La simulation multi-agent des déplacements

Sommaire

2.1 Introduction

Dans le chapitre précédent, nous avons exprimé notre préférence pour la simulation comme moyen d'évaluation de l'impact de l'information des voyageurs. Dans ce cadre, il existe plu-

sieurs simulateurs de mobilité des voyageurs dans la littérature. Cependant, dans le cadre d'une évaluation d'impact de l'information voyageurs par simulation, il faut que les comportements des voyageurs puissent intégrer les informations dont on désire évaluer l'impact. Ainsi, un processus de prise de décision individuel doit accompagner chaque voyageur simulé. Ce processus définit par la suite les réactions des voyageurs suite aux informations perçues. Ces réactions représentent l'objet de l'analyse nécessaire afin d'évaluer l'impact de ces informations. Par conséquent, les voyageurs simulés doivent être munis d'un degré d'autonomie et de perception de leur environnement pendant leurs déplacements. Pour cette raison, ce chapitre traitera exclusivement des approches pour la simulation des déplacements, qui représentent les voyageurs d'une manière individuelle sous la forme d'agents : les simulations multi-agents de déplacements.

Les systèmes multi-agents (SMA) mettent en réseau un semble dynamiques d'agents, et offrent en effet un outil pertinent dans la gestion de systèmes complexes distribués en général. Les SMA sont efficaces pour la modélisation et la simulation de systèmes nécessitant l'interaction de plusieurs entités. En effet, la simulation de certaines situations (évacuations, attentats, etc.) est délicate, coûteuse et difficile à réaliser. Ainsi, la simulation multi-agent est une solution pertinente pour ce genre de problématiques. Un SMA permet d'intégrer un nombre important d'individus dans une simulation tout en assurant une gestion et un suivi individuel de chacun d'eux. Grâce aux interactions, un SMA permet de modéliser différents modèles sociaux où plusieurs formes d'échanges et d'interactions sont simulées. Ces capacités permettent de simuler, par exemple, une population, des piétons, des voyageurs dans un réseau de transport, des spectateurs dans une salle de spectacle ou un stade, des visiteurs d'un salon ou d'un centre commercial, etc. La simulation des déplacements permet aux architectes de bien modéliser les plans de déplacements avec l'avantage de pouvoir tester et calibrer plusieurs scénarios. Elle offre aux gestionnaires de ses réseaux plusieurs avantages et possibilités en leur permettant d'étudier et d'analyser d'une manière microscopique leurs méthodes d'affectation ou leurs systèmes d'information voyageurs essentiellement en situation de dysfonctionnement.

Ce chapitre est structuré comme suit. La section 2.2 définit la notion d'agent et la section 2.3 définit les systèmes multi-agents. La section 2.4 présente trois simulateurs multi-agents de déplacement de la littérature. Concluant que ces simulateurs ne sont pas pertinents pour nos objectifs, nous présentons dans la section 2.5 les plateformes de simulation multi-agents généralistes. Nous en choisissons une qui sera développée dans le chapitre suivant. La section 2.6 est notre conclusion.

2.2 La notion d'agent

2.2.1 Définition

Dans la littérature, plusieurs définitions sont utilisées pour le terme « agent ». Ces définitions varient selon le contexte et l'utilisation. Nous choisissons deux définitions qui semblent les plus appropriées à notre contexte. La première est celle de Wooldridge qui dans son livre [Wooldridge2002] définit un agent en tant que « système informatique situé dans un environnement et capable d'ap-

pliquer des actions autonomes dans le but de satisfaire ses buts ». La deuxième est une définition plus détaillée que Ferber utilise dans son livre [Ferber1995] et dont il illustre 6 caractéristiques :

Définition AGENT LOGICIEL

on appelle agent purement communicant (ou agent logiciel) une entité informatique qui :

1. se trouve dans un système informatique ouvert (ensemble d'applications, de réseaux et de systèmes hétérogènes),
2. est mue par un ensemble d'objectifs propres,
3. possède des ressources propres,
4. ne dispose que d'une représentation partielle des autres agents,
5. possède des compétences (services) qu'elle peut offrir aux autres agents,
6. a un comportement tendant à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose et en fonction de ses représentations et des communications qu'elle reçoit.

Il ressort de ces définitions des propriétés clés, comme l'autonomie, la réactivité, la perception et la communication. D'autres propriétés peuvent être attribuées aux agents, comme la rationalité.

2.2.2 Caractéristiques

Les caractéristiques les plus communément admises pour décrire un agent sont l'autonomie, la réactivité et la sociabilité.

— L'autonomie : [Russell and Norvig2003] spécifient qu'un agent autonome est un agent qui agit sans l'intervention des humains ou des autres agents, et qui de ce fait possède un contrôle sur ces actions et sur ses états internes. Un agent est ainsi une entité informatique qui possède la liberté de répondre aux appels à ses compétences et de prendre des initiatives sans avoir recours à des interventions externes.

— La réactivité d'un agent est sa capacité de réagir en cas de changements de l'état de son environnement. Les choix d'actions se feront en fonction de sa perception des changements. Cette perception guidera dans la suite ses choix et sa gestion d'actions.

— La sociabilité est le degré d'interaction qu'un agent peut avoir avec les autres agents. L'interaction peut se faire pour un besoin propre à l'agent ou pour un besoin de coordination entre les agents du même système.

2.2.3 Types d'agents

Plusieurs travaux classent les agents selon leurs caractéristiques. [Weiss1999] distingue deux types d'agents en se basant sur leurs capacités de raisonnement : les agents réactifs qui sont fondés sur un mécanisme de stimuli-réactions, et les agents cognitifs qui ont la particularité de posséder un module de délibération qui intègre un processus décisionnel complexe.

— Les agents réactifs sont des agents qui perçoivent et réagissent aux changements de leur environnement. Leurs réactions se font selon des règles prédéfinies d'une façon rapide similaires à des réflexes. Ces caractéristiques rendent ce type d'agents adapté à la simulation de sociétés (humaines ou animales) ou d'un ensemble de machines dans le domaine robotique. Ce type de simulation est souvent utilisé pour analyser l'impact des comportements individuels des agents sur le cheminement du groupe.
— Les agents cognitifs se caractérisent par la possession d'un module de délibération qui intègre un processus décisionnel complexe et qui définit les actions de l'agent. Selon [Mandiau et al.02a], leurs actions sont souvent le résultat d'une planification et d'un raisonnement rationnel qui se base sur un ensemble de connaissances sur leur environnement, sur les autres agents et sur eux mêmes. Parmi les architectures d'agents cognitifs les plus connues est l'architecture BDI (*Belief, Desire, Intention*) [Rao and Georgeff1992]. Cette architecture intègre les notions de croyance, de désirs et d'intention dans le processus décisionnel et s'avère bien appropriée, par exemple, à l'environnement de la réalité virtuelle. Dans [Shendarkar *et al.*2008], l'architecture BDI est étendue pour proposer une simulation d'évacuation de foule. Les caractéristiques de l'agent cognitif le rend bien approprié pour ce type de simulations. En effet, l'influence d'éléments émotionnels sur son processus décisionnel favorise son utilisation pour étudier le comportement humain comme dans le cas de [Lyell and Becker2005] qui étudie le comportement de foules dans un contexte de crise à l'aide d'agents cognitifs.

D'autres critères peuvent être utilisés pour classer les agents. Un agent peut être statique ou mobile en fonction de sa capacité à parcourir les réseaux informatiques tel que Internet ou un réseau local afin d'exécuter ses tâches. Aussi, les degrés d'adaptation, d'autonomie et de coopération sont des critères importants qui permettent de classer les agents ainsi :

— Agent collaboratifs : ces agents favorisent la coopération à l'adaptation afin de résoudre une tâche globale qui nécessite l'association de plusieurs compétences individuelles. Ces agents sont généralement utilisés pour résoudre des problèmes distribués.
— Agents interfaces : ces agents jouent le rôle d'assistants personnels aux utilisateurs en collaborant et communiquant avec eux afin d'orienter l'exécution des tâches. Ces agents favorisent l'autonomie et l'apprentissage afin d'assister l'utilisateur et ils sont généralement utilisés dans le domaine des interfaces homme/machine.
— Agents d'information : ces agents se caractérisent par leurs autonomie en matière de recherche d'informations lorsque leurs tâches le nécessitent. Ils sont capables de collecter et de gérer des informations provenant de sources différentes. Selon [Adam2000], ces agents sont capables de lancer leurs activités de recherche et de collecte d'informations d'une manière autonome ; en fonction d'un raisonnement interne, d'un manque d'informations ou d'une nouvelle disponibilité d'informations ou de sources d'information.

