Université Cheikh Anta Diop de Dakar

Faculté des Sciences et Techniques

École doctorale Mathématique-Informatique

Thèse de Doctorat

Titre :

Un meta-modèle multi-monde et multi-échelle pour la simulation à base d'agents de systèmes complexes présentant des phénomènes historiques et géographiques : Application à l'étude de la diffusion du rat noir au Sénégal

Présentée par

Pape Adama MBOUP

pour obtenir le grade de

Docteur en Informatique

Soutenue publiquement le 17 mars 2017

devant le jury composé de :

Président :			
Hamidou	Dathe	Professeur	UCAD
Rapporteurs :			
Christophe	Cambier	Maitre de Conférences, HDR	UPMC, Paris, France
Jean-Pierre	Müller	HDR	CIRAD, Montpellier, France
Examinateur :			
Alassane	Bah	Maitre de Conférences CAMES	UCAD
Directeur de thèse :			
Karim	Konaté	Maitre de Conférences CAMES	UCAD
Co-Directeur de thèse :			
Jean	Le Fur	Chargé de recherche	IRD, Montpellier, France

Remerciements

Après avoir remercié *Allah* Le créateur de l'Univers et prié sur son *Prophète Mouhammad* paix et salut sur lui, je remercie mon *Père Tafsir Balla Mboup* et ma *Mère Oumy Thioune*.

Je tiens à remercier :

Monsieur Jean Le Fur, mon co-directeur de thèse, pour la confiance qu'il a portée en ma modeste personne en me proposant cette thèse, en m'encadrant et en me donnant la liberté de la mener à bien. Je le remercie vivement pour sa disponibilité, son sens de l'écoute, son efficacité dans ses remarques et orientations, ses conseils et encouragements, ses corrections ainsi pour tout son soutien permanent malgré la distance géographique qui nous a souvent séparé ;

Monsieur Karim Konaté, mon directeur de thèse, pour la confiance qu'il a lui aussi portée en ma modeste personne en acceptant d'être mon directeur de thèse et en me donnant la liberté de la mener à bien. Merci pour ses remarques pertinentes, ses corrections, ses conseils, pour m'avoir encouragé et soutenu pour le bon déroulement de ce travail ;

Monsieur Jacques Ferber et *Monsieur Fabien Michel* du Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) pour s'être penché avec attention sur mon travail. Merci particulière à *Jacques Ferber* qui m'a donné le terme "monde" à la place du terme "système" ;

Les *membres du jury* d'avoir accepté d'évaluer ce travail. Je commence par remercier sincèrement mes deux rapporteurs *Monsieur Christophe Cambier* et Monsieur *Jean-Pierre Müller* qui m'ont honoré en acceptant de juger en profondeur ma thèse. Je les remercie pour leurs remarques et suggestions pertinentes et précises pour améliorer la qualité du manuscrit. Puis, je remercie sincèrement le président du jury *Monsieur Hamidou Dathe* et l'examinateur *Monsieur Alassane Bah* pour leur disponibilité et l'intérêt qu'ils ont porté à évaluer mon travail ;

Tous les enseignants et professeurs, durant toutes mes études au Sénégal ou ailleurs, qui m'ont prodigué la formation indispensable à la réussite de ce travail, Je remercie particulièrement les professeurs de la faculté des sciences et techniques de l'université Cheikh Anta Diop de Dakar (et ceux qui ont contribué ou qui contribuent à la formation BioMathématique- BioInformatique) ainsi que l'ensemble de son personnel administratif ;

Toute l'équipe du laboratoire BIOPASS de Dakar et toute l'équipe du CBGP de Montpellier pour leur accueil chaleureux et leur hospitalité. Merci particulièrement à *Christophe Amidi Diagne* pour ses corrections et pertinentes suggestions durant la rédaction de ce manuscrit ;

Toutes les *personnes* dont les conversations sur l'informatique, les mathématiques en général, les systèmes complexes ou les systèmes biologiques m'ont stimulé ou inspiré ;

Le Service de Coopération et d'Action Culturelle (SCAC) de l'ambassade de France au Sénégal pour m'avoir octroyé une bourse de mobilité pour financer mes séjours ainsi que les frais pédagogiques à Montpellier et en Tunisie ;

Tout mon *entourage*, mes *amis*, tous *ceux* et toutes *celles* qui m'ont apporté de près ou de loin, leurs précieux concours ;

Mes *parents,* mes *frères* et mes *sœurs* qui m'ont encouragé et soutenu aussi bien pour la réalisation de cette thèse que sur toutes les autres étapes de ma vie. Je les remercie et leur rends un hommage spécial pour leur affection et leur considération.

Enfin une pensée va à l'endroit de mes *grands-parents, oncles* et *tantes.*

A mon bien aimé papa Tafsir Balla Mboup ;

A ma bien aimée maman Oumy Thioune ;

A mes futures épouses et enfants (in cha Allah) ;

A tous mes frères et sœurs qui m'aiment tant et que j'aime tant ;

A ma maman Khady Anta Thioune qui prenait soin de nous lorsqu'on était enfant ;

A ma grand-mère Adji Ngoma Dieng et sa famille qui m'ont hébergé avec hospitalité après l'obtention de mon bac ;

A tous mes parents et amis.

Résumé

La problématique générale de cette thèse concerne la modélisation et la simulation d'un système complexe présentant des *phénomènes historiques* et *géographiques* à plusieurs échelles d'espace et de temps. Le modèle présenté vise particulièrement à simuler l'histoire (sur un siècle) de la diffusion (par le transport humain) du rat noir (*Rattus Rattus*) au Sénégal, des comptoirs coloniaux (extrême ouest) à l'intérieur de la ville de Kédougou (extrême est). Il conduit à une diffusion simultanément représentée sur 3 niveaux d'échelles spatiales et temporelles différentes et d'environnements qui se construisent au fil du temps à partir de données pluridisciplinaires. Cette thèse s'inscrit dans la problématique plus vaste de *modélisation multi-échelle* dans les systèmes multi-agents. Elle répond au besoin d'outils dans ce domaine, en proposant un *meta-modèle multi-agent*, *multi-monde* et *multi-échelle*, couplé à une *approche orientée connaissances et événements* et les *algorithmes* qui lui sont associés. Ceci permet une bonne représentation de phénomènes multi-échelles faisant intervenir plusieurs types de *disciplines* telles que l'histoire, la géographie, la *biologie* ou encore *l'écologie*. Les *mondes* représentent des *sous-modèles complets* et *autonomes*. Ils disposent chacun de leurs propres échelles spatiale et temporelle et un environnement qui peut se construire et se mettre à jour au fil du temps à partir de données appréhendées sous forme d'événements historiques. Les mondes représentés sont *emboîtés* : un monde peut être un sous-modèle représentant une partie d'un autre monde au sein de laquelle on souhaite entrer plus en détail tout en prenant en compte le fait que les mondes s'alimentent mutuellement. L'emboîtement est effectué en *cascade* : la simulation d'un premier monde peut démarrer avant les autres, puis, après un certain temps de simulation, une deuxième simulation pour un deuxième monde qui concerne une zone plus restreinte du premier monde démarre, avec des échelles spatiale et temporelle plus fines. Le processus peut être répété pour des échelles plus fines sans limitation formelle de nombre. Ce meta-modèle prend en compte l'*interaction entre les mondes* de deux façons : *(1)* les agents (transporteurs, pouvant transporter d'autre type d'agents) dans un monde, peuvent choisir des destinations dans d'autres mondes et s'y déplacer en passant par des *itinéraires* et par des *portes inter-mondes*. A l'arrivée dans un monde de destination, les agents *s'adaptent* au nouvel environnement et aux nouvelles échelles spatiale et temporelle ; *(2)* un agent dans un monde est capable de *percevoir* et d'agir sur les agents qui se trouvent dans son disque de perception (son voisinage sur l'espace continu), même s'ils se trouvent dans des mondes différents.

Mots-clés : approche orientée connaissance, approche orientée événement, diffusion rat noir, environnement évolutif, géographie, histoire, modèle multi-échelle, modèle multi-monde, modèle multi-niveau, modélisation, pluridisciplinarité, porte, Sénégal, simulation, sous-modèle, systèmes multi-agents.

Abstract

The general problematic of this thesis concerns the modeling and simulation of complex systems with *historical* and *geographical phenomena* in several space and time scales. The model particularly aimed to simulate the history (over a century) of the diffusion (through human transport) of the black rat (*Rattus Rattus*) in Senegal, from colonial trading posts (extreme west) to the city of Kedougou (extreme east). It leads to a diffusion simultaneously represented in three different levels of spatial and temporal scales and environments that are built over time from multidisciplinary data. This thesis is part of the wider problematic of multi-scale modeling in agent-based model. It meets the need of tools in this field by proposing an agent-based meta-model, multi-worlds and multiscale, coupled with knowledge and event oriented approach as well as the algorithms associated with it. This approach allows a good representation of multiscale phenomena involving several types of disciplines such as history, geography, biology or ecology. Worlds represent complete and autonomous sub-models. They each have their own spatial and temporal scales and an environment that can be built and updated over time using data accounted for as historical events. Represented worlds are nested: a world can be a sub-model representing a part of another world in which we want to enter in more detail taking into account the fact that the worlds feed each other. Nesting is performed in cascade: the simulation of a first world can start before the others and then, after a certain time of simulation, a second one for a second world concerning a smaller area of the first world starts with a finer spatial and temporal scales. The process can be repeated for finer scales without formal limiting number. This meta-model takes into account the interaction between worlds on two ways: (1) agents (carriers, which can transport other types of agents) in a world can choose destinations in other one and travel there via itineraries and inter-worlds doors. Upon arrival in a destination world, agents adapt themselves to the new environment and the new spatial and temporal scales; (2) an agent in a world is able to perceive and act on agents that are located in its perception disk (on the continuous space), even if they are in different worlds.

Keywords : agent-based model, black rat spread, event-oriented approach, evolving environment, gate, geography, history, knowledge-oriented approach, modeling, multidisciplinary, multi-level model, multiscale model, multi-world model, Senegal, simulation, sub-model.

Table des matières condensée

Table des matières étendue

Abréviations

A_ classe Abstraite (dans des diagrammes UML).

ADN Acide DésoxyriboNucléique.

BIOPASS BIOlogie des Populations et d'écologie des communautés Animales des écosystèmes naturels et anthropisés Sahélo-Soudaniens

C_ Classe (dans des diagrammes UML).

CBGP Centre de Biologie pour la Gestion des Populations.

CHANCIRA CHANgements environnementaux, CIrculation de biens et de personnes de l'invasion de réservoirs à l'apparition d'anthropozoonoses, le cas du RAt noir dans l'espace sénégalo-malien.

Coord_Uec Coordonnées sur le repère de l'Espace Continu (dont l'Unité est celle de l'espace continu).

Coord_Ugrille Coordonnées sur le repère de la grille (dont l'Unité est celle de la grille).

CRIO Capacité Rôle Interaction Organisation.

DEVS Discrete Event System Specification.

DPS Durée du Pas de Simulation.

DTG Durée du Tick Global.

DTM Durée d'un Tick Monde.

DTM_{Mi} Durée du Tick Monde pour un monde M_i

DUA Durée de l'Unité d'Activité.

DUA_{Ai} Durée de l'Unité d'Activité pour une liste d'agents Ai.

E libéré (%) Pourcentage de l'Espace libéré.

EMAC Espace Occupé par la Matrice d'Adjacence Classique.

EMAO Espace Occupé par la Matrice d'Adjacence Optimisée.

FIPA-ACL Foundation for Intelligent Physical Agents - Agent Communication Language.

GAMA Gis & Agent-based Modelling Architecture.

GAML Gama Agent-based Modeling Language.

GIS Geographic Information System.

GNT Ground Nut Trade.

GUI Graphical User Interface.

I_ Interface (dans des diagrammes UML).

IA Intelligence Artificielle.

IAD Intelligence Artificielle Distribuée.

IDE Integrated Development Environment.

IODA Interaction Oriented Design of Agent simulations.

IRM4MLS Influence Reaction Model for Multi-Level Simulation (Modèle Influence-Réaction pour la Simulation Multi-Niveau).

KED Kédougou.

MAC Matrice d'Adjacence Classique.

MAO Matrice d'Adjacence Optimisée.

ML Multi-Level.

N1 Nombre de ticks monde au bout duquel les agents d'une liste *Ai* reçoivent la main pour exécuter leur unité d'activité. On va utiliser *N1* quand DUA_{Ai} est supérieure à *DTM*.

N2 Nombre de fois où les agents d'une liste *Ai* ont la main au sein d'un tick monde pour exécuter leur unité d'activité. On va utiliser *N2* quand DUA_{Ai} inférieure à *DTM*.

NDS Nearly Decomposable System.

NKNP Parc National de Niokolo Koba.

NormVectUnitaire Norme du Vecteur Unitaire.

PADAWAN Pattern for Accurate Design of Agent Worlds in Agent Nests.

SIG Système d'Information Géographique.

SMA Système Multi-agents.

SPARK Simple Platform for Agent-based Representation of Knowledge.

TCG Taille des Cellules de la Grille.

TG Tick Global.

Liste des figures

Liste des tableaux

Introduction

I. Questionnement

Face aux interrogations actuelles sur la diffusion de maladies animales passant à l'homme (anthropozoonoses) et de la contamination d'espaces jusque-là indemnes, la question qui nous anime dans ce travail est :

Comment illustrer, afin de mieux comprendre, les processus régissant (i) la diffusion spatio-temporelle des hôtes et (ii) la transmission de pathogènes à l'Homme, à travers la dynamique du rat noir (R. Rattus) et des maladies dont il est porteur dans l'espace sénégalais ?

Le rat noir est un animal commensal (i.e., il vit en étroite association avec l'homme). Il s'agit d'une espèce invasive majeure à travers le monde entier (Global Invasive Species Database http://www.issg.org/database/) qui a colonisé tous les continents depuis son aire natale (Péninsule indienne) en suivant les migrations humaines historiques et actuelles (revue dans [Morand *et al.*, 2015]). En plus de cela, il est porteur de nombreux pathogènes (bactéries et virus notamment) potentiellement zoonotiques (i.e., transmissibles à l'Homme) [Kosoy *et al.*, 2015]. Il participe par exemple, au maintien potentiel de la peste. Les rats *Rattus rattus*, *Rattus norvegicus* et la souris *Mus musculus* ont été les rats les plus concernés pour les épidémies de peste qui ont survenues au Sénégal entre 1912 et 1945 par des déchainements de temps à autre [Granjon et Duplantier, 2009; Pollitzer, 1954]. Par conséquent, le rat noir est pour l'Homme un réservoir redouté d'anthropozoonoses, et les questions liées à son invasion sont préoccupantes. Pour étudier ces questions, relatives à la transmission potentielle ou effective de maladies et particulièrement à l'invasion du rat noir, il faut étudier comment les modifications climatiques et surtout anthropiques qui affectent le pays interagissent pour créer des conditions propices à sa diffusion et par exemple étudier le passage des pathogènes à l'Homme dans des conditions spécifiques de transmission. Dans cette dynamique, l'un des obstacles majeurs à surmonter tient au fait que les problèmes à résoudre résultent de processus multiples qui opèrent et doivent être saisis à différentes échelles d'espace, de temps et d'analyse [Auger *et al.*, 1992; Pumain, 2006].

