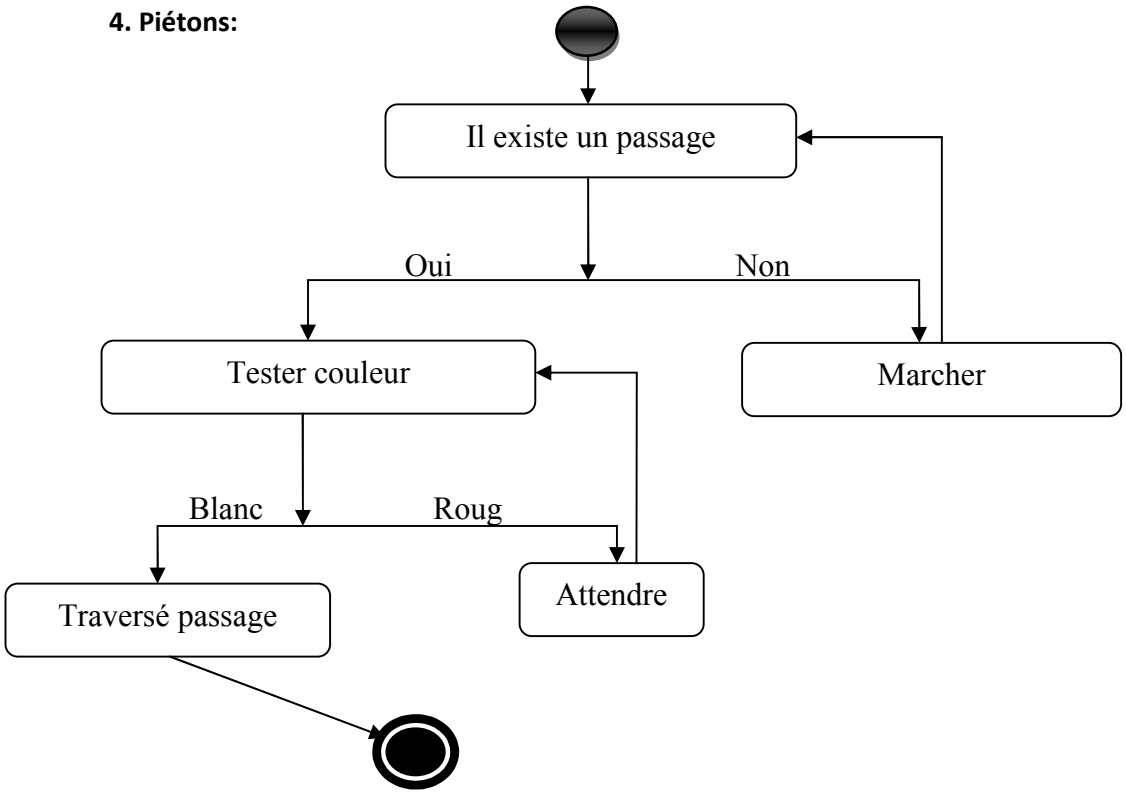

Figure 24: diagramme d'activité des piétons.

3.4. Démarche :

Suivant les recommandations d'Amblard **[**F.Amblard, N.Marileau**]**, nous entamons, réintégrons ce démarche dans notre méthodologie suivie, afin de facilité le passage entre la conception UML aboutis et le programme multiagent NetLogo inspiré:

1. Identification des variables globales :

Temps : une variable qui nous permet de maitriser le fonctionnement automatique des feux de signal.

2. Identification des propriétés des objets à modéliser:

En termes de structure d'environnement, notre univers est présenté comme une grille régulière, donc notre environnement est borné (non ouvert)

Pour les voitures et piétons : Définit comme des composant totalement mobile, continuellement supposés comme des Turtles Netlogo.

Pour les feux de signal et passages piétons : définit comme des composants immobiles. On peut les définit comme des patches Netlogo seulement.

Pour l'environnement : définit comme un ensemble de patches, Modélisé par défaut dans la plateforme Netlogo, comme un ensemble de patchs, toutefois sans aucune utilité de définir un breed comme celle des quatre autres agents (feux, voitures, passage_ps, piétons).

On différencier entre les quatre types Turtles employés par les primitives breed et la primitive Shape comme suit:

- **breed [feux feu]** avec set Shape "circle".
- **breed [voitures voiture]** avec set Shape " voiture-top".
- **breed [passage_ps passage_p]** avec set Shape "passage_p".
- **breed [piétons piéton]** avec set Shape person.

3-Organiser les entités entre elle :

C'est l'étape clé de ce passage inspiré, respectent les deux contraintes un seul cran /degré d'héritage possible à partir de Turtle, et pas de comportements attribués à une liste d'agent (AgentSet), ce sera à faire dans le programme principal.