2.3 Les systèmes multi-agents

2.3.1 Définition

[Ferber1995] donne la définition suivante d'un système multi-agent :

Définition SYSTÈME MULTI-AGENT

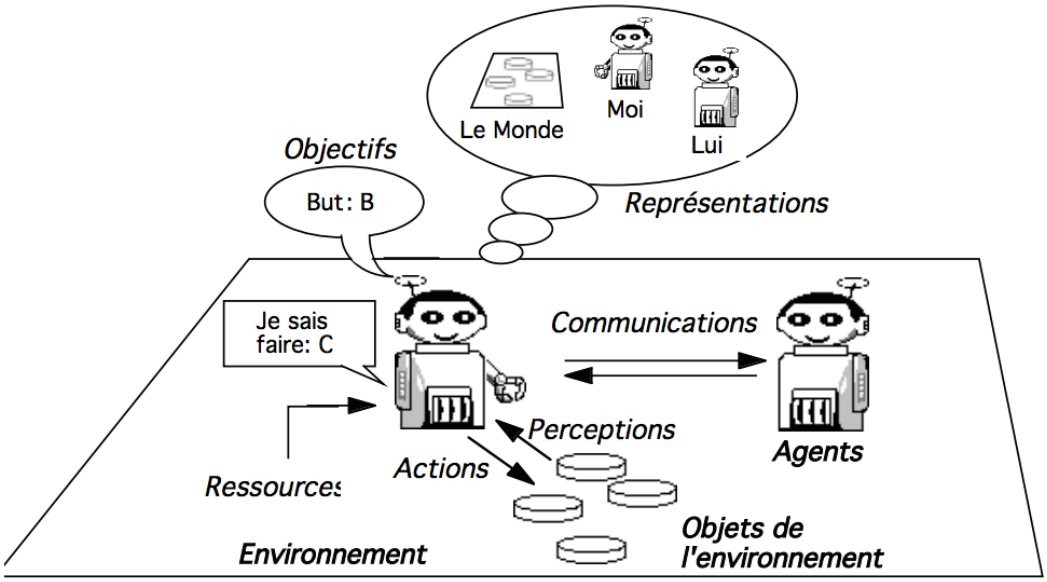

FIGURE 2.1 – Illustration de la notion de système multi-agent selon Ferber

On appelle système multi-agent, un système composé des éléments suivants :

1. Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.
2. Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.
3. Un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système.
4. Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.
5. Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O.
6. Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.

La figure 2.1 donne une illustration de la notion de système multi-agent. [Fayech2003] définit les SMA ainsi : « Un système multi-agent est un réseau d'agents (solveurs) faiblement couplés qui coopèrent ensemble pour résoudre des problèmes qui dépassent les capacités ou les connaissances individuelles de chaque agent. Les agents sont autonomes et peuvent être de natures hétérogènes. »

Ainsi, les agents d'un SMA, n'ayant pas une visibilité globale sur leur environnement, ne peuvent avoir qu'un champ d'actions limité sur l'ensemble des objets de cet environnement. De

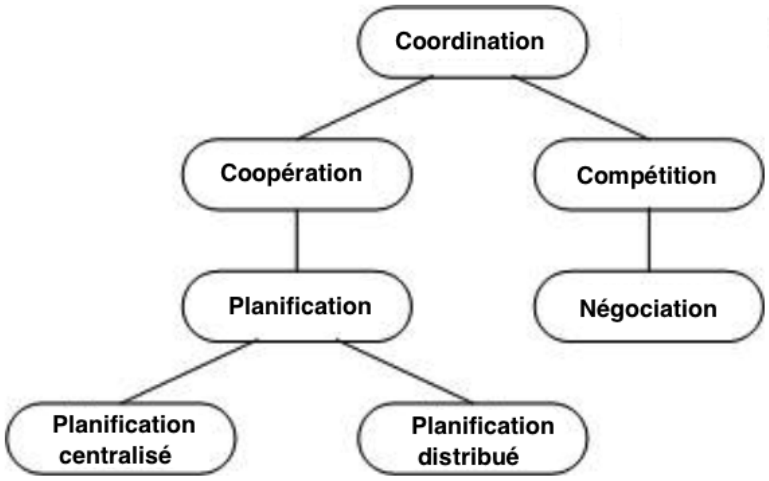

FIGURE 2.2 – Les formes d'interactions entre agents

ce fait pour résoudre un problème global, ces agents sont amenés à coopérer et à communiquer pour échanger des informations et pour mieux coordonner leurs actions individuelles et locales. Par conséquent, un système multi-agent doit avoir une forme d'organisation qui définit les règles de ces échanges.

2.3.2 Organisation des SMA

La notion d'organisation des agents dans un SMA, s'impose pour définir les rôles des agents et les mécanismes de coordination et de communication entre eux. [Weiss1999] propose le schéma de la figure 2.2 afin d'illustrer les différents types d'interaction qui peuvent avoir lieu entre les agents d'un SMA.

[Weiss1999] définit la coordination comme « la propriété d'un système composé d'au moins deux agents, exécutant des actions dans un environnement partagé ». Ainsi, la notion d'environnement partagé implique le besoin de coordination entre les différents agents du système. Deux types de coordinations existent :

— la négociation pour les agents antagonistes (ayant des buts et des objectifs contradictoires) afin d'éviter les situations de conflits.
— la coopération pour les agents non antagonistes afin d'améliorer l'efficacité et l'utilité de chacun d'eux.

La négociation peut être considérée comme un moyen de résolution de conflits entre plusieurs agents. Selon [Fayech2003], la négociation est une méthode de coordination qui permet à plusieurs agents d'atteindre un accord mutuel pour entreprendre une action donnée d'une certaine manière . Elle induit, par cette communication, des relaxations de buts initiaux, des concessions mutuelles, des mensonges ou des menaces. Ensuite, d'après [Weiss1999], la coopération est la coordination parmi des agents non antagonistes, qui cherchent à se satisfaire mutuellement sans se gêner. Ce type de coordination basé sur l'échange d'informations peut avoir la forme d'une

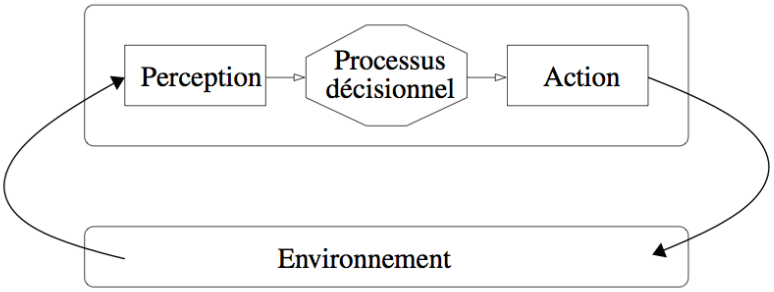

FIGURE 2.3 – Le lien entre l'environnement et l'agent

collaboration où la répartition des tâches entre les agents est étudiée.

2.3.3 Interaction

Pour se coordonner entre eux, pour négocier, et pour coopérer les agents ont besoin d'interagir entre eux. Dans les SMA, deux types d'interactions existent : la communication directe et la communication indirecte.

— La communication directe utilise un langage de communication pour pouvoir échanger des messages entre les agents. Les messages contiennent des informations que les agents désirent échanger afin d'aboutir aux besoins de leurs tâches. L'émetteur utilise le langage pour coder l'information à envoyer qui sera ensuite décodée par l'agent récepteur via le même langage. D'après [Roze2003], deux langages sont couramment utilisés : KQML (*Knowledge Query and Manipulation Language*) et ACL (*Agent Communication Language*).

— La communication indirecte utilise l'environnement comme support d'interaction. Pour transmettre une information, un agent émetteur effectue une modification sur l'environnement qui sera perçue puis interprétée par l'agent récepteur. Le meilleur exemple de ce type de communication est celui des fourmis qui communiquent entre eux le chemin vers la source de nourriture à travers le dépôt de phéromones.