Dans cette optique, il apparait que le problème à résoudre est :

a) pluridisciplinaire : il résulte de déterminants multiples qui opèrent (population, climat, mobilités, mode de vie, mode d'habitat…) et donc qui doivent être saisis par plusieurs types de *disciplines* telles que la *biologie* (ex. caractéristiques éthologiques du rat noir), la *géographie* (ex. distribution respective du rat noir et de l'Homme au Sénégal), l'*histoire* (ex. évolution dans le temps des facteurs sociaux-environnementaux) ;
b) multi-échelle spatiale et temporelle : les processus se déroulent *(i)* globalement sur le siècle écoulé à l'échelle du Sénégal, *(ii)* régionalement dans les vingt dernières années de développement et *(iii)* localement dans les centres urbains de ces mêmes régions à l'échelle quotidienne du contact entre rats et hommes.

Ces travaux s'inscrivent ainsi dans le cadre du projet CHANCIRA dont l'objectif est d'illustrer et de comprendre les processus qui régissent la diffusion multi-échelle et le passage à l'homme des pathogènes à travers la dynamique de la population du rat noir et des maladies dont il est porteur dans l'espace sénégalais.

II. Positionnement scientifique et proposition de contribution

Pour essayer de répondre à la question, nous proposons de concevoir un modèle (par les systèmes multi-agents) qui permettrait la prise en compte de différents aspects disciplinaires (approche orientée connaissance-événement) et la formalisation de différentes études d'échelles spatio-temporelles différentes.

Ce travail de thèse se situe ainsi dans le domaine de la simulation multi-agent et aborde plus concrètement le problème de la gestion multi-échelle de l'espace et du temps qui constitue un problème important qui n'a pas encore de solution générale [Gil-Quijano *et al.*, 2012; Morvan, 2012, 2013; Pumain, 2012; Ratzé *et al.*, 2007].

Dès lors, nous proposons un meta-modèle générique, multi-monde et multi-échelle pour la modélisation multi-agent de systèmes complexes représentant différents niveaux de détails et/ou d'abstraction, comme ceux présentant des phénomènes géographiques inscrits dans une histoire. Les mondes (ou niveaux d'abstraction) représentent *(i)* des emboîtements en cascade : la simulation d'une phase (représentée par un monde) permet de déterminer certains éléments de la situation initiale pour la phase suivante (représentée par un autre monde) et que la nouvelle phase soit spatialement et temporellement incluse dans la phase précédente et ainsi de suite ; chacun ayant *(ii)* ses propres échelles spatiales et temporelles et *(iii)* un environnement qui se construit et se met à jour au fil du temps à partir de données réelles appréhendées sous forme d'événements historiques (approche orientée connaissances et événements dont nous détaillons la méthodologie plus tard dans ce manuscrit).

Le meta-modèle présente un espace continu et un ordonnanceur général comme pivots de représentation respectivement de plusieurs grilles (environnements) et de plusieurs ordonnanceurs de granularités différentes, avec un environnement et un ordonnanceur par monde. Il prend en compte le fait que des agents dans un monde puissent *(i)* choisir des destinations dans d'autres mondes et s'y déplacer en passant par des itinéraires et par des portes inter-mondes. A l'arrivée dans un monde (de destination), les agents s'adaptent au nouvel environnement et aux nouvelles échelles spatiales et temporelles que sont respectivement la granularité de l'environnement et la durée du pas de simulation de ce monde *(ii)* percevoir et agir sur les agents qui se trouvent dans leur disque de perception (leur voisinage sur un espace continu commun à tous les mondes) même s'ils se trouvent dans des mondes différents. En fin le meta-modèle et les algorithmes qui lui sont associés permettent non seulement d'exécuter (lors d'une simulation) de façon intégrée des mondes ayant des échelles temporelles différentes mais aussi, au sein d'un monde, des unités d'activité de durées diverses des agents.

Ce meta-modèle permet particulièrement de représenter et d'étudier l'histoire de la diffusion du rat noir (*Rattus Rattus*) au Sénégal, à plusieurs échelles d'espace et de temps appréhendées simultanément, avec des environnements qui évoluent sur un siècle. Un questionnement (détaillé plus tard dans ce manuscrit) qui suppose trois niveaux d'abstraction (mondes) avec des échelles spatiales et temporelles différentes que sont (voir Figure 10 et Figure 11) :

1. un monde représentant l'ensemble du territoire national pour étudier sur un siècle (1910 - 2010) la diffusion du rat noir des comptoirs coloniaux vers différentes régions du Sénégal. La simulation de ce monde est réalisé avec un environnement discrétisé en cellules de 7,5 kilomètres (km) de côté et des pas de temps d'1 jour (*échelle nationale-centennale*) ;
2. un monde représentant les régions de Tambacounda et de Kédougou pour étudier la diffusion du rat noir à l'intérieur de ces régions (des régions aux villes). Pour ce monde la simulation porte sur un environnement discrétisé en des cellules de 2 km de côté et des pas de temps d'1 heure (*échelle régionale-décennale*) ;
3. un monde représentant la ville de Kédougou pour étudier la propagation du rat noir à l'intérieur de cette ville (des villes aux quartiers). La simulation pour ce monde correspond à un environnement discrétisé en cellules de 38 m de côté et des pas de temps de 5 minutes (*échelle urbaine-mensuelle*).

Pour ce faire, la mise en place d'un tel meta-modèle apparaissait nécessaire. En effet, une étude de la littérature (présentée plus tard dans ce manuscrit) a montré que cette problématique complexe, purement multi-échelle, multi-monde avec des emboîtements en cascade et des environnements évolutifs (prise en compte de l'histoire et de la géographie), n'était pas aisément représentable avec les architectures classiques de modélisation à base d'agents qui existaient.

Ce cas d'étude demeure un cas d'application pratique de notre travail. Cependant, le meta-modèle présenté dans ce manuscrit peut être étendue pour modéliser d'autres systèmes complexes qui présentent des phénomènes multi-échelles et/ou qui nécessite la prise en compte de données permettant de construire et de mettre à jour des environnements qui sont évolutifs. En effet, notre approche repose sur un ajustement des notions fondamentales de la modélisation à base d'agents (agent / environnement / scheduler) afin d'acquérir une meta-modèle générique.

III. Structure du manuscrit

En dehors de l'introduction et des annexes, cette thèse est structurée en quatre principales parties.

La partie A présente l'état de l'art de la diffusion du rat noir au Sénégal, de l'approche orientée connaissance-événement et de la modélisation par les systèmes multi-agents. Enfin, elle s'intéresse aux approches de modélisation multi-niveau et multi-échelle qui constituent les approches adoptées ici pour appréhender le problème.

La partie B présente la conception et l'implémentation du meta-modèle que nous avons mis en place. Elle est divisée en deux parties. La première s'intéresse aux modèles mono-échelles où le travail de modélisation est représenté et divisé en trois modèles d'échelles spatiales et temporelles différentes comme décrit précédemment. Elle présente ensuite successivement les modèles qui concernent l'échelle *nationale-centennale* (*modèle centennal*), l'échelle *régionale-décennale* (*modèle décennal*) et l'échelle *urbaine-mensuelle* (*modèle contact*). La seconde partie concerne le modèle multi-échelle. Elle propose un meta-modèle permettant l'emboîtement en cascade des modèles mono-échelles et la prise en compte des interactions entre eux.

Dans la troisième partie, nous présentons une discussion générale de ce travail.

Enfin, dans la quatrième partie nous concluons cette thèse - en évoquant sa contribution scientifique plus concrètement à la modélisation informatique par les systèmes multi-agents et - en dégageant un certain nombre de perspectives liées à ce travail.

Partie A
Etat de l’art

Les interactions avérées ou potentielles des rongeurs avec l'homme sont nombreuses, en particulier dans les domaines de l'agriculture [Fiedler, 1988; Leirs, 1994; Skonhoft *et al.*, 2006] et de la santé publique [Gratz, 1997; Meerburg *et al.*, 2009; Taylor *et al.*, 2008]. Néanmoins, jusqu'à très récemment, leur diversité était assez mal connue et en tout état de cause sous-estimée [Granjon et Duplantier, 2009]. Dans cette partie, nous présentons un état de l'art autour du questionnement et de notre proposition de contribution.

IV. Revue des connaissances disciplinaires autour du rat noir et de sa diffusion

Comme nous l'avons expliqué précédemment, le problème à résoudre résulte de facteurs multiples qui doivent être saisis par plusieurs *disciplines* telles que l'*histoire*, la *géographie,* la *biologie* ou encore *l'écologie*. Les connaissances à présenter portent ainsi sur *(i)* les rongeurs, en particulier sur le rat noir (voir IV.A ci-après) et *(ii)* le contexte historique ainsi que géographique ayant conduit à sa diffusion au Sénégal dont particulièrement l'évolution des transports commerciaux (voir IV.B, p. 7).

IV.A Présentation du modèle biologique de l'étude

Les éléments et informations sur la description morphologique, la biologie et l'écologie en général du rat noir ont été synthétisées dans Granjon et Duplantier [2009].

L'espèce du rat noir est *Rattus rattus* du sous-ordre Myomorpha de la super-famille Muroidea, de la famille Muridae et de la sous-famille Murinae [*ibid*]. Ce rat est essentiellement commensal, il vit dans des villes et villages de toute importance. Néanmoins il est parfois capturé aux alentours des villages où il y a des cultures maraîchères et des vergers. Etant surtout nocturne, le rat noir est parfois aussi diurne quand il est en milieu commensal et en fortes densités. Il est arboricole à l'état sauvage. Il se déplace dans les toits en paille des habitations. Il se nourrit surtout des céréales dans les stocks, mais il est omnivore. Il est probable que *Rattus rattus* soit en compétition avec *Mastomys natalensis* et les autres Murinae africains commensaux [*ibid*]. Cet animal peut se reproduire durant toute l'année, tant que les ressources sont disponibles (Ouganda, [Delany, 1975]; Zimbabwe, [Smithers et Black, 1975] ; Nigeria, [Buxton, 1936]). En général il atteint la maturité sexuelle à 68 jours, sa gestation dure entre 20 et 22 jours, son poids à la naissance est 4,5 g et il est sevré au bout de 28 jours [Brooks et Rowe, 1987]. Cette espèce peut être localement très abondante. Par exemple plus de 200 individus ont été capturés en 2 semaines dans un poulailler au Sénégal [Granjon et Duplantier, 2009]. La Figure 2 présente la localisation de tous les points de capture de *Rattus Rattus* par CBGP/BIOPASS en 2009.

Si la *peste* est la *maladie humaine* la plus connue associée au rat noir, il est aussi porteur de *Borrelia crocidurae* au Sénégal [Trape *et al.*, 1996], réservoir de la leptospirose au Zimbabwe [Dalu et Feresu, 1997] et du typhus murin dans divers pays d'Afrique [Gratz, 1997]. Le rat noir est aussi porteur d'hantavirus : des séroprévalences très élevées ont été signalées en Basse Casamance (23 %) et au Sénégal oriental (15 %) [Saluzzo *et al.*, 1985].

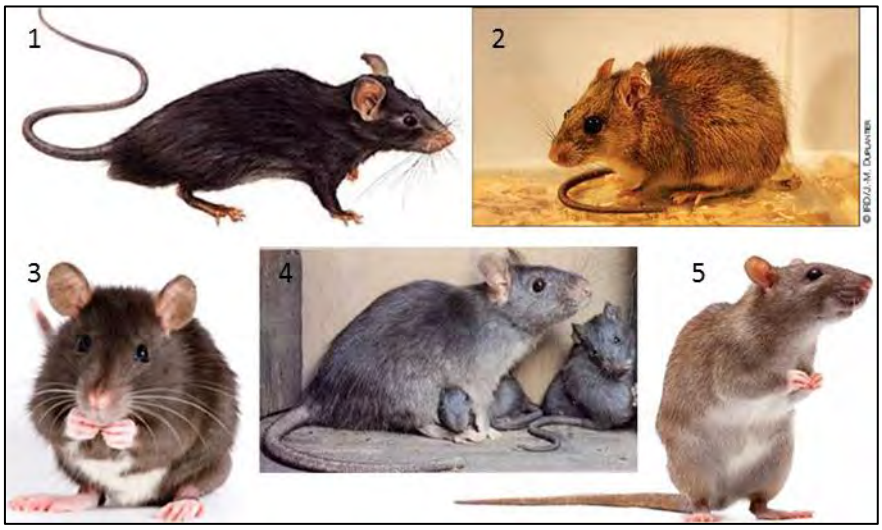

Figure 1 – Quelques images de *Rattus rattus*.
Source : (1) http://nobugs.ca/rodents/roof-rat/, (2) © IRD / JM Duplantier, (3) www.centre-anti-parasitaires.fr, (4) www.planet-mammiferes.org, (5) http://ajspestcontrol.co.uk/).

Figure 2 – Localisation de tous les points de capture de *Rattus Rattus* par CBGP/BIOPASS en 2009.
Source [Granjon et Duplantier, 2009].

IV.B Diffusion du rat noir au Sénégal

IV.B.1 Contexte historique et géographique

Le rat noir a commencé à s'étendre en dehors de sa zone d'origine située dans la péninsule indienne [Musser et Carleton, 2005] il y a environ 10 000 ans. Il a été introduit en Europe il y a environ 2500 ans, et ses populations dans ce continent ont longtemps été confinées aux routes et aux ports commerciaux [Aplin *et al.*, 2003; Audouin-Rouzeau et Vigne, 1994]. Il a atteint l'Afrique de l'Ouest très probablement au cours du 15ème siècle, avec les premiers

navigateurs portugais [Rosevear, 1969]. L'arrivée de populations de rats noirs a été favorisée par l'établissement de ports commerciaux permanents le long de la côte atlantique (Figure 3). Le premier port d'Afrique de l'Ouest a été fondé en 1659, à Saint-Louis (Sénégal), et est rapidement devenu le centre commercial le plus important en Afrique de l'Ouest, en conservant ce rôle jusqu'à la construction de Dakar en 1860. Bien que moins importants, les ports de Banjul et Ziguinchor ont joué un rôle non négligeable dans le commerce depuis respectivement 1816 et 1888 [Sinou *et al.*, 1989].