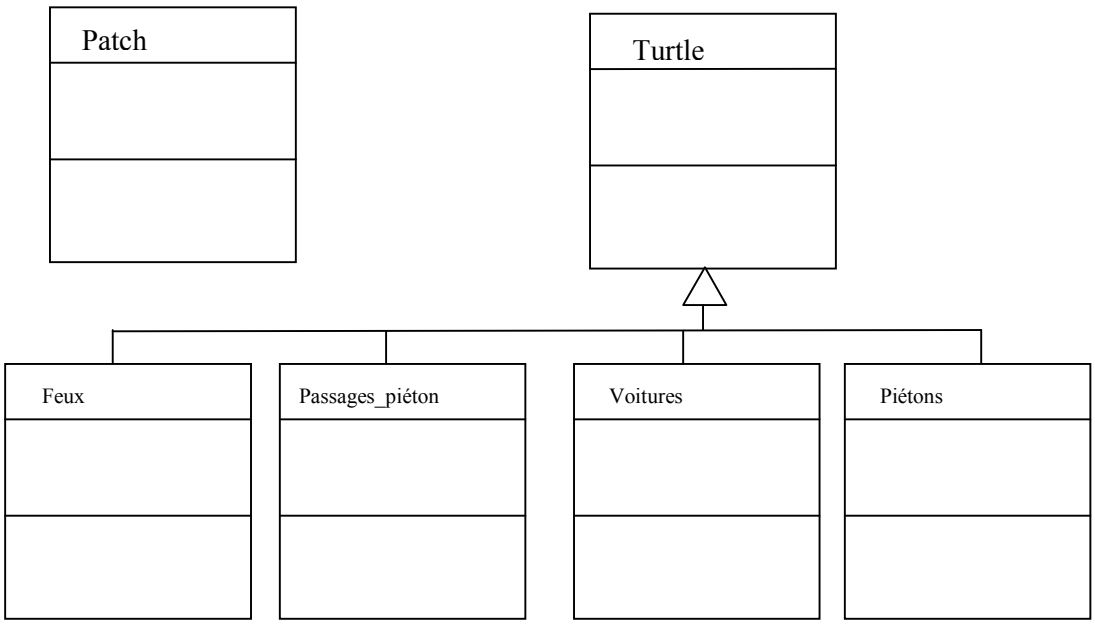

Figure 25 : Diagramme de passage UML/NetLogo

3. 5. Implémentation :

Nous commençons par le principal programme présenter dans la **Figure 26**

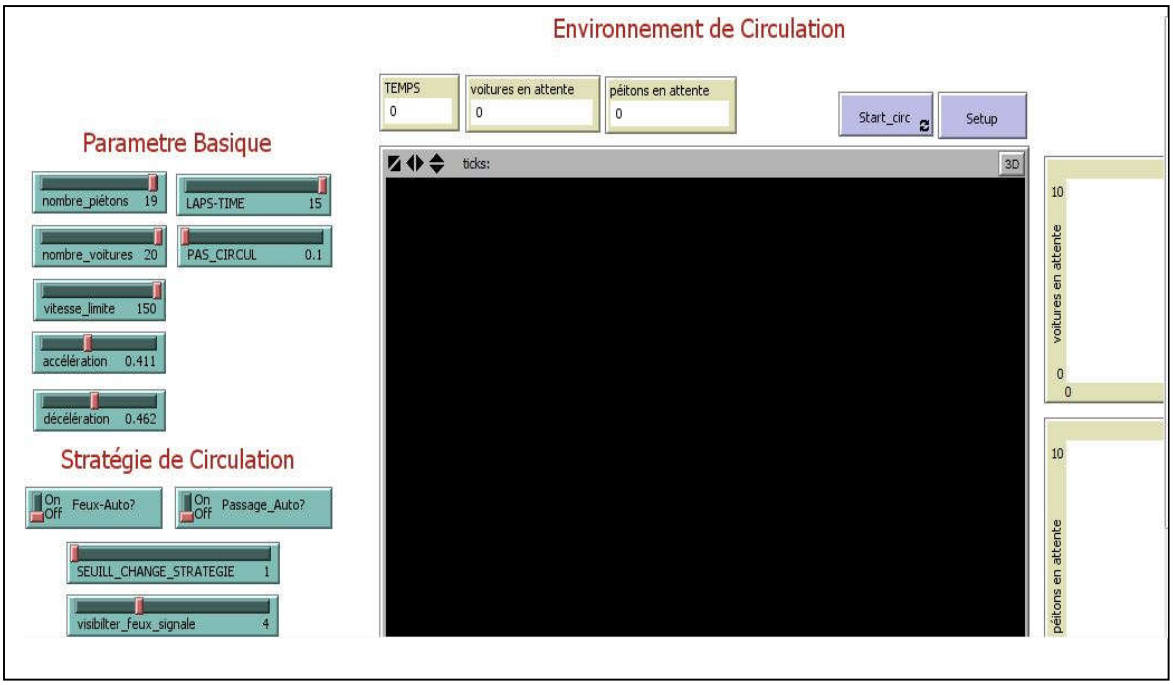

Figure 26: Un Canevas pour notre programme principal.

Concernent le sujet 'Setup', reproduit l'étape d'initialisation des différents paramètres de notre modèle.et tracer l'environnement de simulation suivant la **Tableau 4.**Fixé les routes, les feux de signal, les passages piétons, les quartiers et placer les voitures et les piétons de façon aléatoire.

En passent au deuxième corps de notre implémentation qu'il s'agit de la partie 'Start_Circ' ou de démarrer, elle à été représenté par le programme NetLogo suivante :

```
Every PAS_CIRCUL [
 TIME
  IFELSE Feux-Auto?        [START_Automatique]  [START_AUTONOMIQUE]
  IFELSE Passage_Auto?     [Changé_coleur_passage_automatique]
                           [Changé_coleur_passage_autonomique]
]
```

Les procédures des nœuds mobiles (voiture et piéton) et immobile (feux et passage piéton) :

	Procédure d'emplacement
Feux	config_feux_signal
Passage piéton	config_passage_péitons
Voiture	configurer_voitures
Piéton	configurer_pietons
Environnement	CONFIGURER_ROUTES config_zone_urbaine

Tableau 5: tableau regroupe les procédures dans notre simulateur "SimCirc".

3.5.1. Présentation Du Simulateur "SimCirc" :

Un simulateur concerne le trafic d'un ensemble de véhicule/ piétons (virtuelle), avec un but primordial, autonomiser les feux de circulation et les passages piétons.

3.5.2. Le Tableau de Bord Global :

Stratégie de Circulation:

- Feux-Auto? : pour spécifier le fonctionnement des feux de signal (automatique /autonomique).
- Passage_Auto? : pour spécifier le fonctionnement des passages piétons (automatique /autonomique).
- SEUILL_CHANGE_STRATEGIE : pour le changement autonomique.
- visibilter_feux_signale : pour la visibilité des feux.

Indicateur basique :

- nombre_piétons : pour indiquer le nombre des piétons.
- nombre_voitures : pour indiquer le nombre des voitures.
- vitesse_limite : pour imposer les contraintes de la vitesse des voitures.
- accélération : pour imposer les contraintes de la vitesse des voitures.
- décélération : pour imposer les contraintes de la vitesse des voitures.
- PAS_CIRCUL : pour notre propre unité de temps.
- LAPS-TIME : pour crée notre propre horloge de simulation

Simuler :

- Setup : pour initialiser le simulateur
- Start_Circ : pour lancer la simulation

Les moniteurs :

- voitures en attente : pour afficher le nombre des voitures qui attendent dans l'intersection.
- piétons en attente : pour afficher le nombre des piétons qui attendent dans le passage piétons.
- TEMPS : pour le temps général de simulation.