2.3.4 Environnement

En se référant à la définition des systèmes multi-agents de Ferber présentée précédemment, nous constatons que l'environnement représente une composante majeure et essentielle dans ces systèmes. En effet, tous les états et les actions d'un agent dépendent de l'état de l'environnement. La figure 2.3 montre le rôle de l'environnement dans le processus décisionnel de l'agent. Aussi, il est la base des échanges et interactions entre les différents agents d'un SMA, ce qui représente le moteur du paradigme multi-agent.

L'environnement dans les systèmes multi-agents possède plusieurs propriétés, ce qui offre une richesse au processus de modélisation de ces derniers afin de répondre aux besoins de l'utilisateur. Voici les différentes propriétés selon la suggestion de [Russell and Norvig2003] de présenter chaque propriété en opposant ses deux extrêmes.

— Accessible ou inaccessible : un environnement accessible est un environnement dans lequel un agent peut obtenir une information complète, exacte et mise à jour de son état. La majorité des environnements modérément complexes (incluant, par exemple, le monde réel et Internet) sont inaccessibles. Plus l'environnement est accessible, plus simple est la construction d'un agent afin d'y opérer.
— Déterministe ou indéterministe : un environnement déterministe est un environnement dans lequel n'importe quelle action a un seul effet garanti : il n'y a aucune incertitude quant à l'état qui résultera de l'application d'une action. Les environnements non déterministes posent plus de problèmes pour le concepteur du système.
— Épisodique ou non épisodique : dans un environnement épisodique, la performance d'un agent est dépendante d'un nombre d'épisodes discrets, sans lien entre les performances d'un agent dans des scénarios différents. Un exemple d'environnement épisodique pourrait être un système de tri de courrier électronique. Les environnements épisodiques sont plus simples du point de vue du développeur agent parce que l'agent peut décider quelle action appliquer en se basant seulement sur l'épisode courant ; il n'a pas besoin d'interaction entre l'épisode courant et les épisodes à venir.
— Statique ou dynamique : un environnement statique est un environnement qui ne risque de changer d'état que lors de l'application d'actions par l'agent. Un environnement dynamique est un environnement qui a d'autres processus agissant dessus et qui risque de subir des changements au détriment de la volonté de l'agent. Le monde réel est un environnement hautement dynamique.
— Discret ou continu : un environnement est discret s'il comporte un nombre fixe et fini d'actions et de perceptions possibles. Russel et Norvig donnent un jeu d'échec comme exemple d'environnement discret et la conduite d'un taxi comme exemple d'environnement continu.

Les classes les plus complexes d'environnements concernent ceux qui sont inaccessibles, indéterministes, non-épisodiques, dynamiques et continus. Dans ce qui suit, nous nous intéressons à la simulation multi-agent, et aux plateformes de simulation existantes.

Dans la section qui suit, nous nous intéressons aux applications qui se fondent sur le paradigme multi-agent présenté dans cette section, pour la simulation des déplacements. Cette étude a pour objectif d'éventuellement choisir parmi ces applications, celle sur laquelle fonder notre proposition.

2.4 Simulateurs multi-agents des déplacements

On distingue principalement trois types de modèles de simulation de déplacements. Les modèles de déplacements macroscopiques considèrent une représentation des déplacement comme des flux, dont les comportements sont décrits par des équations. Les variables définies dans le modèle sont les débits, concentrations et vitesses sur les différents tronçons du réseau modélisé. Les modèles de déplacements mésoscopiques utilisent généralement les mêmes outils, mais représentent les déplacements sous forme de pelotons ou des groupes de voyageurs ou de véhicules (e.g. de [De Palma and Marchal2002]). Il s'agit d'une représentation intermédiaire entre la représentation microscopique et la représentation macroscopique. Enfin, les modèles microscopiques étudient des systèmes composés d'éléments individuels que sont les voyageurs ou les

véhicules. A cet égard, la simulation multi-agent des déplacements peut être vue comme un cas particulier des modèles microscopiques. Le recours à des modélisations agrégées est encouragé entre autres par à la nature agrégée des données disponibles aux concepteurs, mais aussi par les limites des outils de modélisation et d'exécution [Michael *et al.*2009].

Cependant, la dernière décennie a été marquée par l'évolution continue des outils informatiques qui permettent de répondre au grand besoin de fournir des simulations microscopiques des réseaux de transport. Les données de calibration des modèles microscopiques sont également de plus en plus disponibles et de meilleure qualité. Les modèles de simulation microscopique et multi-agent ont plusieurs avantages. Face à la complexité continue de systèmes de transport, la simulation des déplacements à une échelle microscopique est pertinente pour les étudier finement et les analyser, pour en proposer une meilleure gestion. Parmi les avancées les plus marquantes, la capacité des équipements multi-processeurs à élaborer des calculs parallèles ainsi que les architectures à mémoires partagées permettant un accès rapide à la mémoire physique pour un nombre important de processeurs [Michel2015]. Cette simulation microscopique des modèles de transport offre ainsi aux gestionnaires un large choix de résultats qui peuvent aller des simples statistiques générales jusqu'à des informations individuelles de chaque voyageur simulé. Aussi, elle permet une modélisation explicite des processus individuels de prise de décision assurant une dynamique réelle à la simulation à travers l'hétérogénéité des comportements des voyageurs simulés.

Il existe plusieurs simulateurs de mobilité des voyageurs de ce type dans la littérature. Par exemple, Transims [Nagel and M.Rickert2001] simule des voyages multimodaux et évalue les impacts des changements de politique de mobilité dans les caractéristiques du trafic tandis que Miro [Chipeaux *et al.*2011] reproduit les dynamiques urbaines d'une ville française et propose un prototype de simulation multi-agent capable de tester la planification de scénarios en précisant les comportements des individus. AgentPolis [Jakob *et al.*2012] est également une plateforme multi-agent pour le transport multimodal et MATSim [Maciejewski and Nagel2012] est une plateforme bien connue de micro-simulation de mobilité. Sumo [Behrisch *et al.*2011] et Vissim [Fellendorf and Vortisch2010] sont des simulateurs microscopiques largement utilisés et principalement axés sur la simulation de trafic. Il existe également des simulateurs tels que Archisim [Doniec *et al.*2008] qui décrivent de manière très précise le comportement des conducteurs à l'échelle microscopique. Notre travail étant centré sur la simulation multi-agent des déplacements des voyageurs sur un réseau de transport en commun, nous choisissons d'étudier dans la section suivante les architectures des modèles de simulation à base d'agents pertinentes pour ce type de simulation : AgentPolis, Transims et MATSim.

2.4.1 AgentPolis

AgentPolis est une récente plateforme à base d'agents pour la modélisation de systèmes de transport multimodal. Les entités individuelles de ces systèmes sont représentées comme des agents autonomes ayant des modules de contrôle asynchrone et continus, ils interagissent librement avec leur environnement et entre eux. L'approche à base d'agents permet de concevoir des scénarios où les agents ajustent leurs plans à tout moment durant la journée en fonction de leur perception de l'environnement et/ou des coordinations avec les autres agents. AgentPolis est composé de trois composantes centrales : le coeur de la simulation, la librairie du domaine de transport et l'outil de simulation.

1. Le coeur de la simulation : la partie centrale d'AgentPolis contient une plateforme de simulation à événements discrets basée sur l'outil multi-agent ALITE [8]. Cette plateforme consiste en un moteur de traitement assurant la construction des modèles de larges systèmes d'événements discrets à base d'agents [Jakob *et al.*2012].
2. La librairie du domaine de transport : cette librairie fournit un ensemble de composantes spécifiques qui assurent une création rapide de modèles de transport et de mobilité. Elle est constituée des classes et modules suivantes :
   — Composants du réseau de transport : classes qui permettent de construire l'environnement de transport (routes, réseau ferroviaire et métro, carrefours, arrêt, correspondance).
   — Véhicules : classes représentant les véhicules (voitures, bus, trains) et leurs propriétés.
   — Actions et capteurs de transport : classes assurant les liens et accès entre les agents et leur environnement de transport comme l'embarquement et la descente des véhicules, le déplacement sur une route ou la détection d'arrivée des trains.
   — Activités de transport et cycles de vie : structures réactives de contrôle qui peuvent être composées pour créer un comportement d'agent désiré (e.g. voyager via le transport en commun ou conduire une voiture entre deux points du réseau).
   — Planificateur d'itinéraires et de voyage : module qui fournit un agent des capacités de raisonnement, en particulier planifier son itinéraire dans un réseau de transport multimodal.
3. Outils de simulation : cet ensemble d'outils permet la conception et l'exécution des modèles sur AgentPolis.
   — Interfaces de données et filtres : ils permettent de travailler directement avec des données en formats standards tels que OpenStreetMap pour les données de cartes et GTFS pour le transport public (réseau et tableau de marche).
   — Outil de génération de population : cet outil permet de générer un grand nombre d'agents avec une distribution réaliste d'attributs démographiques (âge, genre, salaires, domicile, voiture, etc.) basé sur des données réelles de recensements.
   — Outil de configuration, gestion et déploiement d'expériences : cet outil assure la mise en place des scénarios d'expérimentations et leur exécution automatique sur les ressources disponibles.
   — Outil de visualisation et de rapport : cet outil permet de visualiser le déroulement de la simulation ainsi que les résultats. Cette visualisation inclut l'aspect temporel et géospatial et l'agrégation suite aux multiples simulations d'une manière interactive basée sur GoogleEarth.