Le rat noir est resté limité aux zones côtières jusqu'à ce que se développe le transport commercial le long des fleuves Sénégal et Gambie au cours des 18e et 19e siècles ([Sinou, 1981] ; Figure 3). Cette pénétration de l'intérieur des terres le long des rivières est confirmée par la présence dans le Musée britannique de spécimens de *R. Rattus* provenant de Bakel sur le fleuve Sénégal, et à partir de Kuntaur et Maka Colibentan sur le fleuve Gambie [Rosevear, 1969]. Le commerce de la rivière diminua progressivement après les années 1950, et les rats noirs ne sont plus trouvés le long du fleuve Sénégal (Figure 3, 1930-1990). Cependant, une voie alternative pour l'invasion à l'intérieur des terres est apparue dans les années 1930, avec la construction de routes et le développement du transport routier. La répartition des rats noirs au Sénégal à la fin du 20e siècle a été décrite en détail par Duplantier *et al.* [1991]. Les rats sont abondants dans les villages et villes de la région de la Casamance (sud de la Gambie), de Tambacounda (est de la Gambie) et le long de la côte Atlantique (au sud de Dakar). Cependant, ils sont absents de la partie orientale du Sénégal en bordure du Mali et de la moitié nord du pays (Figure 3). Le rat noir est actuellement en train d'envahir le sud-est du Sénégal. Il a été capturé pour la première fois dans plusieurs camps de garde à l'entrée du Parc National du Niokolo Koba dans les années 1980 et dans la ville de Kédougou en 1998 [Bâ, 2002], [Konečnỳ, 2009].

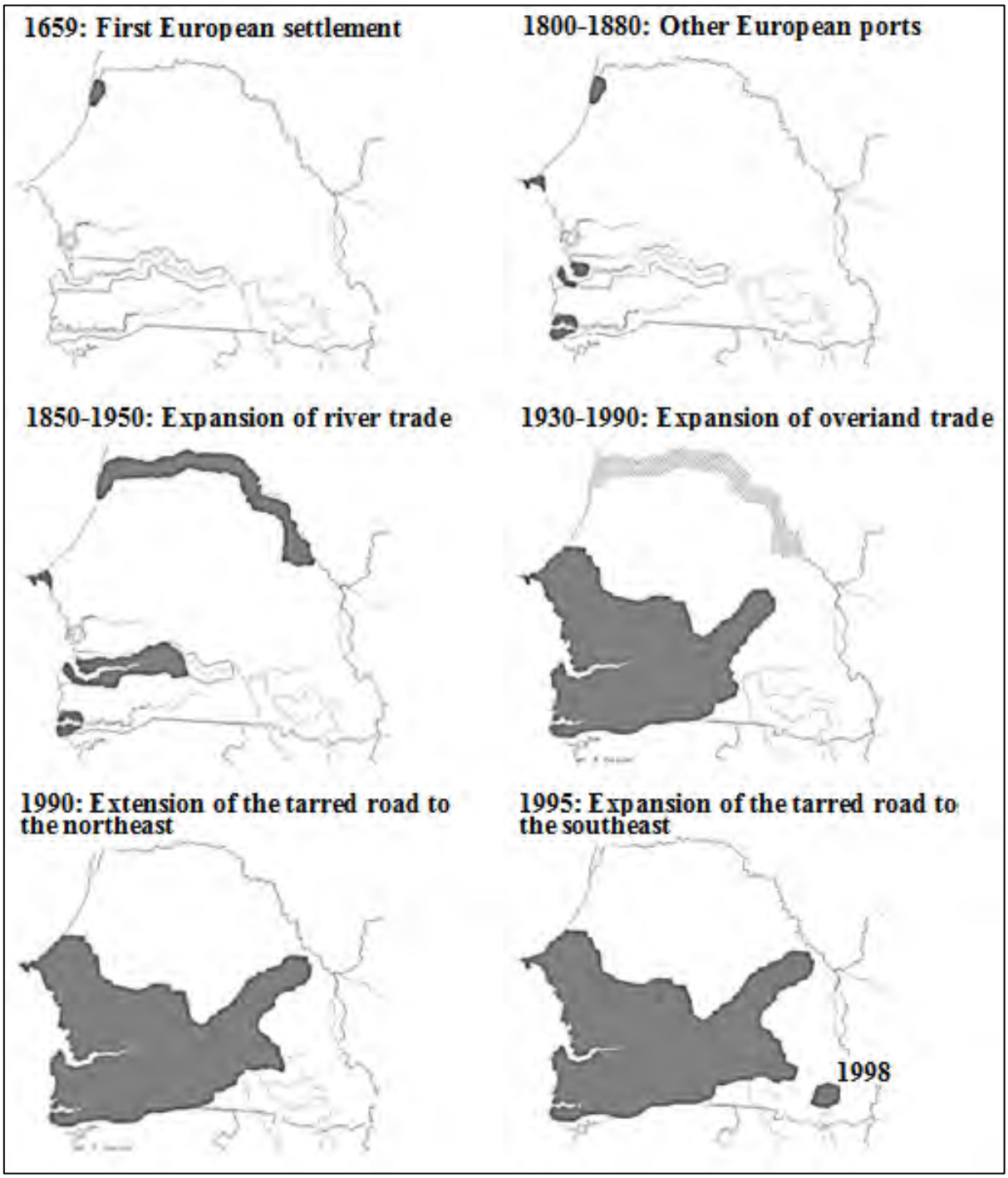

Figure 3 – Invasion du Sénégal par le rat noir fondée sur des données historiques. Les zones grises représentent la répartition approximative au fil du temps (voir dans le texte pour la datation et références). On peut noter la disparition du rat noir le long du fleuve Sénégal, après la diminution du commerce fluvial après les années 1930 (zone ombragée au Nord). Source : [Konečnỳ, 2009].

IV.B.2 Mode de diffusion

L'introduction du rat noir dans la ville de Kédougou vers 1998 [Konečnỳ *et al.*, 2013] indique que les rats de Kédougou viennent directement de Tambacounda. En effet, Entre 2004 et 2007, 257 rats noirs ont été échantillonnés dans 24 sites dispersés sur toute la plage de

répartition de l'espèce au Sénégal (suivis écologiques réalisé par le CBGP et le laboratoire BIOPASS). Cet échantillonnage a permis la détermination de la structure génétique spatiale du rat noir au Sénégal (Figure 4). Cette structuration génétique a montré une étroite parenté entre la population de rat noir à Kédougou et celle de Tambacounda. En plus de cela ils n'ont pas trouvé de rat noir dans les villages se trouvant entre Kédougou et Niokolo Koba. Ce qui montre qu'il ne s'agit pas de progression de proche en proche de village en village, mais bien d'un saut d'une ville à l'autre. Le rat est donc arrivé à Kédougou via les véhicules. Ceci est conforté par le fait qu'il n'a été retrouvé à Kédougou que quelques années après le bitumage de la route entre ces deux villes.

L'invasion du rat noir s'effectue par les transports humains. Les géographes du projet CHANCIRA ont déterminé que ces transports étaient principalement le fait des véhicules de commerce et a priori les gros véhicules où les rats peuvent se dissimuler (bateau, train, camions).

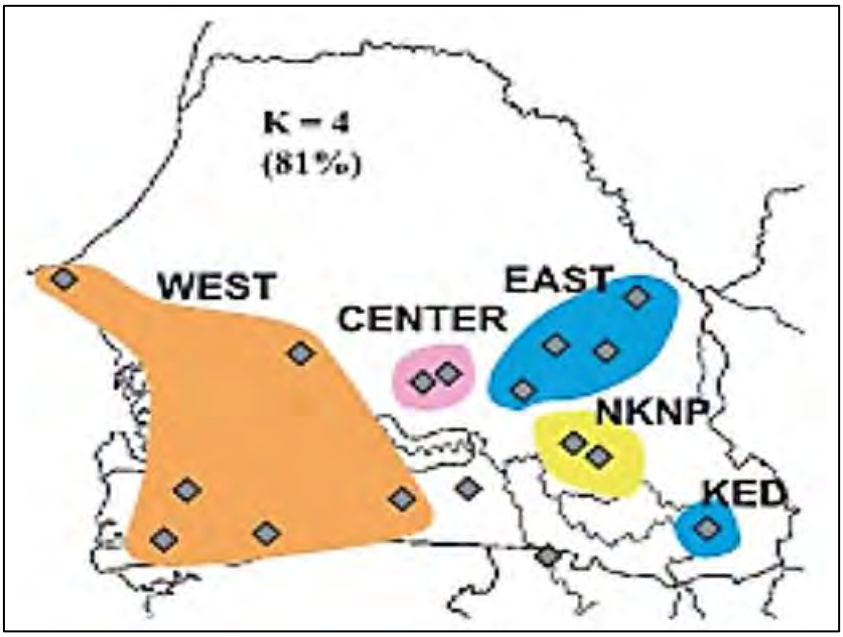

Figure 4 – Sites d'échantillonnage et structuration génétique des populations de rat noir au Sénégal.
Les icones gris correspondent aux sites échantillonnés. Chaque couleur représente un groupe de populations appartenant à un même groupe génétique. KED : Kédougou ; NKNP : Parc National de Niokolo Koba. Source : [Konečnỳ *et al.*, 2013].

Pour que le modèle puisse intégrer en temps utile et indifféremment toutes ces connaissances pluridisciplinaires, nous avons choisi et adapté une approche orientée connaissance-événements dont nous présentons un état de l'art dans la section suivante.

V. L'approche orientée connaissance-événement pour la prise en compte de la pluridisciplinarité et des environnements évolutifs

Dans ce travail, nous ne cherchons pas à construire un modèle théorique sur comment fonctionne un monde ou en essayant de faire des simulations pour retrouver quelles sont les données qui vont avec. Mais nous nous focalisons uniquement sur ce que les différents chercheurs biologistes, écologues, géographes veulent que l'on inscrive dans le modèle. C'est en fonction de ces connaissances (théories et discours accompagnés par fois de données) [Aubert et Müller, 2013; Belem *et al.*, 2011; Le Fur, 2013a] transcrites sous forme de données et ensuite sous forme d'événements, que l'on construit et met à jour de l'environnement de simulation et que l'on prend en compte les événements historiques déterminants du phénomène étudié. En ce qui concerne l'approche orientée événement nous citerons ceux de Worboys [2005] et de Gasmi *et al.* [2015].

Les travaux de Worboys [2005] portent sur les fondations de la théorie de l'information sur lesquelles des modèles explicatifs et prédictifs utiles de phénomènes géographiques dynamiques peuvent être fondés. A partir de séquences de snapshots temporels, il retrace l'évolution de ces fondations à travers les histoires de vie de l'objet, jusqu'aux chroniques d'événements. Cet auteur fait une distinction ontologique entre les « choses » (entités constituantes) et les « événements » (entités occurrentes). Au lieu de ne faire porter la plupart des recherches que sur la représentation de l'évolution dans le temps des entités géographiques, Worboys soutient que les « événements » devraient être mis à niveau à un statut égal à celui des «choses» dans les représentations de la géographie dynamique et suggère des manières de le faire. Il a ainsi développé une théorie orientée événement pure de l'espace et du temps, et suggère les possibilités que la théorie prévoit en l'utilisant pour représenter le mouvement d'un véhicule à travers une région.

Gasmi *et al.* [2015] ont proposé une méthodologie pour construire des modèles à base d'agents d'événements historiques, en particulier les événements de crise, afin de répondre à de nouvelles questions à leur sujet ou de les explorer avec de nouvelles approches. Cette méthodologie consiste à recueillir, numériser et indexer de nombreux documents historiques provenant de diverses sources, construire un système d'information histo-géographique pour représenter les événements concernant l'environnement et le phénomène étudié et enfin concevoir un modèle à base d'agents d'activités humaines dans cet environnement reconstitué.

Cette approche permet ainsi de rendre *dynamique* un environnement de simulation en l'alimentant de données pluridisciplinaires telles que des données historiques et géographiques. Nous présentons dans la section suivante la géographie statique et la géographie dynamique.

VI. Modèles statiques et modèles dynamiques en Géographie

Treuil et ses collègues [2008] définissent les concepts de modèle *statique* et de modèle *dynamique* : un modèle est statique quand « *il a pour propos la représentation de la structure*

d'un système de référence photographié à un instant donné, sans allusion à son évolution dans le temps ». Dans ce cadre un système d'information géographique (SIG) peut être considéré comme un modèle *statique* permettant de répondre à l'aide d'opérateurs d'analyse spatiale à des questions sur le système de référence [Taillandier, 2015]. À l'inverse, un modèle *dynamique* est un modèle qui inclut dans sa représentation « *des hypothèses ou des règles concernant l'évolution dans le temps du système de référence* » [Treuil *et al.*, 2008]. Apporter une composante *dynamique* à la géographie ou aux systèmes d'information géographique permet d'élargir le nombre de questions auxquelles le modèle peut répondre [Taillandier, 2015]. Cette composante dynamique peut être obtenue par une démarche computationnelle c'est à dire la modélisation puis la simulation informatique [Varenne a, 2010].

VII. Modélisation par les Systèmes multi-agents

Dans cette section nous présentons le passage du système réel à la modélisation et la simulation par les systèmes multi-agents avec une revue de quelques plates-formes dédiées, notamment la plateforme SimMasto qui a été utilisée dans ce travail.

VII.A Du système réel à la simulation

Un *système* est une *collection d'objets* interagissant dans un environnement. Les objets, pouvant être eux aussi des systèmes (sous-systèmes), constituent autant d'entités du système et sont caractérisés par un (des) attribut(s) et une (des) activité(s). Une activité est tout processus susceptible de changer l'état du système. Cet état est défini par la description de l'ensemble des entités attributs et activités qui composent le système à un instant t. Un *système complexe* est une collection de sous-systèmes entretenant des relations entre eux (dont les interactions produisent un comportement global qui n'est pas réductible à la somme des comportement locaux) avec chaque sous-système pouvant à son tour être constitué de collections de sous-sous-systèmes… et ainsi de suite jusqu'au niveau d'abstraction choisie [Coquillard et Hill, 1997]. En d'autres termes, un *système complexe* est un système dans lequel les interactions produisent un comportement global qui n'est pas réductible à la somme des comportements locaux [Aniorté *et al.*, 2006].