Les plots :

- Courbe d'attente des voitures dans l'intersection : pour tracer le graphe d'attente des voitures dans l'intersection.
- Courbe d'attente des piétons dans les passages piétons : pour tracer le graphe d'attente des piétons dans les passages.

On passe maintenant aux comportements qui nous intéressé :

- **feux de signal** : procédure control-traffic-autonomique présente au suivants :

```
TO  CONTROL-TRAFFIC-AUTONOMIQUE
ask feux [set cFHG count voitures-on patches with [role_passage_p = "routeFHG" ]  ;feu  horizontal gauche
          set cFHD count voitures-on patches with [role_passage_p = "routeFHD" ]  ; horizontal droit
         set cFVA count voitures-on patches with [role_passage_p = "routeFVA" ] ; feu vertical descendant
          set cFVD count voitures-on patches with [role_passage_p = "routeFVD" ] ; feu vertical ascendant]
ask  one-of feux with [color = red]
[  ifelse Position_Feux = "feuVD" or Position_Feux = "feuxVA" [
if ( CFVA >=  SEUILL_CHANGE_STRATEGIE OR CFVD >= SEUILL_CHANGE_STRATEGIE  ) [ Changé_couleur]  ]
[if Position_Feux ="feuxHG" or Position_Feux ="feuxHD"[
 if ( CFHD >= SEUILL_CHANGE_STRATEGIE OR CFHG >= SEUILL_CHANGE_STRATEGIE ) [ Changé_couleur ] ]
]] END
```

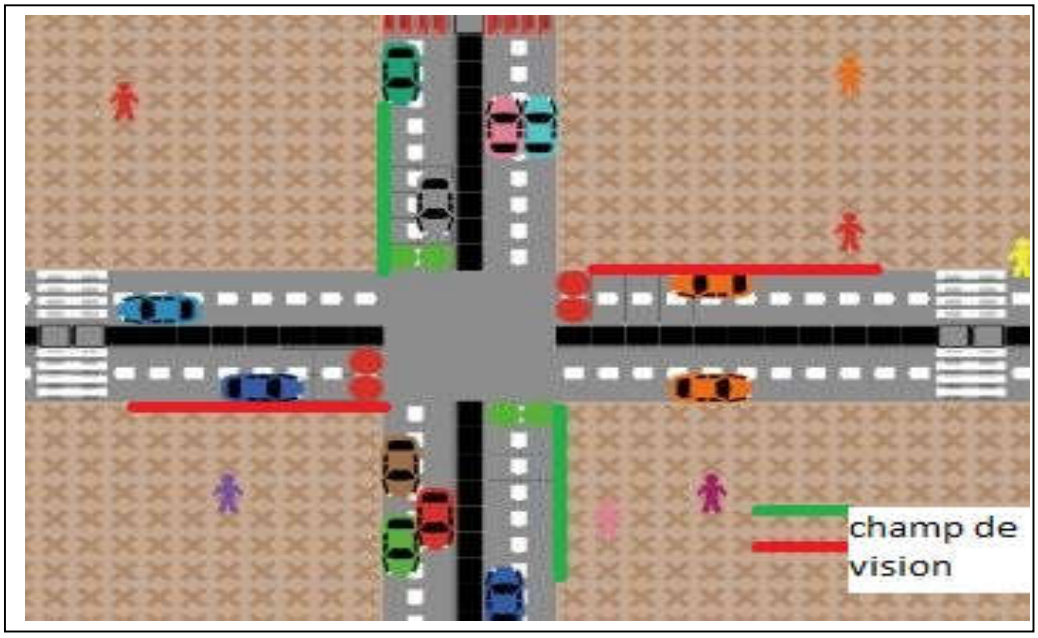

Figure 27: Illustration de visibilité des feux de signal.

❖ **passage piétons** : procédure de changé de couleur d'une façon autonomique.

```
to changé_coleur_passage_autonomique
capteur_voiture_piéton_passage_p
ask passage_ps[
 if capteur_piétons_passage < capteur_voitures_passage and not any? pietons-on passage_ps
      [set shape "passage_p_interdit"]
 if capteur_piétons_passage = capteur_voitures_passage = 0
      [set shape "passage_p"]
 if capteur_piétons_passage > capteur_voitures_passage and not any? voitures-on passage_ps
      [set shape "passage_p"]
  if capteur_piétons_passage = capteur_voitures_passage > 0
      [ifelse random 2 = 1 [set shape "passage_p"]
        [set shape "passage_p_interdit"]
      ]  ] end
```