Ces outils permettent ensemble une construction rapide d'un modèle basé sur des données réelles. Ils permettent de définir une variété de scénarios de simulation, d'exécuter ces scénarios avec la possibilité d'exécution parallèle, d'analyse des résultats et d'affichage d'une manière interactive [Jakob *et al.*2012].

Exemples d'application :

AgentPolis a été utilisé pour développer un modèle de simulation de système de contrôle de titres de transport dans un réseau de transport en commun. Ce modèle permet d'évaluer

8. http ://agents.fel.cvut.cz/projects/alite

l'efficacité des stratégies de contrôle mises en place par des experts humains ou par des outils informatiques. Ce modèle contient deux types d'agents : un agent voyageur et un agent contrôleur. En plus du processus de déplacement, l'agent voyageur dispose d'une logique d'achat de ticket qui détermine si un ticket doit être acheté ou non pour un voyage particulier. L'agent contrôleur contrôle les voyageurs dans certains trains et sur certaines stations suivant une stratégie d'inspection précise. La performance de chaque stratégie peut être testée sur des voyageurs avec différents niveaux de rationalité et de perception de leur environnement. Plusieurs mesures de performance peuvent être collectées par ce modèle comme la somme des amendes collectées, les revenus perdues et les coûts des opérations de contrôle. Le modèle a rapidement été développé pour le système de métro de Los Angeles et a simulé environ 400 milles trajets sur cinq lignes par jour.

Cependant, AgentPolis ne fournit pour le moment pas de code source de la plateforme, ni d'exécutable. Il s'agit en fait d'un projet de recherche parallèle au nôtre. Par ailleurs, l'impact de l'information de voyageurs n'a pas été traité dans cette plateforme, et les auteurs ne montrent pas la possibilité d'intégrer des couches d'informations permettant le guidage de ces agents pendant une situation particulière.

2.4.2 Transims

Transims est parmi les outils de modélisation de réseaux de transports les plus avancés. Transims est basé sur un système de simulation multi-agent capable de simuler instantanément les déplacements des voyageurs et des véhicules sur un réseau de transport d'une large zone urbaine. Il est composé de cinq modules : le synthétiseur de population, le générateur d'activités, le planificateur d'itinéraires, le microsimulateur et le module de rétroaction. Le dernier module fournit des outils d'analyse des résultats de simulations. Il s'agit essentiellement du système d'affichage qui intègre les informations géospatiales (GIS) et des statistiques ainsi qu'un estimateur d'énergie consommée et d'émission de gaz lors des simulations.

1. Le synthétiseur de population : ce module est utilisé pour créer une population artificielle de la zone urbaine étudiée. Il combine des données agrégées en provenance des tableaux de recensement démographique et des données désagrégées des recensements des échantillons de microdonnées à usage public afin de créer une base de population synthétique où chaque individu est associé à un domicile. Les statistiques agrégées de cette population reproduisent celles de la vraie population. Les attributs de la population synthétique contiennent des informations individuelles tels que l'âge, le genre, le travail/étude, voiture, etc.
2. Le générateur d'activités : ce deuxième module identifie la liste des activités journalières de chaque individu synthétique dans chaque domicile. Le nombre des voyages que chaque individu programme pendant sa journée est déterminé en comptant le nombre de changement de lieux dans sa liste d'activités journalières. Ainsi, la liste d'activité définit la séquence quotidienne de voyages désirée par chaque voyageur de la population. Les données d'entrées de ce module sont une liste d'informations élaborée par le synthétiseur de population.
3. Le planificateur d'itinéraires : l'itinéraire à emprunter pour chaque voyage généré lors du module 2 est calculé par ce module. Cet itinéraire inclut les liens à emprunter, le mode

de transport, les changements de modes, l'emplacement des parkings et d'autres informations. Ce module utilise une version modifiée de l'algorithme Dijkstra pour calculer le plus court chemin. Ce module prend comme entrées les informations des voyageurs et leurs activités issues du module 2 et les informations du réseau de transport (le graphe représentant le réseau, les modes de transport de chaque arc, les temps de voyage de chaque arc, etc.). Pour chaque itinéraire calculé, le temps de départ, les arcs à emprunter avec le mode correspondant et le temps total de voyage sont affichés.

4. Le microsimulateur : le module microsimulateur exécute les plans de voyages individuels de chaque voyageur générés par le module 3. Les déplacements des voyageurs et leurs interactions avec le réseau de transport sont simulés à une échelle microscopique. La première étape de la simulation consiste à lire les données du réseau telles que les arcs, les noeuds, les voies et leurs connexions, les parkings, etc. La deuxième étape intègre les véhicules et la localisation et plans des voyageurs puis les positionne sur les réseau. La troisième étape gère les déplacements des véhicules et des voyageurs selon leurs plans de voyage en utilisant l'approche des automates cellulaires. Cette approche consiste à diviser les liens du réseau en cellules, une cellule permet de contenir un seul véhicule. Le mouvement d'un véhicule est représenté par un saut d'une cellule à une autre. Différentes vitesses de véhicules sont représentés par différentes distances de saut (e.g. un saut de 5 cellules par pas de temps correspond à une vitesse de 135 km/h). Les véhicules se déplacent selon les règles suivantes :
   — Accélération linéaire jusqu'à la vitesse maximale s'il n'y a pas de véhicule devant.
   — Si véhicule devant, la vitesse sera ajustée en fonction de la distance séparant les deux véhicules.
   — Quelquefois la vitesse est aléatoirement plus lente que ce qui résulte des deux règles précédentes.

   Le microsimulateur permet d'utiliser plusieurs processeurs afin de supporter un grand nombre de voyageurs et des réseaux de transport de grande taille.

5. Le système de visualisation : ce module permet à l'utilisateur de visualiser les données d'entrées et de sortie de la simulation avec des outils d'analyses. Ces outils incluent le traçage, les statistiques, la visualisation spatiale et les animations. Les plans de voyage, les véhicules, les signaux, les carrefours ainsi que plusieurs autres données peuvent être affichées. L'interface graphique du système de visualisation permet de manipuler l'affichage en 3D et de filtrer l'affichage pour une requête individuelle. Le système permet aussi d'animer la simulation par l'envoi de clichés instantanés pendant le déroulement de la simulation. Une version commercialisée de ce système est créée par IBM Business Consulting qui permet d'afficher les données de la simulation en 2D et 3D et d'animer l'exécution de la simulation selon les attributs sélectionnés par l'utilisateur.