Pour comprendre le fonctionnement d'un *système réel*, il est nécessaire de faire une *expérimentation* qui consiste à perturber le système selon un protocole déterminé et à l'aide d'un *dispositif expérimental*. Ces expérimentations sont souvent très compliquées voire impossibles à mettre en œuvre. A la place on fait donc recours aux expérimentations informatiques que l'on appelle *simulations* (qui est un modèle plongé dans le *temps* [Coquillard et Hill, 1997]). Dans ce cas, le dispositif expérimental est informatique et s'appelle *simulateur* [Treuil *et al.*, 2008]. Un modèle étant défini par Ferber et Perrot [1995] comme suit :

> *Un modèle, en science, est une image stylisée et abstraite d'une portion de réalité. L'intérêt d'un modèle est d'abord d'être plus explicite, plus simple et plus facile à manipuler que la réalité qu'il est censé représenter. Les modèles éliminent ainsi un grand nombre de détails considérés comme inutiles par le modélisateur afin de mieux se*

consacrer aux données que celui-ci juge pertinentes relativement au problème qu'il désire résoudre.

VII.B Systèmes multi-agents

La modélisation par les systèmes multi-agents est un domaine assez connu et largement utilisé pour étudier les systèmes complexes dans divers domaines [Coquillard et Hill, 1997; Epstein, 2006; Ferber et Perrot, 1995; Gilbert, 2007; Railsback et Grimm, 2011; Resnick, 1994; Treuil *et al.*, 2008]. Nous allons donc, sans entrer dans le détail, décrire ci-dessous les principes et utilités de la modélisation par les systèmes multi-agents.

La modélisation du comportement intelligent d'une seule entité informatique ou agent est l'intelligence Artificielle (IA) classique. Pour traiter plusieurs agents, l'intelligence artificielle distribuée (IAD) cherche à représenter le comportement collectif via le comportement individuel pour la résolution de problèmes complexes. Elle s'intéresse alors aux comportements intelligents provenant de l'activité coopérative de plusieurs agents. Ce qui introduit le concept de système multi-agents.

VII.B.1 Définitions

Suivant les auteurs, il y a plusieurs définitions de la notion de système multi-agents. Nous retenons celle de Ferber et Perrot [1995]. On appelle *système multi-agents* (ou SMA), un système composé des éléments suivants :

a. un environnement E, c'est-à-dire un espace disposant généralement d'une métrique ;
b. 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 ;
c. un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système ;
d. un ensemble de relations R qui unissent des objets (et donc des agents) entre eux ;
e. un ensemble d'opérations permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O ;
f. 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.

Ces mêmes auteurs proposent une définition minimale commune du terme *agent* dans laquelle un *agent* est une entité physique ou virtuelle :

a. qui est capable d'agir dans un environnement ;
b. qui peut communiquer directement avec d'autres agents,
c. qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser) ;
d. qui possède des ressources propres ;
e. qui est capable de percevoir (mais de manière limitée) son environnement ;
f. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune) ;

g. qui possède des compétences et offre des services ;
h. qui peut éventuellement se reproduire ;
i. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

Les objets (y compris les agents) ont donc des attributs qui varient dans le temps. L'ensemble des valeurs des attributs d'un objet à un instant donné constitue l'état de cet objet, et la réunion de l'ensemble des états des objets forme l'état du système [Treuil *et al.*, 2008].

VII.B.2 Les systèmes multi-agents et la Géographie dynamique

Le concept central sur lequel la modélisation multi-agents se fonde est donc la notion d'*objet*. En informatique, « *un objet est un conteneur symbolique et autonome qui contient des informations et des mécanismes concernant un sujet, manipulés dans un programme* » [Unhelkar, 2005]. En géographie, ce concept est pris dans un sens « physique » très général et non pas seulement à un niveau technique, celui de la programmation orientée-objet. Etant avant tout un *objet*, un *agent géographique* peut donc représenter un être humain (un animal en général), un camion, une ville, une parcelle de terrain, une route, une goutte de pluie, …, voire une nation entière [Daudé et Langlois, 2006].

L'approche orientée objet/agent est donc particulièrement intéressante pour étudier des systèmes complexes tels que les problématiques géographiques. En effet, en plus de son formalisme objet-agent, cette méthode permet de modéliser et de simuler de manière simple différents types d'agents géographiques, leurs interactions, et d'analyser de manière intuitive le fonctionnement global du système [Lammoglia, 2010]. En effet ce sont les acteurs individuels qui font ou produisent les phénomènes collectifs [Daudé, 2006]. Ce point de vue correspond exactement à la modélisation par les systèmes multi-agents [Ferber et Perrot, 1995].

La Géographie artificielle, reliée à l'informatique, en particulier les systèmes multi-agents, est un "laboratoire" qui permet la formalisation et la validation de processus locaux aptes, grâce à la simulation, à produire des *dynamiques* et des structures spatiales de niveaux supérieurs ou patterns [Daudé et Langlois, 2006; Grimm *et al.*, 2005]. La souplesse qu'il offre à la formalisation est aussi appropriée pour représenter et étudier à coût moindre (par rapport à une expérimentation réelle) des phénomènes où des acteurs variés évoluent dans des *espaces composites* à *plusieurs échelles de temps* [DeAngelis et Mooij, 2005; Ferber, 1999; Grimm et Railsback, 2013; Lammoglia, 2010; Varenne a, 2010].

Nous allons à présent passer en revue quelques plates-formes de simulation multi-agents, en particulier la plate-forme SimMasto qui a été utilisée dans ce travail.

VII.B.3 Plates-formes de simulation multi-agents

Un grand nombre d'outils (plates-formes, toolkits, Frameworks) de modélisation et simulation à base d'agents ont été développés [Allan, 2010]. La plupart d'entre eux ont terminé leur cycle de vie avant même d'être connu par les utilisateurs. Seul un petit nombre d'entre eux sont adoptés par les modélisateurs et sont devenus populaires [Vo, 2012]. Ces

outils offrent des environnements de développement de modèles équipés de langages de programmation ou de modélisation pouvant exister sous plusieurs formes. On peut citer les langages de programmation généraux tels que Java et C ++ ; les langages de domaine spécifique tels que le langage Logo (utilisé dans NetLogo [Wilensky, 1999a], ReLogo de Repast Symphony [OZIK, 2012], ...), GAML (le langage de GAMA) ; les langages à base de graphiques tel que Groovy (Repast Symphony) [Tatara et North, 2012]. Dans cette section nous faisons une revue des langages de modélisation de cinq plates-formes de modélisation et de simulation multi-agents que sont : NetLogo, MASON, GAMA, Repast et SimMasto. Les quatre premiers sont des plates-formes populaires de modélisation à base d'agents aujourd'hui. SimMasto est un projet de plate-forme de modélisation à base d'agents actuellement développés dans notre équipe et dédié à la connaissance des rongeurs. Ainsi nous allons brièvement présenter les plates-formes NetLogo, MASON, GAMA et Repast Symphony. Après cela, nous présenterons de façon plus détaillée SimMasto car le modèle que nous proposons est implémenté dans cette plate-forme qui est elle aussi bâtie sur Repast Symphony.

VII.B.3.a NetLogo, MASON, GAMA, Repast

NetLogo[1] [Wilensky, 1999b], écrit par Uri Wilensky, est une plate-forme de modélisation à base d'agents développée au CCL ("Center for Connected Learning and Computer-Based Modeling"). Son langage de modélisation, inspiré du langage Logo [2012], fournit des concepts adaptés au développement de modèles à base d'agents spatialisés (par exemple, tortue, race). Le langage de modélisation de NetLogo cache toute la complexité d'un langage d'usage général (*e.g.* Java). Ce qui permet aux modélisateurs "non-informaticiens" d'écrire facilement leur modèle.

MASON[2] est une plate-forme de simulation multi-agents à événements discrets. Elle est développée en Java à l'Université George Mason [Luke *et al.*, 2005]. C'est une plate-forme rapide. Son premier objectif de conception est de supporter la simulation d'un grand nombre d'agent (jusqu'à un million). MASON offre un ensemble de classes que le modélisateur peut étendre et utiliser pour développer son modèle. Elle contient à la fois une bibliothèque de modèles et une suite facultative d'outils de visualisation en 2D et 3D. Cependant elle est destinée à être utilisé par les programmeurs expérimentés en Java.

GAMA[3] ("Gis & Agent-based Modelling Architecture") est la plateforme la plus avancée de modélisation multi-agents et multi-niveau (voir VIII.D et VIII.E) et permettant d'étudier les phénomènes géographiques [Drogoul *et al.*, 2013; Gasmi *et al.*, 2015; Gaudou *et al.*, 2014; Soyez, 2013; Taillandier, 2015; Taillandier *et al.*, 2014; Vo, 2012]. La plate-forme GAMA [Grignard *et al.*, 2013] permet d'intégrer facilement des données SIG et offre la possibilité d'utiliser FIPA-ACL [FIPA, 2002] pour la communication des agents [Gasmi *et al.*, 2015]. GAMA est dotée d'un langage GAML ("Gama Agent-based Modeling Language") facile à comprendre et à utiliser. Elle est particulièrement équipée d'une représentation spatiale bien

[1] https://ccl.northwestern.edu/netlogo/, consulté le 14 mars 2016
[2] http://cs.gmu.edu/~eclab/projects/mason/, consulté le 14 mars 2016
[3] https://code.google.com/p/gama-platform/, consulté le 09 décembre 2015

outillée intégrant de nombreux modèles avec des agents ayant des géométries (permettant de représenter leur forme) et des environnements ayant différents types de topologies (espace continu, graphe, grille, données géographiques réalistes ...) en 2D et en 3D où les agents peuvent évoluer [Taillandier *et al.*, 2014; Vo, 2012; Vo *et al.*, 2012].

Repast ("The Recursive Porous Agent Simulation Toolkit") *Simphony*[4] est un système multi plates-formes très interactif de modélisation à base d'agents basé sur *Java*. Il est développé au Laboratoire national d'Argonne [Collier *et al.*, 2003; North *et al.*, 2013] et incorporé dans l'environnement de développement *Eclipse*[5]. Il est open-source et prend en charge le développement extrêmement flexible de modèles d'interaction d'agents pour une utilisation sur des ordinateurs personnels et sur des machines en clusters de petite taille. Il offre un environnement avec trois formalismes pour le développement de modèle. Le premier formalisme est Java avec lequel le modélisateur peut étendre et utiliser les classes utilitaires fournis par Repast. Le second formalisme est ReLogo [OZIK, 2012], inspiré du langage de modélisation de NetLogo, avec lequel le modélisateur peut développer son modèle en utilisant les principaux concepts tels que: "turtle", "link", "observer". Le troisième formalisme concerne les organigrammes [Tatara et North, 2012] dans lequel le modélisateur peut construire son modèle grâce à un ensemble d'icônes graphiques prédéfinis.

Dans Repast Simphony il est possible d'utiliser quatre types de topologie (projections) que sont : SIG, réseau, grille et espace continu.

Repast Simphony offre un *modèle de paramètres* permettant de définir les paramètres utilisés dans une simulation tout en donnant à l'utilisateur la possibilité de les changer à sa guise grâce à un onglet "Parameters" de l'interface graphique[6] (Figure 5).

[4] http://www.repast.sourceforge.net/, consulté le 14 mars 2016
[5] http://www.eclipse.or, consulté le 14 mars 2016
[6] Repast Simphony gère l'interface graphique de la simulation

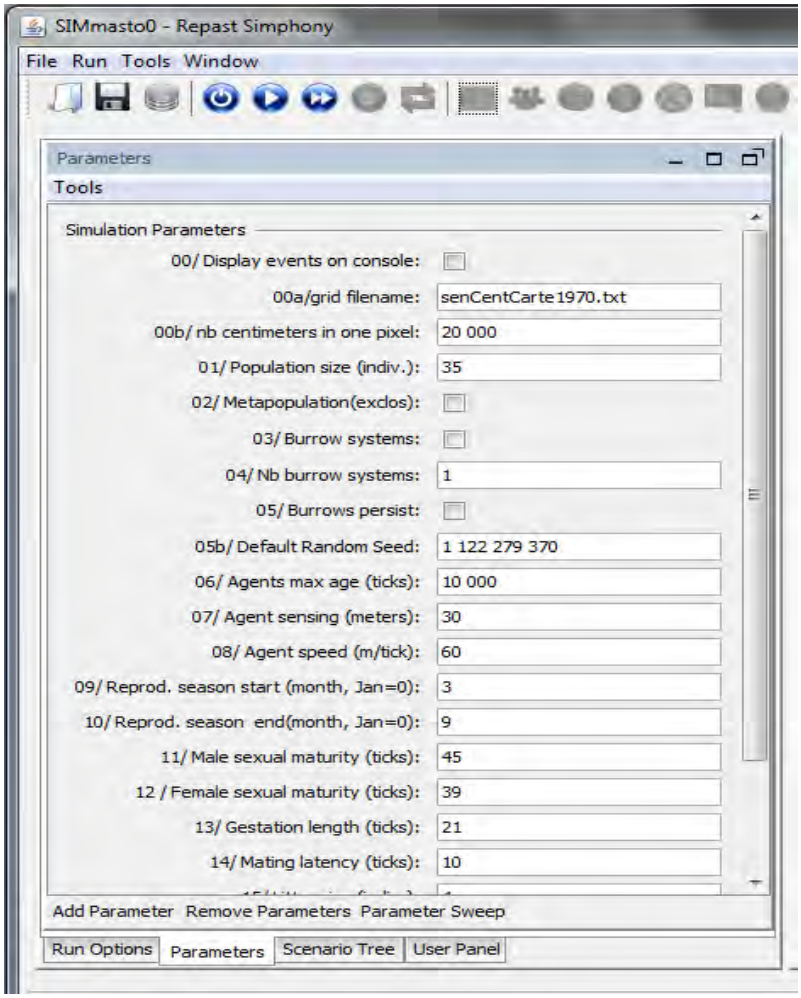

Figure 5 – L'onglet Parameters de Repast Simphony.

VII.B.3.b Le projet SimMasto

Hébergé par le laboratoire CBGP[7], un centre de recherche pluridisciplinaire français en biologie, et développé sous Repast Simphony, SimMasto est un projet de plate-forme de simulation de population de rongeurs. Il est conçu comme un projet expérimental et vise à être un centre de connaissances dynamiques sur la coévolution bioécologique des petits rongeurs et des parasites qu'ils hébergent. Le champ y est également considéré comme une approche

[7] Centre de Biologie pour la Gestion des Populations (http://www6.montpellier.inra.fr/cbgp)

intégrée de toute échelle, des gènes à l'écosphère. Son objectif est de contribuer à une meilleure compréhension de l'état, du rôle et du destin de ces nuisibles en coévolution.