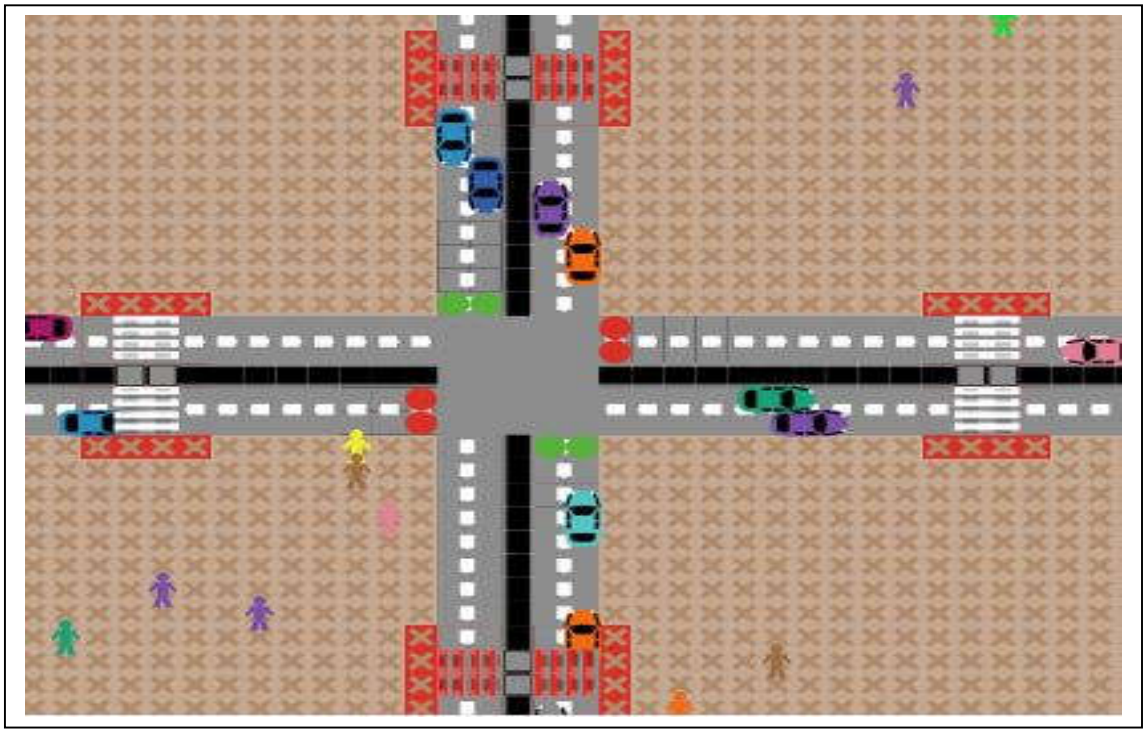

Figure 28: Illustration des champs de visibilité du passage piéton.

3.5.3. Résumé des valeurs :

		Par défaut	**Intersection**		**Passage piétons**		**Terrain (route/ quartier)**
			Feu rouge	Feu vert	rouge	blanc	
Voiture	**vitesse**	v-max - random 20	0	vitesse	vitesse	0	vitesse
	v-max	vitesse_limite - 15 + random 10	v-max		v-max		v-max
	tourné?	peut-être	peut-être/ ouiD		peut-être		peut-être
	att-inter?	False	True	false	false		false
	stop?	peut-être	peut-être		non	oui	peut-être
Piéton	**vitesse_p**	random 7+5	/		0	vitesse_p	/
	temps_mouv	Random 100	temps_mouv++		0		temps_mouv ++
	Pass	0	0		0	1/2/3	0
	att?	False	False		true	false	false

Tableau 6: affectation des valeurs selon leur état (voitures/ piétons).

3.6. Exécution & Analyse:

Notre simulation informatique regroupant la conception basé principalement sur des digrammes UML, à travers le langage Netlogo, par la suite nous voudrons réaliser une Analyse/ comparaison entre quatre stratégies de circulation détaillé par la suite, nous voulons tester l'influence de chaque stratégie sur le nombre de voitures, piétons bloquées respectivement dans l'intersection et le passage piéton par rapport au de temps d'attente :

A-FEUX Automatique **ET** passage automatique (FULL AUTOMATIQUE).

B-FEUX Automatique **ET** passage autonomique (SEMI AUTO+),

C-FEUX Autonomique **ET** passage automatique (SEMI AUTO++) ou

D-FEUX Autonomique **ET** passage autonomique. (FULL AUTONOMIQUE)

❖ **Simulation FULL AUTOMATIQUE**:

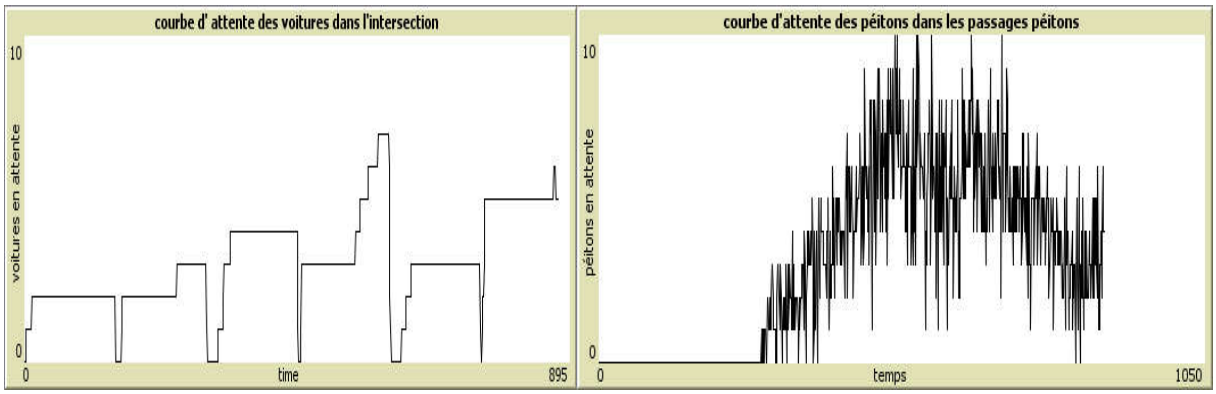

Figure 29 : courbes stratégie 01.

Analyse : Le **premier courbe**, démontre qu'avec des feux Les voitures attentent un long de temps égal LAPS-TIME. Le deuxième courbe démontre que 'avec des passages piétons automatique les piétons prend un temps assez moyen dans l'attente quant à compléter la circulation