6. L'estimateur d'émission : ce module fournit une conversion des déplacements de voyageurs en termes de consommation d'énergie et émission d'oxyde d'azote, hydrocarbures, monoxyde de carbone, dioxyde de carbone et de matières particulières. Pour cela, le module exploite les données sur la composition de la flotte de véhicules issue du module synthétiseur de population, les résultats de tests d'inspection et de maintenance issus des bases de données locales et nationales de la zone étudiée et les schémas du traffic à partir du microsimulateur de Transims. L'inventaire des émissions élaboré par ce modèle a pour

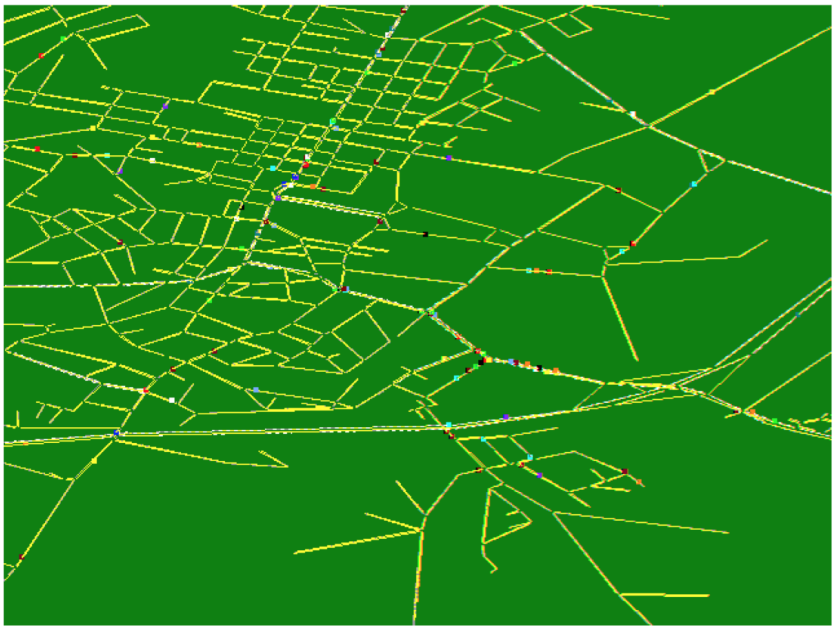

FIGURE 2.4 – Un cliché d'affichage d'une simulation Transims

but de calculer les concentrations des polluants dans une zone urbaine en se basant sur les conditions atmosphériques, le transport des dispersions locales et les réactions chimiques.

Transims est essentiellement utilisé pour étudier les déplacements sur des larges zones urbaines impliquant un grand nombre d'individus comme la microsimulation d'un réseau routier d'une région de la ville de Dallas (États-Unis) impliquant 200 000 voitures. Les résultats des simulations permettent aux autorités locales de calibrer leurs politiques de mobilité et offres de voyages. Cependant, Transims n'intègre pas de processus de prise de décision chez les voyageurs/conducteurs ainsi les réactions des usagers des réseaux de transport face à un événement externe ou à une information ne peuvent pas être modélisés dans Transims [Gu2004].

2.4.3 MATSim

MATSim est un système de modélisation de micro-simulation de réseaux de transport destiné à des applications à grande échelle. Il est développé par une équipe issue du domaine de transport et de l'ingénierie civile qui a exploité le paradigme multi-agent afin de répondre à leurs besoins. Dans MATSim, la demande est modélisée individuellement pour chaque agent. La modélisation concerne toute une journée et est appelée plan. La figure 2.5 montre un exemple de plan d'un agent écrit en xml qui décrit toute son activité journalière. Cette structure reste la même durant toute la modélisation et la simulation de la demande. lors de la phase d'affectation, tout le plan est exécuté et non pas chaque voyage séparément. Afin de fournir les plans de chaque agent, des données concernant l'infrastructure, la population et la demande de la région de la simulation sont créées par l'utilisateur. Ensuite, le système procède à la génération de la demande totale de voyage. Ce processus s'étire sur 4 parties :

— La création de scénario.

```
<plans name="example plans file">
  ...
  <person id="393241" sex="f" age="27" license="yes" car_avail="always"
     employed="yes">
    <travelcard type="regional-abo" />
    <plan>
      <act type="home" link="58" start_time="00:00" dur="07:00" end_time="07:00" />
      <leg mode="car" dept_time="07:00" trav_time="00:25" arr_time="07:25">
        <route>1932 1933 1934 1947</route>
      </leg>
      <act type="work" link="844" start_time="07:25" dur="09:00" end_time="16:25"/>
      <leg mode="car" dept_time="16:25" trav_time="00:14" arr_time="16:39">
        <route>1934 1933</route>
      </leg>
      <act type="home" link="58" start_time="16:39" dur="07:21" end_time="24:00" />
    </plan>
  </person>
  ...
</plans>
```

FIGURE 2.5 – Exemple d'un plan journalier d'un agent dans MATSim

— La modélisation des demandes individuelles initiales.
— L'optimisation itérative de la demande.
— L'analyse.

Le processus d'optimisation itérative de la demande est considéré comme la partie centrale de MATSim [Michael *et al.*2009]. Le processus s'exécute jusqu'à que chaque demande d'agent voyageur soit adéquate avec les restrictions du scénario (offre de voyage, réseau) et l'interaction avec les autres composantes de la simulation.

Généralement, une méthode de relaxation est utilisée afin de trouver un état d'équilibre. Pour le choix d'itinéraire l'équilibre de Wardrop [Wardrop1952] est défini comme état de repos. Cependant, l'optimisation concerne tout le plan journalier (routes, horaires, localisation, séquence d'activités et types d'activités).

Chaque agent essaie d'exécuter son plan avec l'utilité la plus élevée. L'utilité d'un plan journalier dépend de contraintes d'infrastructures (capacités des voies, horaires d'ouvertures des établissements, etc.) et des plans des autres agents dans le système. Cela implique que l'utilité effective d'un plan journalier ne peut être déterminée seulement par l'interaction de tous les agents. C'est ainsi que l'algorithme co-évolutif [Holland1992] joue son rôle. L'algorithme suit les étapes suivantes :

1. Initialiser les plans journaliers pour chaque agent du système.
2. Calculer l'utilité d'exécution des plans individuels de chaque agent.
3. Supprimer les mauvais plans (avec faible utilité).
4. Dupliquer et modifier les plans.
5. Faire de ces plans les plans pertinents pour l'itération suivante. Incrémenter le compteur.
6. Aller à 2.

Nous nous intéressons au processus de re-planification des plans journaliers des agents dans MATSim. Cette partie nous aide à comprendre comment l'activité d'un agent peut être modifiée suite aux changements d'état du réseau. D'après la description de cette phase dans [Michael *et al.*2009], nous constatons que les plans journaliers des agents sont modifiés par l'intermédiaire de modules centraux qui se chargent chacun d'une partie spécifique des plans et agissent indépendam-

ment pendant le processus d'optimisation. De ce fait, les agents n'ont pas de contrôle sur leurs choix et se montrent passifs dans leur processus de déplacement dans un réseau de transport. Parmi les modules qui contrôlent l'optimisation des plans le module « *Router* » qui calcule les chemins à emprunter en se basant sur les données de la simulation précédente. Aussi, le module *Time Allocation Mutator* qui modifie les temps de départ et les durées d'activités des plans journaliers. Les changements se font aléatoirement. Il existe aussi un module qui modifie tous les aspects d'un plan, il est nommé « *planomat* ». Ce module est capable de coordonner les plans d'agents appartenant au même foyer. Plusieurs autres modules peuvent être ajoutés et développés par l'utilisateur de MATSim. L'optimisation des plans des agents continue jusqu'à que le système atteint un état d'équilibre, ce qui se traduit par une moyenne d'utilité constante. Dans le cas de la simulation du réseau national complet de la Suisse, avec 7 millions d'agents, un réseau de 24000 noeuds et 60000 et 22 millions de voyages générés dont 7,1 millions de voyage par transport routier, l'état de relaxation a été atteint après 70 itérations [Meister *et al.*2009].

Il est clair que MATSim permet de simuler des larges réseaux de transport multimodaux en intégrant un nombre très important d'agents voyageurs. Cependant, la centralisation des modules de modifications des plans de voyages ainsi que l'absence d'autonomie chez les agents limitent la flexibilité des modèles de simulation de MATSim à intégrer des couches d'informations voyageurs et de nouveaux types d'agents.

Après cette revue des simulateurs multi-agents des déplacements, nous concluons qu'aucune des propositions, très intéressantes par ailleurs, ne répond à notre problématique : représenter les flux d'information de voyageurs et les intégrer dans les comportements des voyageurs. Il nous faut donc concevoir et développer notre propre outil pour atteindre cet objectif. Néanmoins, nous ne sommes pas totalement démunis face à ce problème et nous ne sommes pas obligé de créer *ex nihilo* une nouvelle plateforme. En effet, il existe des plateformes multi-agents génralistes, *open source*, qui permettent de faciliter grandement le travail de conception et de développement. Ces plateformes font l'objet de la section suivante.