Le projet SimMasto est développé par Jean Le Fur et son équipe [Le Fur *et al.*, 2017a] depuis 2007 en France et au Sénégal. Il offre un environnement de développement intégré (IDE) pour le développement de modèles et l'exécution de simulations. Son langage de programmation est Java.

Le projet SimMasto est conçu avec un objectif de robustesse et de flexibilité, ce qui lui permet d'englober plusieurs simulateurs permettant d'étudier des processus d'hybridation, de ravage des cultures, des expérimentations en cages et en enclos, un protocole de capture-marquage-recapture pour l'étude de la dynamique d'une population de rats, …, et ici, la diffusion et la propagation multi-échelle du rat noir (Figure 6). Il est doté d'une interface de sortie des résultats sous forme de courbes, de diagrammes et de graphes.

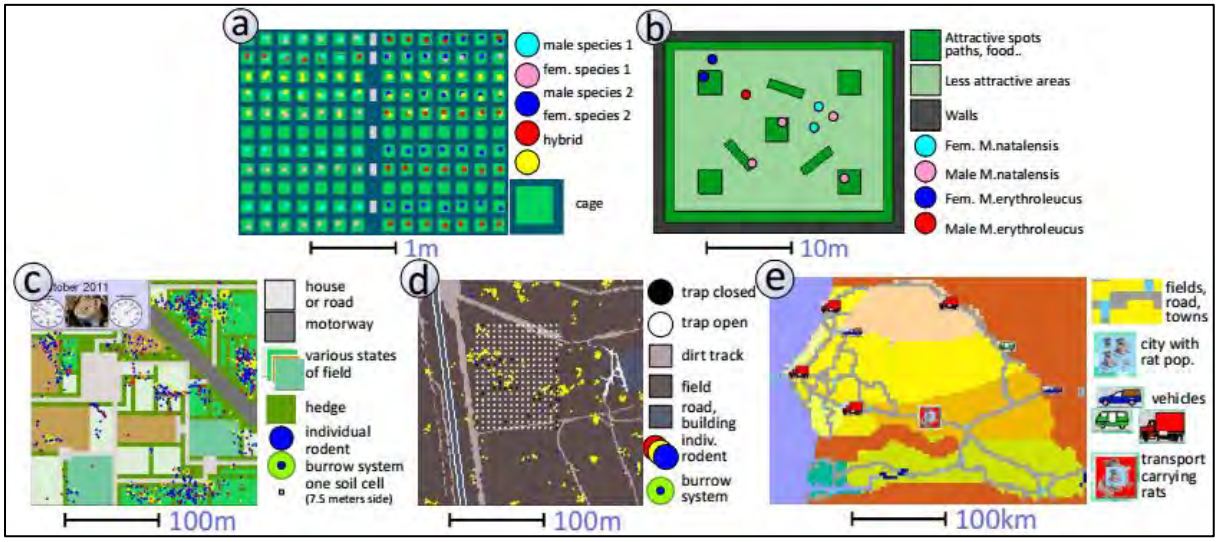

Figure 6 – Quelques cas d'utilisation intégrés dans SimMasto.
(a) expériences d'hybridation en cage et (b) enclos sur des espèces jumelles, (c) population de campagnols évoluant dans un paysage agricole, (d) expérience de capture-marquage-recapture dans une réserve africaine, (e) épidémiologie et transport du rat noir par des véhicules commerciaux au Sénégal (Afrique de l'Ouest). (Source [Le Fur, 2013a]).

Objectif de SimMasto

Les connaissances sur les rats couvrent de nombreux niveaux fonctionnels, de l'ADN aux écosystèmes, ainsi que de multiples échelles d'observation. L'objectif de SimMasto est donc d'élaborer un modèle permettant une représentation intégrée de ces éléments et phénomènes.

Pour cela la contrainte quasi-unique imposée au système formel à construire est la robustesse. Pour être valide, il est en effet nécessaire que la plateforme soit apte à intégrer toute problématique du domaine, quelle que soit sa dimension spatiale, temporelle ou fonctionnelle et quelle que soit son actualité, autrement dit qu'elle corresponde à des problématiques définies il y a dix ans, de nos jours ou dans dix ans.

Nous présentons dans ce qui suit la méthodologie dans SimMasto et l'état de la plate-forme avant l'intégration de notre travail.

Méthodologie et Modèle de SimMasto

Compte tenu de la variété des domaines qui peuvent être abordés (de l'ADN à l'écosystème), la construction du modèle est abordée selon un mode incrémental où des études de cas contrastées sont successivement représentées au sein de la même plateforme. Chaque étude permet d'enrichir la plateforme et bénéficie des développements antérieurs. La contrainte de robustesse qui fonde la validité de l'approche nécessite que la plateforme (*viz*. le modèle général) soit compatible simultanément avec toutes les études de cas représentées.

Après un certain nombre d'études de cas et avant l'apport de mon travail, le modèle simplifié de SimMasto peut être présenté par le diagramme UML de la Figure 7 ci-après.

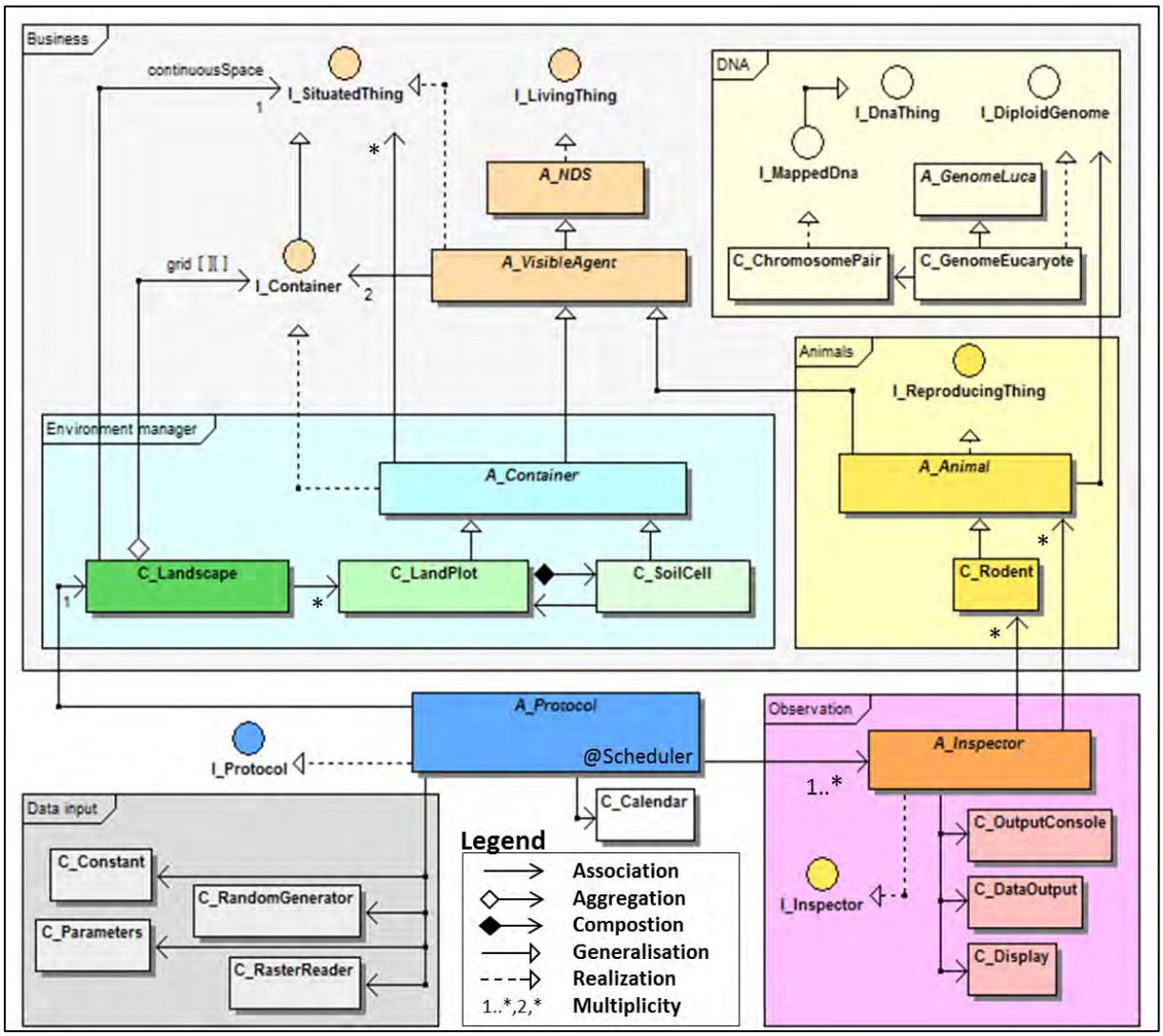

Figure 7 – Modèle simplifié de SimMasto avant intégration de notre travail.
Nomenclature des classes: *I_* : interface, *A_* : classe abstraite, *C_* : autre classe.

Ce modèle peut être subdivisé en 4 modules que sont :

- Un module *business* représentant le cœur du modèle. Il prend en charge la création et la gestion de l'environnement et des agents (animaux) qui y évoluent :

  - *I_SituatedThing* représente une interface que va implémenter tous les objets et agents devant être situés sur l'environnement. Elle contient ainsi des méthodes abstraites permettant la localisation ;
  - *I_LivingThing* permet l'implémentation de caractères fondamentaux aux êtres vivants (naissance, vieillissement) leur permettant d'évoluer au cours d'une durée de 'vie' ;
  - *A_NDS* permet d'appréhender les agents et objets en tant que système quasi-décomposable [Simon, 1962]. C'est la réalisation de *I_LivingThing* ;
  - *A_VisibleAgent* permet à certains agents d'être visibles sur l'interface graphique (GUI[8]) et de prendre la forme souhaitée.

  Pour la gestion de l'environnement :

  - *I_Container* représente une interface que vont implémenter toutes les structures pouvant contenir des objets et des agents. Elle contient ainsi des méthodes abstraites permettant de déterminer par exemple la liste d'agents dans ce conteneur, et/ou récursivement dans les conteneurs qu'il contient ;

  - *A_Container* est une classe abstraite qui implémente *I_Container* et qui permet aux conteneurs de contenir des *I_SituatedThing*. Donc un *A_Container* est un *I_SituatedThing* pouvant contenir d'autres *I_SituatedThing* en particulier d'autres conteneurs. Les conteneurs sont des agents visibles (voir *A_VisibleAgent*) ;

  - *C_SoilCell* est une classe qui étend *A_Container*. L'environnement est discrétisé en cellules élémentaires de même taille, objets de la classe *C_SoilCell*. Cette dernière pouvant contenir plusieurs informations et permettant de connaître au besoin position (*I_SituatedThing*), nombre d'agents (*A_Container*), affinité pour les rats, types de terrain… La taille des cellules dépend du niveau de détail que l'on a choisi ;

  - *C_LandPlot* : Un ensemble de cellules élémentaires (*C_SoilCell*) contiguës de même type est un *C_LandPlot*. Cette classe composée de *C_SoilCell* permet la délimitation et l'identification des différentes zones de l'environnement. Une cellule (*C_SoilCell*) connait dans quel unique *C_LandPlot* elle se trouve. Les *C_LandPlot* sont aussi des conteneurs ;

  - *C_Landscape* : cette classe se charge de la construction et de la gestion globale de l'espace ainsi que de mettre en relation les agents et l'espace au cours de la simulation. Il contient :
    - l'espace continu qui représente une topologie où les rats se déplacent ;

[8] Graphical User Interface

- une grille (matrice) de *I_Container*, dont les éléments sont instanciés en *C_SoilCell*s, représentant la discrétisation de l'environnement et une topologie qui permet aux agents de percevoir leur voisinage (voisinage de Moore, Von Neumann). Les valeurs (affinités) de cette grille sont lues à partir de fichiers de type raster avec la classe *C_RasterReader* ;
- des listes et des procédures permettant détection, construction et stockage des *C_landPlots* représentant les différentes zones de l'espace.

Pour la gestion des agents évoluant sur l'environnement :

- *I_ReproducingThing* permet d'appréhender ces agents comme un système quasi-vivant [Kelly, 1994; Le Fur, 2013b], susceptibles de se reproduire ;
- *A_Animal* : c'est une classe abstraite disposant des propriétés de mouvement propre à un animal lui permettant de mettre en œuvre un schéma comportemental de type perception-délibération-décision-action adapté de Ferber et Perrot [1995]. Elle dispose d'un ADN (voir la partie ADN) et implémente *I_ReproducingThing*, ce qui lui permet de se reproduire tout en transmettant son ADN à ses descendants. Elle étend *A_VisibleAgent* ce qui lui permet d'être visible sur le *Display* et de changer de forme suivant son état ;
- *C_Rodent* est une classe qui hérite de *A_Animal* et permet de créer des agents rats avec des propriétés spécifiques ;

Pour l'aspect génétique (ADN) :

La partie nommée ADN représente tous les aspects liés à la génétique du vivant (dont seules certaines classes sont représentées ici). Elle comprend *(i)* les éléments mécaniques (gènes, allèles…) permettant les opérations liées aux génomes telles que méiose, ségrégation, fécondation[9], *(ii)* un modèle mimant une phylogénie et représenté par des génomes (*A_GenomeLuca*, *C_GenomeEucaryote*). Chacun des génomes agrège diverses paires de chromosomes (*C_ChromosonePair*) héritables par reproduction. Ce paquetage exploite le principe d'héritage du paradigme objet pour instancier les caractéristiques (traits) des objets vivants (agents) de façon héritable et cumulative jusqu'aux valeurs spécifiques aux espèces. Cette part du modèle est considérée comme une base de connaissances utilisable par les agents pouvant se reproduire.

Ce modèle est connecté à trois autres composantes d'importance identique que sont les modules *donnée*, *Protocole* et *observation* :

- Le module *donnée* (*Data input*) gère les constantes et les données d'entrée pour un modèle. Il est constitué principalement des classes suivantes :

[9] Une grande part du code liée à cette partie de la plateforme est issue des travaux de Shaw et Wagner [2008] sur les génomes de criquets et des travaux de Comte [2012a; 2012b].

- *I_Constant* est une interface contenant les constantes numériques et alphanumériques ;
- *C_Parameters* est une classe permettant de lire les données en paramètre (les variables du modèle) ;
- *C_RandomGenerator* est une classe contenant les générateurs de nombres aléatoires et permettant de fixer les graines pour garantir la répétabilité des simulations;
- *C_RasterReader* est une classe permettant la lecture de fichier Raster pour la construction de l'environnement discrétisé.