❖ **Simulation SEMI AUTO+:**

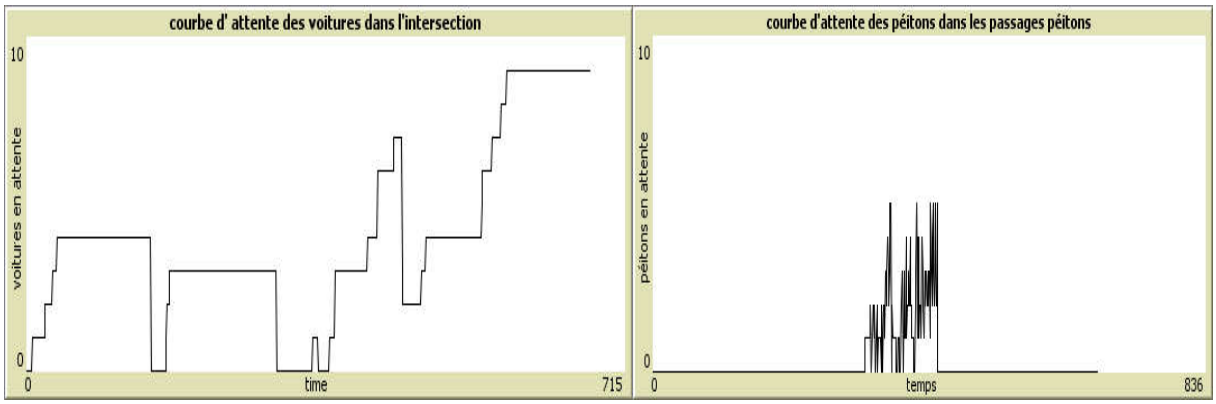

Figure 30: courbes stratégie 02.

Analyse: Le **premier courbe**, démontre qu'avec des feux automatique comme la première stratégie, Les voitures attentent un long de temps égal LAPS-TIME. Le deuxième courbe

démontre que 'avec des passages piétons autonomique les piétons prend un temps assez faible dans l'attente quant à compléter la circulation.

❖ **SIMULATION SEMI AUTO ++:**

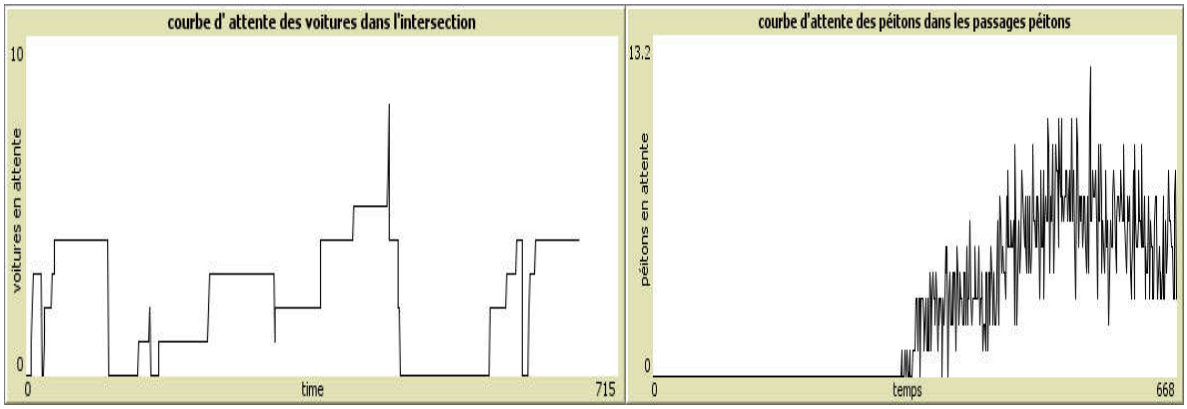

Figure 31: courbes stratégie 03.

Analyse: Le **premier courbe**, démontre qu'avec des feux automatique Les voitures prend un temps assez long égal à LAPS-TIME. Le deuxième courbe démontre que 'avec des passages piétons autonomique les piétons prend un temps assez faible dans l'attente quant à compléter la circulation.

❖ **SIMULATION FULL AUTONOMIQUE :**

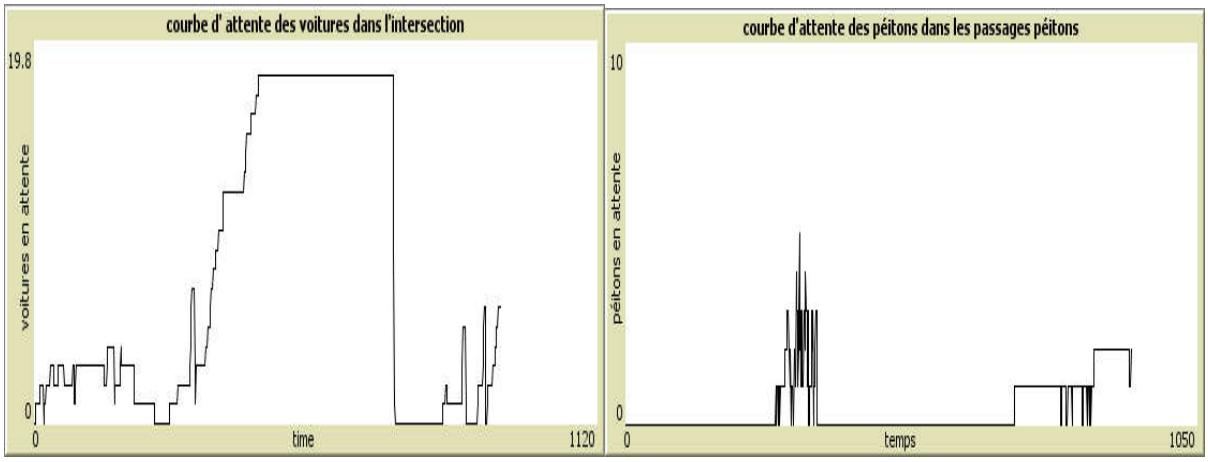

Figure 32:courbes stratégie 04.

Analyse: Le **premier courbe**, démontre qu'avec des feux autonomique Les voitures attentent un long de temps assez faible. le deuxième courbe démontre que 'avec des passages piétons autonomique les piétons prend un temps assez faible aussi bien dans l'attente quant à compléter la circulation.

Récapitulatif de l'étude : La performance de la circulation en termes de temps d'attente dépend relativement de l'adoption associative d'une politique autonomique pour les feux d'intersection, et de passage des piétons.

Conclusion :

Ce qu'il faut retenir de ce dernier chapitre, on a profité d'UML comme langage d'annotation et des plateformes de simulation SMA (NetLogo). Après la présentation de notre simulateur, nous avons vu et proposé des stratégies pour optimiser le temps d'attente des voitures dans une intersection, et adapté les piétons avec les voitures dans les passages piétons.