2.5 Plateformes de simulation multi-Agent

Pour concevoir et implémenter un simulateur multi-agent, il est possible de développer une application directement dans un langage de programmation hôte. Cependant, il est souvent plus rapide, plus utile et plus efficace de fonder le simulateur sur une plateforme multi-agent existante. Un critère central pour choisir la plateforme de simulation concerne sa capacité à créer des modèles d'agents avec des résultats reproductibles. Dans les perspectives que nous donnons à notre travail, nous désirons également pouvoir déployer les simulations sur plusieurs serveurs. Enfin, la troisième capacité que nous recherchons pour la plateforme de simulation est sa capacité à concevoir des modèles géo-spaciaux, i.e. sa capacité d'intégrer et de traiter des données géographiques.

Plusieurs autres critères interviennent dans le processus de choix de plateforme. Le premier groupe de critères concerne les facilités de développement offertes par la plateforme comme la taille de la communauté qui l'utilise, la disponibilité de supports d'aide (généralement animés par la communauté), la disponibilité de modèles de démonstration et de documentation technique. Le deuxième groupe concerne des critères en lien avec les fonctionnalités de la modélisation. Il s'agit du nombre d'agents que la plateforme permet de modéliser, les degrés

d'interaction entre les agents, la capacité de représenter différents niveaux d'organisation des agents, les différents modèles d'environnement possibles, la gestion des liens topologiques et spatiales entre agents, les mécanismes de planification, etc. Le critère *open source* est aussi important car il permet aux développeurs d'explorer le code permettant si besoin la modification et l'extension du système [Crooks and Castle2012].

Selon ces critères, nous avons choisis de nous focaliser sur cinq plateformes Swarm [Minar *et al.*1996], Mason [Luke *et al.*2004] , Repast [Tatara and Ozik2009], NetLogo [Wilensky and Evanston1999] et Gama [Patrick *et al.*2012]. Elles remplissent la majorité des critères citées. Elles sont quotidiennement mises à jour, largement utilisées et supportées par une large communauté d'utilisateurs, accompagnées par une variété de modèles de démonstration, *open source* pour certaines et essentiellement capables de développer des modèles géospatiaux à travers l'intégration de fonctionnalités GIS. Ensuite, nous choisissons celle qui s'avère la plus compatible pour modéliser et simuler un système de transport multimodal en tenant en compte sa complexité et son aspect distribué et fortement dynamique.

2.5.1 Swarm

Swarm est une plateforme de simulation et de conception open source qui est conçue spécialement pour le développement de simulations multi-agents de systèmes complexes. Swarm permet aussi de développer des modèles à base d'agents [Patrick *et al.*2012] en proposant un ensemble de concepts standards pour la description et la conception de ces modèles. La plateforme fut développée en Objective-C d'abord, ensuite un ensemble de libraires ont été ajoutées pour permettre la réalisation de simulations de modèles d'agents en langage Java. Swarm intègre des fonctionnalités GIS à travers sa librairie Kerg qui permet le chargement de couches de données géographiques. Cependant, elle ne fournit pas de primitives spatiales, ni ne donne la possibilité de sauvegarder l'environnement géographique résultat [Patrick *et al.*2012].

Au début, Swarm a été utilisée pour essayer d'étudier des phénomènes biologiques [Minar *et al.*1996]. Ensuite, son utilisation a touché plusieurs autres domaines : l'anthropologie, l'écologie, l'économie, la géographie, les sciences politiques. Par exemple, [Railsback and Harvey2002], [Schelhorn *et al.*1999], [Haklay *et al.*2001] et [Batty *et al.*2003] représentent des modèles explicitement spatiaux développés sur Swarm. Les deux premiers constituent des simulations de piétons dans des centres urbains, et le troisième étudie la congestion de foule pendant le carnaval Notting Hill à Londres [Crooks and Castle2012].

2.5.2 Mason

Mason (*Multi Agent Simulation Of Neighbourhood*) est développé par le laboratoire ECLab et le centre de la complexité sociale à l'université George Mason [Luke *et al.*2004]. Mason dispose d'une grande base de documentation technique ainsi qu'une communauté active. Plusieurs publications détaillants son implémentation et son application et des modèles de démonstration sont disponibles afin d'évaluer ou de se familiariser avec cet outil. Mason est basé sur Java et offre plusieurs fonctionnalités comme le traçage dynamique (histogrammes, graphes, etc.) et la modélisation de l'affichage des simulations. Au début, Mason n'intègrait pas de données géospatiales jusqu'à l'addition de l'extension GeoMason. Cette extension permet l'importation et

l'exportation, la manipulation et l'affichage de données géographiques. Cependant, ces capacités restent limitées [Coletti2013]. Les fonctionnalités GIS de Mason sont utilisées dans plusieurs modèles d'agents géospaciaux afin d'étudier : le conflit entre les bergers et les fermiers dans l'est africain [Kennedy *et al.*2010], les bergers nomades en Inde Centrale [Cioffi-Revilla *et al.*2010], le dynamisme résidentiel en Arlington County - Virginia [Hailegiorgis2010] et l'industrie de drogue en Afghanistan [Łatek *et al.*2010].

2.5.3 NetLogo

NetLogo [Wilensky and Evanston1999] est une vesrsion de StarLogo développée à l'université Northwestern initialement pour permettre aux modèles StarLogo d'être développés sur des systèmes d'exploitation Macintoch et ensuite pour le déploiement des modèles sur internet [Crooks and Castle2012]. Au début NetLogo et StarLogo fournissaient uniquement des fonctionnalités pour importer des fichiers images pour définir l'environnement dans lequel les agents sont localisés ce qui facilite le développement de modèles spatiaux. Aujourd'hui, NetLogo permet d'importer des données matricielles et vectorielles (*shapefiles*). Cette fonctionnalité multiplie les possibilités de création de modèles d'agents géospatiaux. Cependant, NetLogo ne fournit pas des opérations d'analyses géographiques avancées.

Deux modèles de démonstration accompagnent l'installation de NetLogo. Le premier est destiné aux données vectorielles et comporte trois ensembles de données : un fichier Points des villes du monde, un fichier Polylignes représentant les rivières, un fichier polygone représentant les pays. Ces données sont importées dans un modèle NetLogo et converties en patches. Pour les les données matricielles, un fichier Raster de l'élévation de la surface est chargé dans un modèle NetLogo pour montrer les possibilités de travail avec des données spatiales. Dans cet exemple, les agents suivent la surface à moindre élévation. Cet exemple peut être utilisé pour modéliser un processus qui s'articule sur l'évaluation de surfaces comme l'évacuation de bâtiments [Crooks *et al.*2008].

NetLogo a été utilisé pour développer des applications dans plusieurs domaines comme la biologie, la physique et les sciences sociales. Aussi, plusieurs modèles spatiaux ont été crées à l'aide de NetLogo comme l'étude de gentrification dans [Torrens and Nara2007], l'étude autour de la demande résidentielle dans [Fontaine and Rounsevell2009], l'émergence des modèles d'occupation des territoires [Graham and Steiner2006] et la réimplémentation artificielle du peuple amérindien Anasazi [Janssen2009].

Le site web de NetLogo offre une large base de documentation et de tutoriels, des modèles de démonstration ainsi que des API qui contiennent des fonctionnalités qui peuvent être extraites. A l'exception des plateformes déjà présentées, NetLogo utilise son propre langage de programmation et son code source n'est pas accessible.