- Le module *observation* qui s'occupe de la présentation des données de sortie d'une simulation dans laquelle sont implémentées :

  - *A_Inspector* est une classe abstraite qui est étendue par un ensemble d'inspecteurs constituant un système épiphyte (ou conseiller) [Pachet *et al.*, 1994]. Ces objets inspecteurs se chargent de recenser et de restituer, au besoin et de façon multimodale, les données produites par une simulation. Ils contiennent par exemple la liste des rats males, femelles… Il y a un inspecteur général commun à tous les simulateurs et un ou plusieurs inspecteurs propres à chaque simulateur ;
  - *C_OutputConsol* et *C_Display* gèrent respectivement l'affichage des sorties sur la console (textes) et sur le *Display* (GUI, en image animée, Figure 6) ;
  - *C_DataOutput* gère l'écriture des données de sortie dans des fichiers.

- Le dernier module contient les *Protocol*es :

  Chaque étude de cas agrégée dans le modèle est caractérisée et automatisée par un *Protocole* qui étend la classe abstraite *A_Protocol*. Un *Protocole* est donc la classe centrale du simulateur (pour une étude de cas donnée). On peut l'assimiler à la définition d'un monde car toutes les autres parties sont greffées sur lui. Il dispose de :
  - un gestionnaire de données d'entrées ;
  - un gestionnaire d'environnement ;
  - des inspecteurs permettant de collecter les données produites par le simulateur au cours de la simulation et de les présenter à la partie observation ;
  - un calendrier permettant de convertir la durée des pas de simulation en unités temporelles, par exemple une durée de pas de simulation peut valoir 1 jour ou 2 minutes etc. Il s'incrémente ainsi avec l'incrémentation des pas de simulation. Le calendrier permet aussi de gérer les dates au cours d'une simulation. Ce qui permet par exemple de connaitre l'âge des agents, la saison de reproduction etc.

  Le *Protocol*e distribue donc les tâches, crée les agents, les référence dans les inspecteurs concernés, demande au gestionnaire d'espace (*C_Landscape*) de les positionner sur l'environnement. A l'aide d'un scheduler intégré dans Repast (@Scheduler()), le *Protocol*e gère à chaque pas de temps (tick) son environnement et donne la main à chaque agent pour qu'il réalise son propre step (ses actions pour ce tick). La durée du pas de temps dépend du niveau de détail choisi lors d'une étude de cas.

L'approche utilisée, fondée sur l'articulation incrémentale d'études de cas contrastées dans une même structure formelle, a fait que la plate-forme SimMasto était assez avancée (après 4 études de cas) pour représenter et étudier les rats (déplacement, reproduction, hybridation …). Cependant elle présente des limites à la représentation de phénomènes liés à la problématique de notre thèse. C'est-à-dire la diffusion des rats par le transport humain sur un environnement qui évolue sur un siècle et simultanément sur des échelles spatiales et temporelles différentes.

Donc en respectant la démarche du projet *i.e.* l'approche incrémentale, notre objectif est *(i)* de nous servir de cette plate-forme bien avancée sur la représentation des rats, et *(ii)* de repousser ses limites en intégrant des fonctionnalités lui permettant de pouvoir représenter et d'étudier des phénomènes multi-échelles et pluridisciplinaires liés aux transports des rats sur un environnement évolutif.

VIII. La modélisation multi-niveau multi-échelle

Dans cette partie, nous commençons par présenter les définitions et clarifications indispensables pour préciser les notions manipulées, avant d'illustrer l'intérêt de la modélisation multi-échelle et les limites des plateformes et moteurs de simulations multi-échelles qui existaient à l'époque où nous entamions cette thèse.

VIII.A Définitions des notions manipulées

VIII.A.1 Echelle, niveau et hiérarchie :

Dans une revue qui traite le concept d'échelle et les dimensions humaines du changement global, Gibson *et al.* [2000] proposent les définitions suivantes que nous retenons dans ce travail même si nous reviendrons plus en détail sur la définition d'*échelle* et de *niveau* :

- *Echelle* : Elle fait référence à la mesure ; c'est la dimension spatiale, temporelle, quantitative ou analytique utilisée pour mesurer et étudier tout phénomène. Une échelle a une étendue et une résolution [Marceau, 1999], bien que celles-ci puissent ne pas être clairement notées dans des études particulières :
  - l'*étendue* se réfère à la grandeur d'une dimension utilisée dans la mesure d'un phénomène. En ce qui concerne l'espace, l'étendue peut varier d'un mètre à des millions de mètres carrés ou plus. En ce qui concerne le temps, l'étendue peut être un jour, une semaine, un an, dix ans, un siècle, un millénaire, ou plusieurs millénaires. En ce qui concerne la quantité, le nombre d'individus considérés par l'observateur pour être impliqués dans une relation sociale peut varier de deux à des milliards, de même que la quantité de marchandises et les autres entités d'intérêt pour les chercheurs en sciences sociales. L'étendue d'une mesure fixe la limite extérieure du phénomène mesuré ;
  - la *résolution* fait référence à la précision utilisée dans la mesure; le terme grain ou granularité est souvent utilisé de la même manière. En ce qui concerne l'espace, les scientifiques sociaux utilisent une variété de résolutions allant de un mètre ou moins (études anthropologiques sur les activités domestiques) à des mesures grossières qui

couvrent des milliers de kilomètres (études de l'impact des traités internationaux). En ce qui concerne le temps, les scientifiques sociaux utilisent rarement une résolution de moins d'une heure pour diviser le temps. Cependant ils peuvent faire cela dans certaines situations. La résolution utilisée pour observer la quantité dépend de l'étendue en cause, par exemple, lorsqu'une analyse implique une plus grande quantité, les mesures utilisent normalement une plus grande agrégation des unités individuelles que quand une plus petite quantité est étudiée. Dans ce manuscrit, nous allons toujours utiliser le terme «petite échelle» pour faire référence à des phénomènes qui sont petits en ce qui concerne les échelles d'espace, de temps ou de quantité. Ainsi, «à grande échelle» fait référence à de grands items. Cela est bien conforme à l'usage courant de ce terme (mais est exactement le contraire de la façon dont ces termes sont utilisés par les cartographes) ;

- *Niveau* : est un point de vue sur un système, intégré dans un modèle comme une abstraction spécifique [Morvan, 2013] ;
  - Coquillard et Hill [1997] expliquent les termes *niveau d'abstraction* et *niveau de détail* :

    Niveau d'abstraction

    Le *niveau d'abstraction* dans un modèle est défini par les contraintes suivantes :
    - les objectifs à atteindre (quels types de résultats attendons-nous) ;
    - l'état des connaissances contenant le système et les données à disposition.

    Par exemple on peut étudier un système à un niveau d'abstraction correspondant à une population, à un groupe d'individus, à un individu, à un organe etc.

    Niveau de détail

    Après avoir choisi le niveau d'abstraction à laquelle on veut étudier un système, il convient de préciser quels sont les paramètres à prendre en compte : le *niveau de détail*. Par exemple on peut se poser la question à savoir à quel niveau de détail doit-on étudier un modèle de croissance forestière dont le niveau d'abstraction choisi correspond à une population ? Devra-t-on par exemple tenir compte ou non des paramètres suivants ?
    - Variation saisonnière de l'intensité lumineuse ;
    - Pression partielle en CO2 de l'atmosphère ;
    - Compétition pour les ressources en eau et nutriments du sol ;
    - Compétition pour l'espace ;
    - Etc.
  - Fishwick [1997] considère que le niveau d'abstraction est associe à la réflexion et à la description d'un modèle ;
  - Ratzé *et al.* [2007] considèrent, en écologie, le niveau d'abstraction pour la partie structurelle et le niveau de détail pour la partie fonctionnelle ;
  - Ferber et Perrot [1995] considèrent que la notion de *niveau d'organisation* permet de comprendre l'emboîtement d'un niveau dans un autre ; Une organisation étant une agrégation d'éléments de niveau inférieur, et un composant d'organisation de niveau supérieur. Par exemple en biologie une cellule peut être considérée comme une

organisation de macromolécules et comme un être individuel pour l'organisme multicellulaire dont elle fait partie ;

- *hiérarchie :* Un système lié conceptuellement ou causalement de groupes d'objets ou processus le long d'une échelle d'analyse. On peut en citer trois types :
  - *hiérarchie inclusive* : Les groupes d'objets ou de processus qui sont classés comme inférieurs dans la hiérarchie sont contenus dans les groupes ou subdivisions de groupes qui sont classés en tant que supérieurs dans le système (par exemple, les classifications taxonomiques modernes: règne, embranchement, sous-embranchement, classe, famille, genre, espèce) ;
  - *hiérarchie exclusive* : Les groupes d'objets ou de processus qui sont classés comme inférieurs dans la hiérarchie ne sont pas contenus dans les groupes ou subdivisions des groupes qui sont classés en tant que supérieurs dans le système (par exemple, les systèmes de classement militaires : général, capitaine, lieutenant, sergent, caporal) ;
  - *hiérarchie constitutive* : Les groupes d'objets ou de processus sont combinés dans de nouvelles unités qui sont ensuite combinés dans de nouvelles unités avec leurs propres fonctions et propriétés émergentes (par exemple, organe, tissu, cellules, molécule …).

VIII.A.2 Espace et temps

En ce qui concerne l'espace (géographique) et le temps, Nous retenons dans ce travail les définitions proposées par Langlois [2009] **:**

- *l'espace géographique* : L'espace E est une organisation prédéfinie capable de recevoir des points représentés par des coordonnées géométriques. L'espace E est structuré sous la forme d'un référentiel géométrique paramétrable (dimensionnalité, point origine, axes et unités, taille, morphologie). L'espace E est muni d'une structure mathématique d'espace métrique, à deux ou trois dimensions. Cette métrique permet de localiser et de définir les voisinages des points, objets et agents de l'environnement. On peut en citer *(i)* la métrique euclidienne (pour l'espace continu et pour les graphes) qui donne des coordonnées réelles et le voisinage euclidien ou disque (boule en 3D) (Figure 8 A, B), *(ii)* la métrique de Manhattan qui donne le voisinage de von Neumann sur la grille (Figure 8 C) et *(iii)* la métrique du max qui donne des coordonnées discrètes et le voisinage de Moore sur la grille elle aussi (Figure 8 D) ;

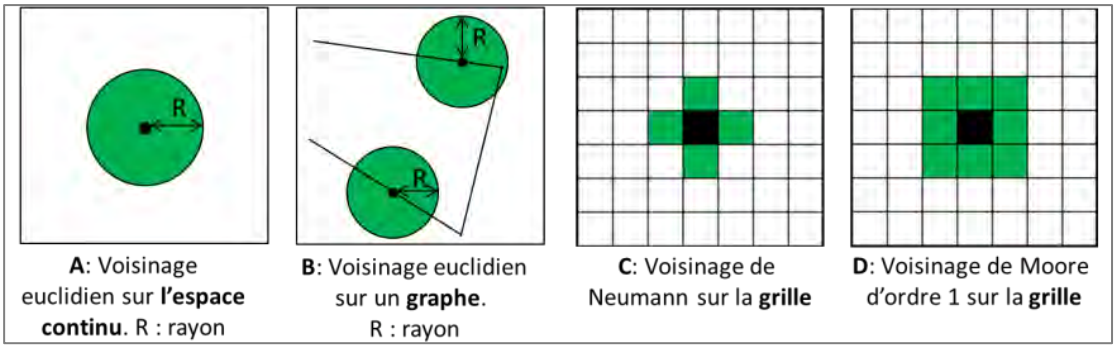

Figure 8 – Voisinages dans différentes topologies.

- *le temps* : L'espace temporel (ou plus simplement le temps) est aussi une organisation prédéfinie, qui est structurée par un référentiel temporel (ou temporalité) dans lequel s'inscrivent toutes les entités dynamiques d'un modèle. Ce référentiel temporel est dit systémique ou global, dans la mesure où il englobe et référence toute autre organisation temporelle plus spécialisée. Un référentiel temporel se définit par une origine, une unité de temps, un pas de temps élémentaire, une durée maximale et un ordonnanceur élémentaire ;
- *l'espace et le temps* : Toute modélisation géographique se situe à une certaine échelle d'espace et de temps. Cette échelle se caractérise par une extension (un domaine d'espace et de temps) et une granularité (discrétisation). L'espace géographique est constitué d'un référentiel géométrique paramétrable (nombre de dimensions, souvent 2, origine géographique, orientation des axes, souvent implicite, unités, pas élémentaires, extension du domaine et métrique). De même le temps est représenté par un référentiel monodimensionnel qui définit une temporalité (par une origine, une unité de temps, un pas de temps élémentaire, une extension).
  A un référentiel d'espace on associe un référentiel de temps qui lui est compatible en termes de discrétisation et d'extension, aussi nous parlerons de référentiel d'espace-temps.

VIII.A.3 Emergence et Auto-organisation

Nous retenons dans ce travail les définitions proposées par De Wolf et Holvoet [2005] qui expliquent la différence entre les concepts d'*émergence* et d'*auto-organisation* :

- *Emergence* : Un système exhibe une émergence quand il y a des structures inédites émergentes au niveau macro, qui proviennent dynamiquement des interactions entre les entités au niveau micro (par exemple un organe vivant est une émergence d'un ensemble de tissu. Ce dernier lui aussi a émergé d'un ensemble de cellules) ;
  Ainsi nous pouvons dire que l'émergence est liée à la hiérarchie constitutive ;

- *Auto-organisation* : c'est un processus dynamique et adaptatif où les systèmes acquièrent et maintiennent la structure eux-mêmes, sans contrôle externe. Exemple les réseaux ad-hoc qui construisent de manière autonome leur structure en tant que dispositifs de réseau de détecter la présence de l'autre [*ibid*].

VIII.B Distinction entre multi-échelle et multi-niveau

Dans la littérature, les termes *multi-niveau* et *multi-échelle* sont fréquemment utilisés de manière interchangeable et de manière différente en fonction des disciplines. Ce qui est potentiellement source de grande confusion [Quijano-Gil *et al.*, 2010]. Etant donné que ces deux termes sont aussi omniprésents dans ce manuscrit nous jugeons nécessaire, avant de continuer, de clarifier ces notions dans le contexte de cette thèse.