L'approche adoptée à notre avis, peut il critiqués simplement au niveau de la crédibilité, cette approche adopté, peuvent il, plus au moins restreinte, le cadre proposé doit être testé dans un large environnement, et surtout sur d'autres plateformes de simulation SMA.

Conclusion Générale & et Perspectives

La modélisation d'un système ou d'un phénomène, quel qu'il soit, permet de formaliser un problème afin de le rendre fiable ou de l'optimiser pour pouvoir l'appliquer dans un cas réel. Nous avons décrit les travaux menés dans ce mémoire, dans le cadre de la modélisation par le paradigme agent avec un contexte multiagent, pour maitriser la complexité d'une circulation routière avec la prise en compte des piétons et les passages piétons.

L'objectif global de ce travail été de montrer l'efficacité quant à la fusion de l'approche multi-agents indiquée par le langage/plateforme Netlogo, avec l'approche orientée objet reflétée par le langage 'UML', afin modéliser la circulation routière d'un un ensemble de nœuds mobiles (véhicules), ces nœuds sont incarnés dans notre application, par des turtles-Netlogo, L'application développée concerne le trafic d'un ensemble de véhicule/ piétons (virtuelle), avec un but primordial, autonomiser les feux de circulation et les passages piétons. Cette circulation est modélisé, programmé, afin d'illustrer par des feux de circulation et des passages piétons autonomes et non pas automatique.

Une grande partie de ce mémoire s'est concentrée sur :

-La rédaction d'un état de l'art sur les approches développant la circulation routière,

-La description, l'explication, et la distinction entre les approches de trafique routière, particulièrement dans un univers multiagent.

-La présentation et l'apprentissage d'une plateforme multiagent prodige nommé Netlogo.

-Le développement d'un simulateur pragmatique nommé **"SimCirc"** dédié à la simulation d'une circulation routière avec la prise en compte des piétons et les passages piétons. Ce développement est issu d'un processus inspiré (UML/Netlogo) et bien détaillé dans ce mémoire, concernant une tâche de circulation générique sélectionnée.

-La réalisation d'un ensemble d'expérimentation préliminaires sur le simulateur **"SimCirc"** que nous avons développé, pour montrer l'efficacité de l'intégration de l'autonomie aux feux de circulation et aux passages piétons. Quoique ces expérimentations soient assez restreintes (limités). Cette limitation provient principalement de la taille de l'environnement, le nombre de véhicules-virtuels, d'intersection et des passages et la plateforme de simulation employés pendant les simulations effectuées, ces limitations influence négativement et considérablement la crédibilité de notre modélisation basée essentiellement sur des agents Netlogo (Turtles). i.e le cadre proposé doit être testé avec un grand nombre d'intersection et

dans un large environnement, et sur d'autre plateforme de simulation SMA, comme Swarm, Repast.Etc.

A court terme, les travaux les plus importants à réaliser concernent l'extensibilité du Simulateur **"SimCirc"**, en termes de gestion de conflits (accidents) dans les intersections, Nous envisageons d'étendre ce modèle, par l'adoption d'une stratégie de gestion des accidents. Toujours à court terme, et comme notre premier modèle dans ce mémoire, est basé sur l'agent turtle (véhicule) comme un axe de modélisation, nous envisageons comme perspective, et pour la même circulation, de proposer d'autres modèles avec un autre axe conceptuel, c'est-à-dire, un deuxième modèle dirigé par les piétons, un troisième modèle dirigé par les passages piétons, un quatrième modèle dirigé par les feux de circulation, de telle sorte que notre modèle d'aujourd'hui (à base de véhicule), et les trois autres modèles ambitionné (à base de feux de circulation, à base piéton, et a base de passage piétons) peuvent être comparés en termes de résultat de simulation (une comparaison entre ces trois modèles).

A moyen terme, et comme ce modèle reste incomplet et à parfaire. En perspective, il s'agira entres autres d'inclure des styles de coopération entre les intersections (nouveaux agents qui regroupe les feux de circulation d'une seul intersection), dans le but à moyen terme d'établir des études sur les approches de coopération basées sur l'apprentissage (qui à été brièvement expliquer dans l'état de l'art), afin d'appliquer ces approches de coopération à nos feux de circulation-virtuels (turtle).
A plus long terme, nous pensons qu'il serait particulièrement intéressant d'étudier des tâches de circulation difficilement maitrisables en termes d'imbrication de comportement, comme celle d'encombrement de voiture a cause d'une panne au milieu de route, comment maitriser ces comportements (dépassé une voiture en panne)?

Le travail présenté dans ce mémoire est une première pierre concernant une démarche flexible consacrée à la maitrise d'une circulation routière entre un ensemble d'agents mobiles (véhicules et piétons) ainsi qu'un ensemble d'agents non mobiles (feux de circulations et passages piétons), Il peut se décliner en différentes extensions, que la durée d'une mémoire n'a pas permis d'aborder, et qui constituent autant de pistes de recherche à explorer.

Bibliographie

CHAMPION et Al, 1999. Traffic generation with the SCANeR© II simulator : towards a multi-agent architecture, pp 3.

P.EHLERT, et L.ROTHKRANTZ, 2001. A Réactif Driving Agent pour la simulation du trafic Microscopique, ESM '01: Actes du 15 ème européenne Simulation Multiconference, SCS Maison d'édition pp: 943-949

J.C. Tanner 1953 : A problem of interference between two queues. Biometrika, 40(1/2):pp: 58-69.

M. Baykal-Gürsoy, W. Xiao et K. Ozbay, 2009: Modeling trafic flow interrupted by incidents. European Journal of Operational Research, 195(1):127-138.

Gaillard, M. Soulignac, C. Dinont et P. Mathieu, 2011 : Deterministic kinodynamic planning with hardware demonstrations. In Intelligent Robots and Systems (IROS),2011 IEEE/RSJ International Conference on, pages 3519_3525. IEEE.