2.5.4 Gama

Développée depuis 2007 [Taillandier2013], Gama est une plateforme de modélisation et de simulation à base d'agents. La plateforme est libre et permet à ses utilisateurs de modéliser des simulations à temps discret. Gama dispose de son propre language de modélisation : le GAML, en plus d'un environnement de développement intégré. Ce language est caractérisé

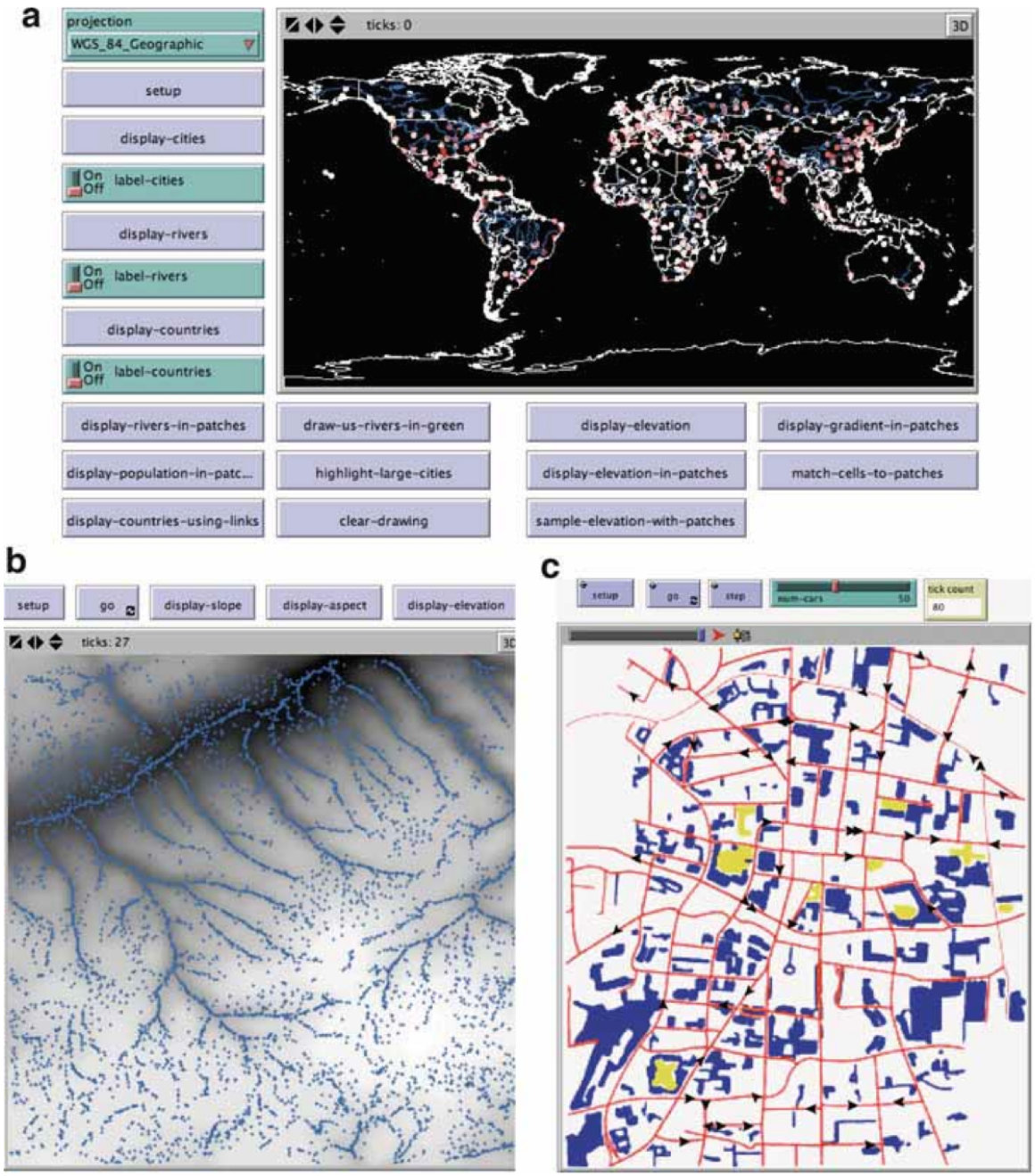

FIGURE 2.6 – Exemples d'integration de GIS dans NetLogo (a) utilisation de shapefiles (b) gradient (c) le modèle de croisière

par son utilisation assez simple. Plusieurs projets de recherche dans des domaines différents ont utilisé Gama afin de répondre à des problématiques variées telles que les risques naturels, l'épidémiologie, l'évolution de l'occupation des sols, la gestion de l'eau avec le projet MAELIA [Gaudou *et al.*2013], la mobilité urbaine avec le projet MIRO 2 [Banos A.2013], ou encore la reconstitution d'événements géo-historiques [Taillandier *et al.*2014]. Plusieurs tutoriels, une documentation complète ainsi qu'une vidéo de démonstration complète sont disponibles sur le site web de la plateforme.

Plusieurs environnements peuvent être définis dans Gama avec un environnement continu de référence assurant la synchronisation entre eux. Gama permet de définir trois types de topologies : l'espace continu, la grille et le graphe. Une topologie est associée à chaque environnement afin de fournir aux agents des services tels que le calcul de voisinage et des distances. Chaque agent dans Gama dispose d'une variable de localisation de type point qui représente le centroïde de sa géométrie. Ainsi, modifier la géométrie d'un agent modifie automatiquement sa localisation. Gama offre plusieurs opérateurs pour effectuer des opérations sur les géométries. Gama est caractérisé par ses nombreux outils permettant l'intégration et la manipulation des données géographiques. Le langage GAML est riche d'instructions facilitant ce type de manipulations. Par exemple, par une simple instruction les agents sont instanciés à partir d'un *shapefile* contenant des données géographiques (les géométries des agents). Ou encore, les géométries des agents et autres variables peuvent être directement sauvegardées dans un *shapefile*. De nombreux opérateurs spatiaux sont aussi fournis tels que les tests géométriques (intersection, croisement, inclusion), les requêtes (calcul de voisinage), les transformations, la triangulation de polygones, la squelettisation et la création de graphes à partir de géométries. Ces opérations sont réalisées en utilisant la librairie JTS. Aussi, les bases de données géographiques telle que PostGis sont interrogeables via GAML [Taillandier *et al.*2014].

Concernant la visualisation des données dans Gama, la 3D est nativement intégrée. L'affichage est géré par la librairie OpenGL ce qui permet de bénéficier des avancées offertes par les cartes graphiques (GPU) en terme de rendu réaliste. Un utilisateur peut facilement centré son affichage sur un agent particulier et le suivre au cours de la simulation tout en définissant différents points de vue. L'utilisation de calques facilite l'abstraction à travers la superposition des indicateurs. Concernant les données en 2D, Gama permet de rajouter des hauteurs aux géométries rendant ces derniers en 3D d'une manière simple. La figure 2.7 illustre un exemple de cette transformation selon deux visions. Gama permet aussi de représenter un modèles numériques de terrain (MNT) dans un environnement 3D et d'exploiter les données d'altitude et de les intégrer dans le comportement d'un agent. La figure 2.7 montre le rendu visuel obtenu par l'ajout d'un MNT à l'exemple précédent.

2.5.5 Repast

Repast (*Recursive Porous Agent Simulation Toolkit*) est développé à l'université de Chicago et actuellement maintenu par le laboratoire Argonne National. Repast est inspiré de la plateforme Swarm et a été initialement implementé sous trois langages : Python (RepastPy), Java (RepastJ) et Microsoft .Net (Repast .Net). RepastPy permet le développement de modèles basiques par des utilisateurs ayant un faible niveau de programmation à travers une interface graphique. Les modèles RepastPy peuvent être exportés/convertis en Java pour des développements avancés sur RepastJ. En effet, RepastJ et Repast Net permettent de développer des