VIII.B.1 Multi-niveau n'est pas multi-échelle

Sans inclure la notion d'échelle, Fishwick [1997] donne la définition suivante de la simulation multi-niveau :

« *La simulation multi-niveau est un type particulier de simulation où le modèle proposé du système intègre différents niveaux d'abstraction (au moins deux) et où les outils nécessaires à son exécution permettent de faire cohabiter ces différents niveaux d'abstraction au sein d'une même exécution et d'assurer la transition dynamique entre eux en fonction de contraintes définies (dépendantes du modèle ou du cadre expérimental).* »

Selon Quijano-Gil *et al.* [2010] la notion de *niveau* fait référence à la dimension d'analyse. Elle permet de situer le phénomène étudié (niveau d'abstraction) et/ou les entités qui le composent (niveau de détail). Tandis que la notion d'*échelle* fait référence à la mesure, à la dimension spatiale ou temporelle dans lequel le phénomène étudié et/ou les entités qui le composent sont étudiés. Si l'on s'en tient en toute rigueur à cette utilisation des notions d'échelle et de niveau, beaucoup de modèles dit *multi-échelles* ne le sont pas. Il s'agit en réalité de modèles dont les entités qui les composent se situent à des *niveaux différents* le long d'une seule échelle spatiale et d'une seule échelle temporelle [Gil-Quijano *et al.*, 2009].

Dans une revue [Morvan, 2012] étendue par la suite [Morvan, 2013], Morvan soutient la position de Quijano en écrivant que le terme *multi-échelle* ou *multi-résolution* est souvent utilisé (à la place de *multi-niveau*), mais il a un sens plus restrictif car il se concentre sur les étendues spatiales et temporelles des *niveaux* et non pas sur leurs interactions et leur organisation. Il définit ainsi le terme *niveau* comme étant un point de vue sur un système, intégré dans un modèle comme une abstraction spécifique.

Après cela, Vo [2012], donne lui aussi une définition de modèle multi-niveau qui n'inclue pas la notion de *multi-échelle* en écrivant :

« *Un modèle à base d'agents multi-niveau est un modèle multi-agents dans lequel les agents sont organisés par le modélisateur en différents niveaux.* »

Soyez [2013] précise cette perception :

« *Quand on évoque différents niveaux de représentation on pense tout d'abord au multi-échelle ou multi-résolution. On peut définir une échelle comme une mesure du temps, de l'espace ou de la granularité d'un environnement. Un niveau est vu comme un concept plus général qui permet de définir un point de vue sur un système. Ainsi, différents niveaux peuvent être à la même échelle mais concernent des aspects différents d'un même système.* »

Nous pouvons ainsi dire qu'un modèle peut être multi-niveau sans être multi-échelle. C'est-à-dire que tous les niveaux peuvent être représentés et étudiés avec une seule échelle spatiale et une seule échelle temporelle. Cette unique échelle spatio-temporelle peut être :

- celle permettant, au moins, d'avoir une bonne représentation du niveau le plus élevé (voir Figure 9 A). Cependant, avec cette manière de faire, les entités des niveaux les plus bas risquent de ne pas être représentées correctement ;
- celle permettant de représenter convenablement le niveau le plus bas (voir Figure 9 B). L'inconvénient avec cette manière de faire est qu'elle ne permet pas l'économie de

ressources informatiques en ne représentant que ce qui est utile à un moment et un espace donnés ;
- une échelle intermédiaire entre les deux citées précédemment. Dans ce cas, on peut avoir, de manière un peu plus atténuée, les deux inconvénients cités ci-avant.

Ainsi, nous pouvons dire que pour un modèle multi-niveau, chaque niveau doit être représenté et étudié avec une échelle spatiale et temporelle adéquates comme sur la Figure 9 C. Ces échelles ne doivent être uniques que si les dimensions des entités représentées et/ou leur temps d'évolution caractéristique sont d'un ordre de grandeur identique ou comparable.

A l'instar de ceci, nous pouvons retenir la définition (que nous adaptons) de modèle multi-niveau que propose Morvan [2013] dans sa revue sur la modélisation multi-niveau :

> Un modèle à base d'agents multi-niveau *intègre* des modèles *hétérogènes* (à base d'agents), représentant des *points de vue (qui peuvent être complémentaires)*, dits niveaux, du même système :
> - *intégration* signifie que les modèles dans un SMA peuvent interagir et partager des entités telles que les environnements et les agents ;
> - *hétérogénéité* signifie que les modèles intégrés dans un SMA peuvent être basés sur différents paradigmes de modélisation (des équations différentielles, les automates cellulaires, etc.), utiliser différentes représentations du temps (à événements discrets, à temps discret) et représenter des processus à des échelles spatio-temporelles différentes ;
> - les *points de vue peuvent être complémentaires* pour un problème donné, c'est à dire qu'ils ne peuvent pas être pris isolément pour aborder ce problème.

VIII.B.2 Multi-échelle n'est pas multi-niveau

Comme nous l'avons déjà expliqué, l'échelle est une question de mesure (étendue et granularité). Si nous observons par exemple une forêt (comme niveau d'abstraction), alors nous pouvons compter le nombre d'arbres à l'are, à l'hectare et au km² sans changement de nature ; ou plus généralement sans changement du phénomène étudier et ou des entités qui le composent. C'est de la multi-échelle spatiale au sein d'un même niveau ; Ce comptage peut aussi se faire chaque année, chaque mois, chaque semaine etc. C'est donc de la multi-échelle temporelle sans changement de niveau.

Nous pouvons ainsi dire que l'on peut avoir un modèle multi-échelle spatiale et temporelle sans que le modèle ne soit multi-niveau. En d'autres termes, dans un modèle à base d'agent, on peut avoir plusieurs échelles spatiales et temporelles au sein d'un même niveau.

Nous pouvons ainsi retenir la définition de modèle multi-échelle que propose Vo *et al.* [2012] :

> Un modèle à base d'agents *multi-échelle* est un modèle qui exige la possibilité de représenter simultanément plusieurs *échelles* spatiales et temporelles ; L'*échelle spatiale* est représentée comme (la granularité de) l'environnement dans lequel vivent les agents ; L'*échelle temporelle* est représentée par la fréquence d'exécution des agents, qui

est commandée par un ordonnanceur (ou scheduler). En d'autres termes le modèle à base d'agents multi-échelle doit permettre la possibilité de représenter à la fois des environnements et des ordonnanceurs à des *échelles* différentes.

VIII.B.3 Modèle multi-niveau et multi-échelle

Principalement, deux situations peuvent nécessiter qu'un modèle soit à la fois multi-niveau et multi-échelle :

1. lorsque l'on étudie un phénomène et que l'on souhaite en même temps, aller *plus en détail* sur ce qui se passe à un niveau plus bas ou à l'inverse, appréhender de manière plus globale les phénomènes se déroulant à un niveau plus élevé ; en prenant en compte l'interaction entre les niveaux et l'apparition de nouvelles entités plus petites ou plus grandes qui nécessitent une nouvelle échelle pour mieux les représenter. Cette situation peut correspondre aux systèmes complexes présentant une hiérarchie de type constitutif (sans émergence de niveaux), mais aussi de type inclusif ou exclusif (voir *hiérarchie*, p. 25) ;
2. lorsque les interactions des entités à un niveau plus bas, étudiées avec une échelle plus petite, font apparaitre des structures ou des motifs qui sont reconnus par le modélisateur comme d'autres entités plus grandes appartenant à un niveau plus élevé, et qui nécessitent une deuxième échelle plus grande afin *(i)* d'économiser les ressources informatiques utilisées et *(ii)* de les représenter convenablement. Cette situation correspond aux systèmes complexes présentant une hiérarchie de type constitutif (avec émergence de niveau).

Partant de ceci, nous proposons la définition suivante d'un modèle à base d'agents multi-niveau et multi-échelle :

> C'est un modèle qui *intègre* simultanément plusieurs sous-modèles (au moins deux) qui peuvent être *hétérogènes* (à base d'agents), représentant des *points de vue qui peuvent être complémentaires*, dits niveaux, du même système. Chaque niveau ayant ses propres échelles spatiales et temporelles (au moins une) pour bien représenter (au sein du niveau) les phénomènes étudiés et/ou les entités qui le composent.

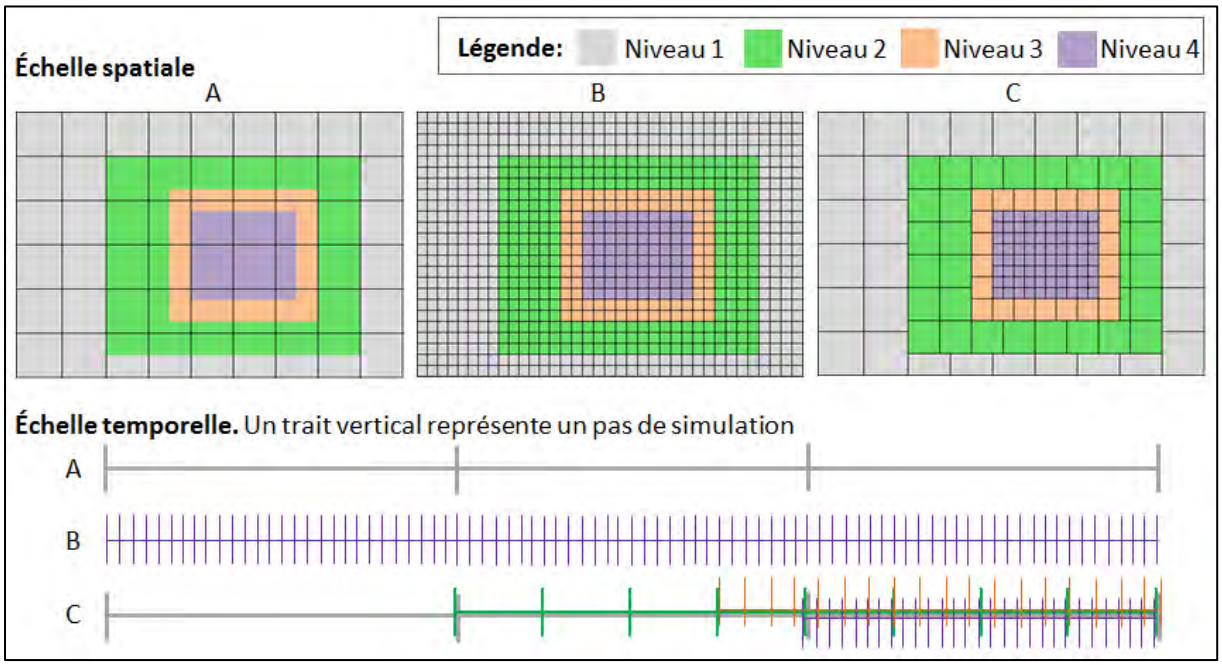

Figure 9 – Multi-niveau et multi-échelle.
Les niveaux sont emboités. A et B : multi-niveau avec une seule échelle (spatiale et temporelle) pour représenter tous les niveaux : cette échelle est celle permettant, au moins, de représenter convenablement le niveau le plus élevé pour A et le niveau le plus bas pour B. C : chaque niveau est représenté avec une échelle qui lui est propre : situation de multi-niveau et de multi-échelle.

Nous pouvons à présent, après avoir défini certains termes, présenter quelques utilités de la modélisation multi-niveau et multi-échelle. Ce qui fait l'objet de la section suivante.

VIII.C Nécessité d'une modélisation multi-niveau et multi-échelle dans le cadre de ce travail

Les systèmes complexes naturels et sociaux sont souvent composés de différents niveaux d'organisation. En effet ces systèmes sont souvent caractérisés par des entités hétérogènes de nature et de dimensions diverses. Ces entités interagissant les unes avec les autres, à des niveaux d'échelles spatiales et temporelles qui peuvent être eux aussi très divers. Ainsi une des raisons qui peuvent motiver le développement de modèles à base d'agents multi-niveaux et multi-échelles peut provenir des problématiques en systèmes complexes sociaux (comme dans le cadre de ce travail) correspondant au point 1 de la section précédente (VIII.B.3).. En effet, dans le cadre des sciences sociales, la géographie en particulier, on peut par exemple vouloir étudier globalement un phénomène au niveau nationale (prise en compte des régions et des liaisons entre elles), plus en détail au niveau régionale (prise en compte des villes et des liaisons entre elles) et encore plus en détail au niveau d'une ville, d'un quartier etc. avec des agents pouvant aller d'un niveau à un autre (ou plus généralement les niveaux pouvant s'alimenter mutuellement).

Par exemple une des problématiques de recherche du Projet CHANCIRA (voir annexe A) est d'étudier les épidémies causées par le rat noir dans la ville de Kédougou. Mais avant cela, les chercheurs de ce projet veulent comprendre comment le rat noir est arrivé à Kédougou même. C'est-à-dire, supposant que le rat noir n'était présent que dans les comptoirs coloniaux au Sénégal en 1900 [Konečnỳ *et al.*, 2013], comment ils sont arrivés aux régions de Tambacounda et de Kédougou, ensuite à la ville de Kédougou et dans ses quartiers avant de causer des épidémies. Pour ce faire nous voulons simuler en cascade et d'une manière emboitée (s'alimentant mutuellement) des modèles permettant de représenter les trois phases suivantes (Figure 10 et Figure 11) :

- **phase 1** : la diffusion du rat noir, par le transport humain, des comptoirs coloniaux aux différentes régions du Sénégal (niveau d'abstraction national-centennal), sur l'espace sénégalais (étendue spatiale) discrétisé en une grille composée de cellules de 7,5 km de côté (échelle spatiale), avec une durée de simulation d'1 siècle (1910 – 2010, étendue temporelle) et un pas de temps d'1 jour (échelle temporelle, voir la partie en gris sur la Figure 10 et sur la Figure 11) ;
- **phase 2** : la diffusion du rat noir, par le transport humain à l'intérieur des régions de Tambacounda et de Kédougou (des régions aux villes, niveau régional-décennal), sur un environnement discrétisé en des cellules de 2 km de côté, avec une durée de simulation de 10 ans (2000 - 2010) et un pas de temps de 1 heure (voir la partie en bleu sur la Figure 10 et sur la Figure 11);
- **phase 3** : la propagation du rat noir à l'intérieur de la ville de Kédougou (des villes aux quartiers, niveau urbain-mensuel), sur un environnement discrétisé en des cellules de 38 m de côté, avec une durée de simulation de 1 mois (en 2010) et un pas de temps de 5 minutes (voir la partie en rouge sur la Figure 10 et sur la Figure 11). Cette phase doit aussi permettre de s'intéresser aux modalités de transmission des zoonoses par le rat à l'homme via les moustiques (probablement) et en fonction de la nature de l'environnement urbain.

La simulation doit prendre en compte des transporteurs pouvant transporter avec eux des rats d'un niveau à un autre.

Nous pouvons ainsi dire que cette problématique présente trois niveaux d'abstraction (national, régional et urbain) avec des niveaux de détails différents (composés respectivement de régions, de villes et de quartiers, maisons etc.) et des échelles spatiales et temporelles différentes (d'un niveau à un autre). En conséquence, cette problématique nécessite une modélisation et une simulation multi-niveau et multi-échelle pour être représentée et étudiée correctement.