Gechter, J.M. Contet, O. Lamotte, S. Galland et A. Koukam : Virtual intelligent vehicle urban simulator : Application to vehicle platoon evaluation. Simulation Modelling Practice and Theory (SIMPAT), 24:103_114, mai 2012.

Mammar : Systèmes de Transport Intelligents, modélisation, information et contrôle .Hermes science, Lavoisier, 2007.

Greenshields : A study of trafic capacity. In Proceedings of the Highway Research Board, volume 14, pages 448_477, 1935.

M. J. Lighthill et G. B. Whitham : On kinematic waves. 2. a theory of trafic flow on long crowded roads. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, 229(1178):317_345, 1955.

Richards : Shock waves on the highway. Operations research, 4(1):42_51, 1956.

Kosonen et M. Pursula : A simulation tool for trafic signal control planning. In Third International Conference on Road Trafic Control., pages 72 _76. IEEE, may 1990.

Hidas : Modelling lane changing and merging in microscopic trafic simulation. Tramsportation Research Part C : Emerging Technologies, 10(5-6):351_371, 2002.

Sheu : Microscopic trafic behaviour modelling and simulation for lane-blocking arterial incidents. Transportmetrica A : Transport Science, 9(4):335_357, 2013.

Chevallier et L. Leclercq : Do microscopic merging models reproduce the observed priority sharing ratio in congestion ? Transportation Research Part C : Emerging Technologies, 17(3):328_336, 2009.

Bazzan : Opportunities for multiagent systems and multiagent reinforcement learning in trafic control. Autonomous Agents and Multi-Agent Systems, 18(3):342_375, 2009.

Bazzan : A distributed approach for coordination of trafic signal agents. Autonomous Agents and Multi-Agent Systems, 10(1):131_164, 2005.

Fok, M. Hanna, S. Gee, T.C. Au, P. Stone, C. Julien et S. Vishwanath : A platform for evaluating autonomous intersection management policies. In Third International Conference on Cyber-Physical Systems (ICCPS), pages 87_96. IEEE, 2012.

Robertson : TRANSYT : A Trafic Network Study Tool. RRL report. Road Research Laboratory, 1969.

Guebert et G. Sparks : Timing plan sensitivity to changes in platoon dispersion settings. In Trafic Control Methods. Proceedings of the fifth ng foundation conference,Santa barbara, California, 1990.

Lowrie : The sydney coordinated adaptive tra_c system-principles, methodology, algorithms. In International Conference on Road Trafic Signalling, 1982, London,
United Kingdom, 1982.

Henry, J.L. Farges et J. Tuffal : The prodyn real time trafic algorithm. In 4th IFAC/IFIP/IFORS Conference On Control, 1984.

Rochner, H. Prothmann, J. Branke, C. Müller-Schloer et H. Schmeck : An organic architecture for trafic light controllers. In Proceedings of the Informatik 2006 Informatik fürMenschen I Jahrestagung (1), pages 120_127, 2006.

Dresner et P. Stone : Multiagent tra_c management : An improved intersection control mechanism. In Proceedings of the Fourth International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2005.

Kosonen : Multi-agent fuzzy signal control based on real-time simulation. Transportation Research Part C : Emerging Technologies, 11(5):389_403, 2003.

Sánchez et J.L. Aguirre : Trafic light control through agent-based coordination. In Artificial Intelligence and Applications, pages 163_168, 2007.

R.Naumann et R. Rasche : Intersection collision avoidance by means of decentralized security and communication management of autonomous vehicles. Univ.-GH, SFB 376,1997.

R. Naumann, R. Rasche, J. Tacken et C. Tahedi : Validation and simulation of a decentralized intersection collision avoidance algorithm. In Intelligent Transportation System (ITSC'97), IEEE, pages 818_823, 1997.

Brockfeld, R. Barlovic, A. Schadschneider et M. Schreckenberg : Optimizing trafic lights in a cellular automaton model for city trafic. Physical Review E, 64(5):056132, 2001.

Steingrover, R. Schouten, S. Peelen, E. Nijhuis et B. Bakker : Reinforcement learning of trafic light controllers adapting to trafic congestion. In Proceedings of the Belgium-Netherlands Artificial Intelligence Conference (BNAIC), pages 216_223. Citeseer, 2005.

France et A.A. Ghorbani : A multiagent system for optimizing urban trafic. In Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on, pages 411_414. IEEE, 2003.

K.Dresner et P. Stone : Multiagent trafic management : A reservation-based intersection control mechanism. In Proceedings of the Third International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2004.

K. Dresner et P. Stone : Multiagent trafic management : An improved intersection control mechanism. In Proceedings of the Fourth International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2005.

Gershenson : Self-organizing trafic lights. complex Systems, 16(1):29_53, 2004.

Wiering : Multi-agent reinforcement learning for trafic light control. In Proceedings of the International Conference on Machine Learning, pages 1151_1158, 2000.

Balan et S. Luke : History-based trafic control. In Proceedings of the fifth International Conference on Autonomous Agents and Multiagent Systems, pages 616_621. ACM, 2006.

Mohring, K. Nokel et G. Wunsch : A model and fast optimization method for signal coordination in a network. In Control in Transportation Systems, volume 11, pages 73_78, 2006.

Köhler, R.H. Möhring et G. Wünsch : Minimizing total delay in fixed-time controlled trafic networks. In Operations Research Proceedings 2004, pages 192_199. Springer Berlin Heidelberg, 2005.

Bhouri, F. Balbo, S. Pinson et M. Tlig : Collaborative agents for modeling trafic regulation systems. In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on, volume 2, pages 7 _13, aug. 2011.

Hounsell et B. Shrestha : A new approach for cooperative bus priority at trafic signals. Intelligent Transportation Systems, 13(1):6 -14, 2012.

Ferreira et P.K. Khosla : Multi agent collaboration using distributed value functions. In Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE, pages 404_409. IEEE, 2000.

Nunes et E. Oliveira : Learning from multiple sources. In Proceedings of the Third International Conference on Autonomous Agents and Multiagent Systems-Volume 3 , pages 1106_1113. IEEE Computer Society, 2004.