FIGURE 2.7 – Fenêtres de visualisation d'une simulation Gama

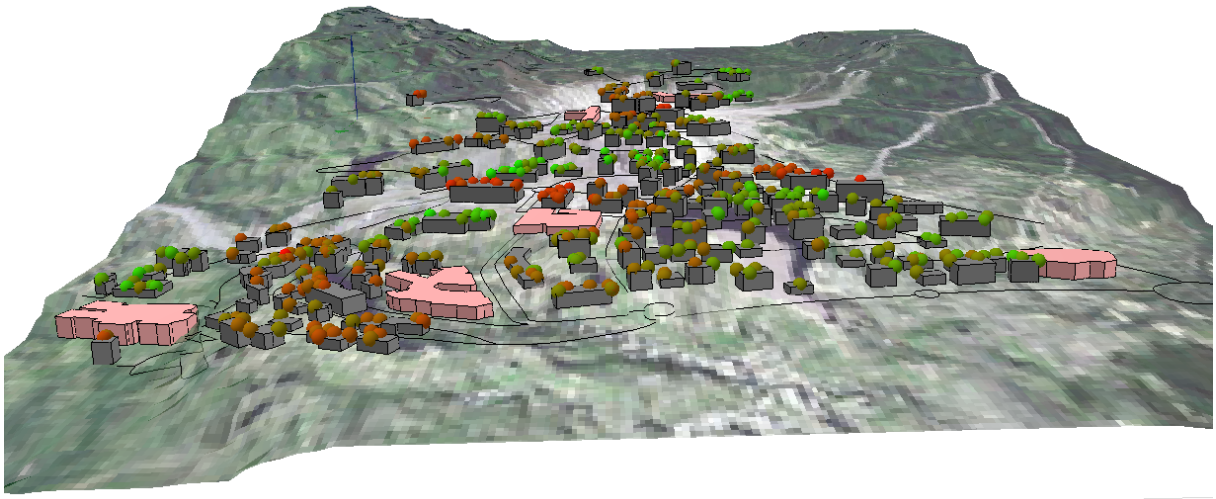

FIGURE 2.8 – Exemple de résultat obtenu après l'ajout d'un MNT

modèles avancées vu qu'ils offrent des fonctionnalités plus complexes que RepsatPy. Ces trois versions ont rapidement laissé la place en 2006 à Repast Simphony (RepastS) qui fournit toutes les fonctionnalités de RepastJ et Repast Net et est implémentée en Java. Cette nouvelle version assure les mêmes fonctionnalisés que les autres et rajoute un environnement d'interfaces graphiques pour des modèles de développement. Aussi, une amélioration de l'interface graphique d'exécution qui permet de créer des affichages en 2D et 3D, des graphiques, interroger les agents et des interfaces avec d'autres programmes (e.g. le logiciel de statistique R). Repast Simphony intègre une librairie d'information géographique (Geotools), et fournit des services additionnels, tels que la modélisation du réseau comme un graphe, le calcul de plus courts chemins, la visualisation et la gestion de données à deux et trois dimensions [Tatara and Ozik2009]. L'extension Agent Analyst permet d'éditer des modèles d'agents Repast à travers ArcGIS [9].

Des méthodes sont aussi fournies pour visualiser les modèles d'agents avec des images satellites, des surfaces élevées et d'autres données scientifiques à travers la bibliothèque interactive du système d'informations géographiques 3D NASA World Wind [10]. La figure 2.9(b) est un exemple de cette visualisation [Crooks and Castle2012]. [Malleson2008] propose un tutoriel qui décrit l'intégration d'autres fonctionnalités GIS à travers la démonstration de la création d'une ville virtuelle via l'importation de *shapefiles* puis le déplacement d'agents sur le réseau de routes (figure 2.9(a)). RepastS permet l'importation de modèles d'agents NetLogo sur sa plateforme via ReLogo. Cette fonctionnalité vise à faciliter le prototypage de modèles d'agents en créant des simples modèles sur NetLogo pour ensuite les étendre sur RepastS [Crooks and Castle2012]. Aussi, RepastS aide au déploiement de simulateurs sur plusieurs hôtes via Repast HPC [Collier2010]. Plusieurs modèles géospatiaux sont développés avec Repast. Nous citons les modèle d'évacuation de piétons d'une station sous-terraine [Castle2007], d'étude de la dynamique résidentielle [Jackson *et al.*2008], d'étude de la criminalité

9. Agent Analyst : Agent Based Modeling Extension for ArcGIS. Available at http ://resources.arcgis.com/en/help/agent-analyst/

10. http ://goworldwind.org/demos/

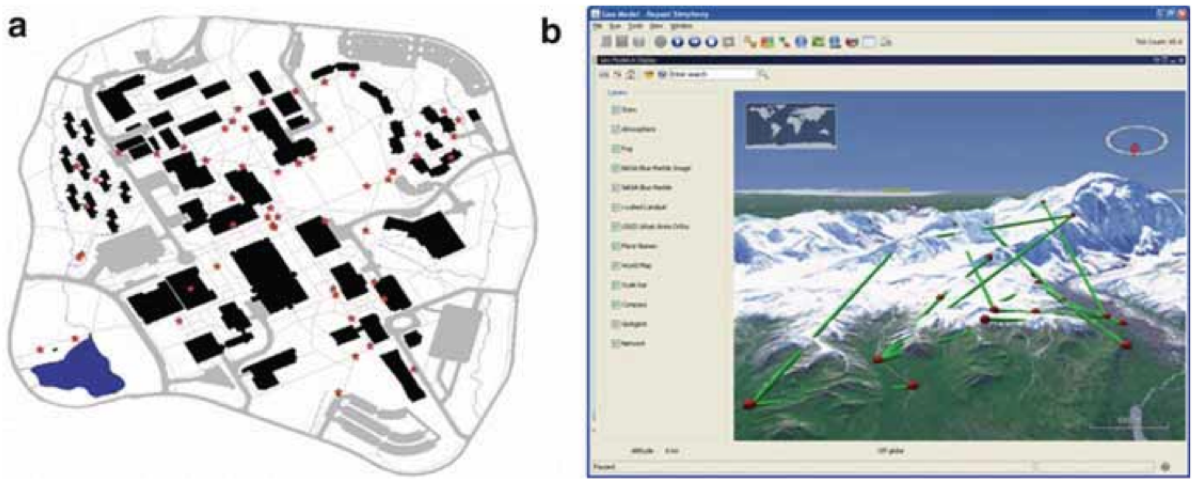

FIGURE 2.9 – Exemples d'un modèle vectoriel à base d'agent sur RepastS. (a) Des agents se déplaçant sur un réseau (b) Un modèle superposé sur *Nasa World Wind*

[Malleson *et al.*2010] et d'étude de la ségrégation [Crooks2010]. L'équipe de développement de Repast a fourni une série d'articles à propos de RepastS. L'architecture et les fonctionnalités de base sont introduites par [North *et al.*2005], et l'environnement de développement est discuté dans [Howe *et al.*2006]. L'enregistrement, l'affichage et le comportement et interaction des agents ainsi que des fonctions pour l'analyse de données (exemple : l'intégration du package de statistique R) sont exposés dans [Macal and North2005]. [Tatara *et al.*2006] fournit une discussion qui explique les capacités de RepastS à travers le développement d'un modèle simple de prédateur. Aussi, Repast dispose d'une large communauté active et une importante base de documentation et de modèles de démonstration disponibles sur le site Web.

2.5.6 Discussion

Toutes les plateformes multi-agents présentées dans cette section pourraient être utilisées dans la modélisation et la simulation de déplacements, dans l'objectif d'évaluer l'impact des informations sur les voyageurs. Cependant, quelques critiques ont guidé notre choix final vers la plateforme Repast Simphony. En effet, Swarm, Mason et Netlogo ne fournissent pas de bibliothèque géospatiale complète, incluant des primitives spatiales. Swarm ni ne donne pas la possibilité de sauvegarder l'environnement géographique résultat [Patrick *et al.*2012].

Les deux plateformes Gama et Repast Simphony sont très proches en termes de fonctionnalités. Notre choix a été guidé par deux considérations. La première concerne la taille de la communauté autour de la plateforme et la réactivité des personnes en charge de la plateforme aux questions des développeurs. La deuxième concerne la facilité de déploiement physiquement distribué de la simulation, qui est l'un de nos axes de recherche en cours. Ces ultimes critères ont fait pencher la balance vers Repast Simphony. C'est pour cette raison que nous fondons notre simulateur multi-agent des déplacements sur cette plateforme (cf. chapitre 3).

2.6 Conclusion

Dans ce chapitre, nous avons présenté les concepts de base du paradigme multi-agent que nous avons choisis comme approche pour concevoir notre simulateur. Cette approche est particulièrement pertinente pour la simulation de mobilité de voyageurs sur un réseau de transport. Le concept d'agent est bien adapté pour la représentation de voyageurs dans les réseaux de transport. Ce sont des entités autonomes, situées dans un environnement, qui adaptent leurs comportements à la dynamique de l'environnement qu'elles perçoivent et interagissent avec les autres agents pour réaliser des buts spécifiques. Ensuite, nous avons mené une étude comparative des plateformes existantes afin de choisir celle qui correspond à nos besoins. Nous n'avons pas trouvé de plateforme « clé en main » permettant de simuler notre système. Nous sommes donc passé à l'étude des plateformes généralistes de simulation multi-agents. Dans notre contexte, deux critères centraux pour choisir la plateforme de simulation concernent d'abord sa capacité à créer des modèles d'agents géospaciaux, i.e. sa capacité d'intégrer et de traiter des données géographiques. Puis, sa capacité à déployer la simulation sur plusieurs hôtes pour pouvoir intégrer un nombre important de voyageurs simulés interagissant dans un environnement distribué et complexe. À l'issue de cette étude, nous choisissons la plateforme Repast Simphony.

Deuxième partie

Contributions

Chapitre 3

Un simulateur multi-agent des déplacements sur les réseaux de transport en commun

Sommaire