S.Richter, D. Aberdeen et J. Yu : Natural actor-critic for road tra_c optimization. In B. Schölkopf, J. Platt et T. Hofmann, éditeurs : Advances in Neural Information Processing Systems 19, Cambridge, MA, 2007. MIT Press.

B.C. Da Silva, E.W. Basso, A.L.C. Bazzan et P.M. Engel : Dealing with onstationary environments using context detection. In Proceedings of the 23rd international conference on Machine learning, pages 217_224. ACM, 2006.

E. Camponogara et W. Kraus Jr : Distributed Learning agents in urban trafic control. In Progress in Artificial Intelligence, pages 324_335. Springer, 2003.

R. Mandiau, A. Champion, J.M. Auberlet, S. Espié et C. Kolski : Behaviour based on decision matrices for a coordination between agents in a urban trafic simulation. Applied Intelligence, 28(2):121_138, 2008.

P. B. Hunt, D. I. Robertson, R. D. Bretherton et R. I. Winton : SCOOT – a trafic responsive method of coordinating signals. Rapport technique, TRRL, 1981.

D. De Oliveira et A.L.C. Bazzan : Trafic lights control with adaptive group formation based on swarm intelligence. In Ant Colony Optimization and Swarm Intelligence, pages 520_521. Springer, 2006.

D. De Oliveira et A.L.C. Bazzan : Swarm intelligence applied to trafic lights group formation. Anais do VI Encontro Nacional de Inteligência Artificial (ENIA), pages 1003_1112, 2007.

A. Doniec, S. Espié, R. Mandiau et S. Piechowiak : Dealing with multi-agent coordination by anticipation : Application to the trafic simulation at junctions. EUMAS, 5:478_479, 2005.

L. Bull, J. Sha'Aban, A. Tomlinson, J.D. Addison et B.G. Heydecker : Towards distributed adaptive control for road trafic junction signals using learning classifier systems. In Applications of Learning Classifier Systems, pages 276_299. Springer Berlin Heidelberg, 2004.

M. Bouzid : Contribution à la modélisation de l'interaction Agent/Environnement, modélisation stochastique et simulation parallèle. Thèse de doctorat de l'université Henri Poincaré, Nancy1 (Spécialité informatique). 2001

P. Blangi : Etat de l'art sur les plates formes et les langages Multi-Agents Appliqués aux écosystèmes. Master de recherche: modélisation et simulation des systèmes complexes 2004/2005.

Selma AZAIEZ Approche dirigée par les modèles pour le développement de systèmes multi-agents**,** thèse de doctorat 2009, université de Savoire.

Fishwick, computer simulation: growth through extension. IEEE potential, February /mars (1996), 24-27.

David.Sheeren, "introduction à la modélisation UML", Atelier 1, MAPS : modélisation multiagents appliquée aux phénomènes spatialisés, la vieille Perrotine, Saint-pierre d'Orléon 21-26 JUIN 2009.

F.Amblard, N.Marileau «Modélisation multi-agents appliquée aux phénomènes spatialisés » Atelier 2 : Passage du modèle conceptuel UML à Netlogo La Vieille Perrotine, Saint-Pierre d'Oléron 21-26 juin 2009.

Wilensky, U. & Reisman, K. (2006). Thinking like a Wolf, a Sheep or a Firefly: Learning Biology through Constructing &Testing Computational Theories and Embodied Modeling Approach.Cognition&Instruction,24(2),pp.171-209.
http://ccl.northwestern.edu/papers/wolfsheep.pdf.

Ilias Sakellariou, Petros Kefalas, and Ioanna Stamatopoulou "Enhancing NetLogo to Simulate BDI Communicating Agents" Department of Applied Informatics, University of Macedonia, Thessaloniki,[2]Department of Computer Science, CITY College, Thessaloniki, Greece J. Darzentas et al. (Eds.): SETN 2008, LNAI 5138, pp. 263–275, 2008. Copyright Springer-Verlag Berlin Heidelberg 2008.

Vidal, J.M., Buhler, P., Guardia, H.: The past and future of multiagent systems. In: Proceedings of 1st AAMAS Workshop on Teaching Multi-AgentSystems (2004)

Wooldridge, M.: An Introduction to MultiAgent Systems. J. Wiley & Sons, Chichester (2002).

Gauthier picard," Méthodologie de développement de systèmes multi-agents adaptatifs et conception de logiciels à fonctionnalité émergente ", thèse de doctorat, Université Paul Sabatier de Toulouse III, 2004.

Lounis Adouane MiRoCo un simulateur pour systèmes multi-robots à forte Dynamique d'interaction LASMEA, UMR CNRS 6602 24 Avenue des landais, 63177 Aubière Cedex, France Lounis.Adouane@lasmea.univ-bpclermont.fr, 2005.

Mohamed Rédha BAHRI. Une approche intégrée Mobile-UML/Réseaux de Petri pour l'Analyse Des systèmes distribués à base d'agents mobiles, Thèse de doctorat 'université mentouri Constantine.

J. P. Briot, Y. Demazeau, Principes et architecture des systèmes multiagents 11 octobre 2001 .Paris**.**

B. Chaib-draa, I. Jarras et B. Moulin, "Systèmes multiagents : Principes généraux et applications", Article à paraître dans : J. P. Briot et Y. Demazeau « Agent et systèmes multiagents » chez Hermès en 2001.

Franklin, Stan and Graesser, "is it an Agent, or just a program? A Taxonomy For Autonomous Agents", Proccedings of the Third International Workshop on Agent theories, Architecture, And Languages Springer-Verlag 1997.

Brooks, R. (1991). Intelligence without représentation. Artificial Intelligence, 47: 139–159.

Labidi, S., Lejouad, W. (1993) ‘ De l’intelligence artificielle distribuée aux systèmes multi-agents’ INRIA N° 2004.

Brahim Chaib-draa, Imed Jarras et Bernard Moulin : Systèmes multiagents : Principes généraux et applications. In J. P. Briot et Y. Demazeau, éditeurs : Agent et systèmes multiagents. Hermès, 2001.