En plus de la possibilité qu'elle offre de représenter et d'étudier correctement des phénomènes hiérarchisés, la simulation multi-échelle a principalement deux avantages : *1)* la possibilité de "zoomer" ou "dézoomer" pendant l'exécution de la simulation pour observer plus en détail ce qui se passe à une plus petite échelle ou au contraire appréhender, de manière plus globale, ce qui se passe à une échelle plus grande; *2)* l'économie de ressources

informatiques en ne représentant que ce qui est utile à un moment et un espace donnés [Soyez, 2013].

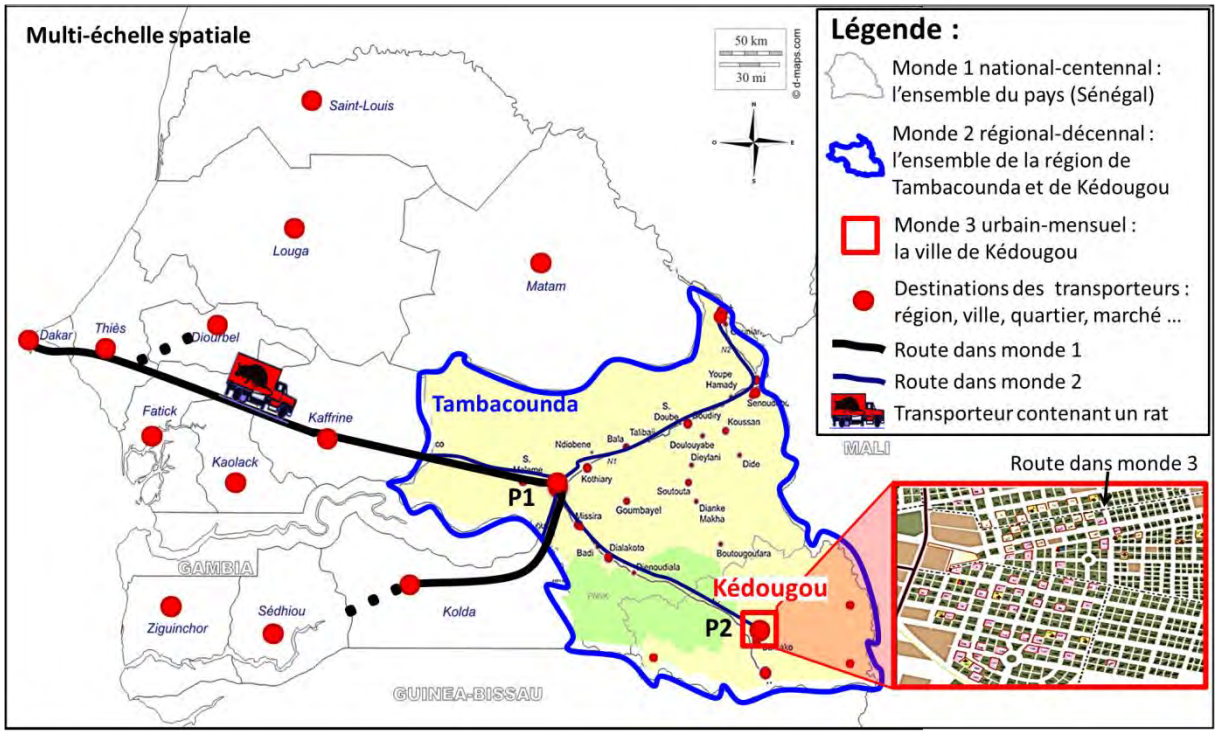

Figure 10 – Présentation de trois échelles spatiales emboitées.
Une première phase concerne le transport du rat noir des comptoirs coloniaux aux différentes régions du Sénégal (concerne le territoire *national*) ; une deuxième phase concerne la difusion des rats à l'intérieur des régions de Tambacounda et de Kédougou (dans les villes de ces deux régions) et une troisième phase concerne la propagation du rat noir à l'intérieur de la ville de Kédougou. Les trois phases sont spatialement emboitées et s'alimentent mutuellement.

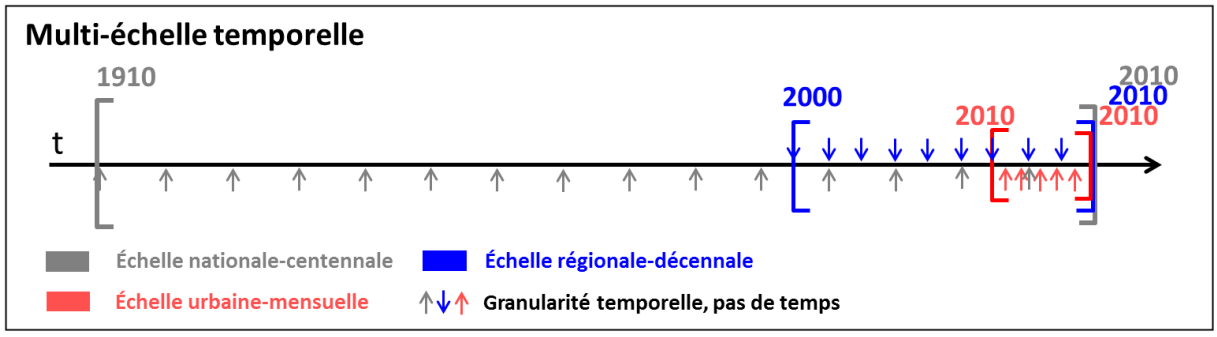

Figure 11 – Présentation de trois échelles temporelles emboitées.
Une première phase, nationale-*centennale* commence en 1910 pour une durée de 100 ans (crochets et années gris), avec des pas de temps de 1 jour (flèches grises) ; une deuxième phase régionale-*décennale* commence en 2000 pour une durée de 10 ans (crochets et années bleus), avec des pas de temps de 1 heure (flèches bleues) et une trosième phase urbaine-*mensuelle* commence en 2010 pour une durée de 1 mois (crochets et années rouges) avec des

pas de temps de 5 minutes (flèches rouges). Toutes les 3 phases se terminent en 2010. Elles sont temporellement en cascade et emboitées.

VIII.D Plateformes et moteurs de simulations multi-niveau et multi-échelles

Morvan [2012; 2013] a recensé les meta-modèles et les moteurs de simulation dédiés à la modélisation à base d'agents multi-niveau qui étaient proposés dans la littérature. Dans cette section nous rappelons celles qui traitent de la modélisation multi-échelle et celles qui sont apparues après.

ML-DEVS [Steiniger *et al.*, 2012; Uhrmacher *et al.*, 2007] est une extension de DEVS [Duboz, 2004; Duboz *et al.*, 2003; Duhail, 2013; Müller, 2009; Zeigler *et al.*, 2000] qui permet la simulation de modèles multi-échelles (et pas seulement les modèles couplés dans lequel le comportement d'un modèle est déterminé par les comportements de ses sous-modèles). Deux types de relation entre les niveaux sont définis dans *ML-DEVS* que sont la propagation d'informations et l'activation d'événements.

CRIO [Gaud, 2007; Gaud *et al.*, 2008a, 2008b] (Capacité Rôle Interaction Organisation) est un meta-modèle organisationnel dédié à la modélisation à base d'agents multi-niveau basé sur le concept d'Holon [Koestler, 1967, 1978]. Il a été utilisé pour développer des simulations multi-échelles de flux de piétons [Morvan, 2013].

SPARK[10] [Solovyev *et al.*, 2010] (Plate-forme Simple pour la Représentation à Base d'Agents de la Connaissance) est un framework de modélisation multi-agents multi-échelle, dédié à la recherche biomédicale.

ML-Rules [Maus *et al.*, 2011] est un langage de modélisation multi-échelle à base de règles dédié aux systèmes de biologie cellulaire. Les règles décrivant la dynamique du système s'écrivent de manière similaire aux équations de réaction chimique. Cette approche ne fait pas explicitement référence aux SMA; Cependant, les langues à bases de règles multi-niveaux semblent un moyen prometteur pour concevoir des modèles complexes à base d'individus [Morvan, 2013].

PADAWAN [Picault et Mathieu, 2011] ("Pattern for Accurate Design of Agent Worlds in Agent Nests") est un meta-modèle de SMA multi-échelle basé sur une représentation matricielle compacte des interactions, conduisant à un framework de simulation simple et élégant. Cette représentation est basée sur le meta-modèle *IODA* ("Interaction Oriented Design of Agent simulations") dédié au SMA classique (un niveau) [Kubera *et al.*, 2008].

Huraux *et al.* [2014a] ont mis en place un modèle pour la simulation multi-niveau en faisant coexister plusieurs niveaux sur différents axes. Ce modèle du nom de *SIMLAB* étend le simulateur *SMACH* qui sert à aider les énergéticiens à analyser l'activité des foyers en rapport avec la consommation [Amouroux *et al.*, 2013; Haradji *et al.*, 2012]. *SIMLAB* [Huraux *et al.*, 2014b] repose sur trois concepts : Premièrement, les caractéristiques des entités modélisées

[10] http://www.pitt.edu/~cirm/spark/ dernière visite 09/12/2015

sont partagées par des agents de différents niveaux, en introduisant la notion d'axe de modélisation qui capture la représentation de composantes transverses aux différents niveaux. Deuxièmement, faire la distinction entre les interactions des agents (*i.e.* l'échange d'informations ou requêtes) et les influences intra-axes des propriétés, capturant les dynamiques multi-niveaux. Enfin, pour avoir une organisation dynamique dans le SMA, proposer l'utilisation d'observations pour détecter et réifier les macro-entités qui ont du sens pour le modélisateur.

Maudet *et al.* [2014] qui ont utilisé le modèle *PADAWAN* expliquent qu'il est souvent nécessaire de raisonner à différents niveaux d'abstraction pour les problèmes de généralisation cartographique dont l'objectif est de simplifier les données géographiques afin de créer des cartes lisibles lorsque l'échelle diminue. Les auteurs ont montré comment ils ont utilisé le modèle *PADAWAN* pour réifier les relations multi-niveaux entre les agents cartographiques d'une part, et d'autre part pour représenter les contraintes et les mesures proposées pour les résoudre, comme les interactions entre les agents.

Abouaissa *et al.* [2014] ont développé un simulateur multi-agents multi-niveau de flux de trafic routier qu'ils ont nommé *JAM-FREE* qui repose sur le framework de modélisation et de simulation multi-agents multi-niveau nommé *SIMILAR* (SImulations with MultI-Level Agents and Reactions) [Morvan et Kubera, 2014], anciennement appelé *IRM4MLS* [Morvan *et al.*, 2011; Morvan et Jolly, 2012; Soyez *et al.*, 2012]. Les auteurs expliquent que *JAM-FREE* permet la simulation multi-agents multi-niveau de réseaux routiers de grande taille de manière efficace en adaptant dynamiquement le niveau de détail et de tester de nouveaux algorithmes de régulation, d'observation et routage.

Même si la notion de niveau n'apparaît pas explicitement, l'équipe de *GAMA* ("Gis & Agent-based Modelling Architecture") [Drogoul *et al.*, 2013; Vo *et al.*, 2012] a proposé un meta-modèle permettant de faire du multi-niveau et de la multi-échelle. Le meta-modèle soutient une représentation *"récursive"* des notions fondamentales de la modélisation à base d'agents (agent / environnement / scheduler). C'est-à-dire chaque agent peut être un modèle avec son scheduler et sa topologie (environnement). En effet les agents sont les objets de la classe "Species" (espèce) ; un concept définissant la structure d'un modèle avec des attributs et des comportements d'une classe d'un même type d'agent, ce qui permet l'imbrication des espèces les unes dans les autres comme l'explique Morvan [2013]. Le meta-modèle a un agent spécial, qui joue le rôle d'un agent de la racine (qu'ils peuvent nommer par agent global, agent de monde, agent du système ou un autre nom qui convient au modélisateur). Cet agent racine offre un environnement global dans lequel ses micro-agents peuvent vivre. Il offre également un scheduler pour exécuter ses micro-agents avec les bonnes fréquences. Dans ce meta-modèle, la dynamique du changement d'échelle et le comportement des agents au cours de la simulation est modélisée par la mise en place de deux opérations *"capture"* et *"libération"*. [Vo *et al.*, 2012]. Ce qui permet aux agents d'être agrégés dans des groupes existants en modifiant leurs attributs et comportements mais ne générant pas de nouvelles structures inédites [Soyez, 2013].

GAMA étant la plateforme la plus avancée de modélisation multi-agents multi-niveau (voir la section VII.B.3.a), nous nous sommes alors intéressés de plus près à sa méthodologie pour voir s'il correspond réellement à nos attentes.

VIII.E Méthodologies dans GAMA

Pour bâtir l'architecture de son meta-modèle multi-niveau couvrant le plus largement possible les exigences des modélisateurs pour le développement de modèles multi-niveaux, Vo [2012] a identifié les quatre facteurs suivants :

1. **changement dynamique de comportement** : Représenter un comportement des agents qui peut changer dynamiquement durant la simulation ;
2. **agent récursif** (le concept d'agent récursif ou composé) : Un agent peut contenir d'autres agents et être contenu dans un autre agent. L'architecture doit permettre *i*) à un agent d'accéder facilement à l'agent qui le contient; *ii*) à un agent d'accéder facilement à tous les agents qu'il contient (agents de membres); *iii*) qu'un agent puisse devenir membre d'un autre agent et cesser d'agir en tant que membre de l'ancien agent qui le contenait (ceci grâce aux opérations "capture" et "libération") ;
3. **interactions inter-niveaux**: cette exigence peut être généralisée comme "Faciliter la modélisation des interactions (bottom-up et top-down influences) entre les agents de différents niveaux" ;
4. **l'accès aux données inter-niveau**: cette exigence peut être généralisée ou formalisée comme "Faciliter la modélisation de la circulation des données entre les agents de différents niveaux. Un agent à un niveau doit être en mesure d'accéder facilement à des données d'agents de niveau supérieur et d'agents de niveau inférieur".

Ces exigences correspondent à celles de six modèles multi-agents multi-niveaux (voir le tableau ci-dessous) appartenant à différents domaines d'application [Vo, 2012] que sont :

1) **Tsunami évacuation** : évacuation de la ville de Nha Trang en cas de tsunami [Tsunami, 2004] ;
2) **BPH** (Brown Plant Hopper) : permettant d'évaluer les politiques de lutte contre les invasions de cicadelles brunes dans le delta du Mékong [Dyck *et al.*, 1979] ;
3) **RIVAGE** : dont l'objectif est de simuler les processus de ruissellement et infiltration [Servat *et al.*, 1998a] ;
4) **Avascular** : qui vise à reproduire le développement d'une tumeur avasculaire [Lepagnot et Hutzler, 2009] ;
5) **SimulBogota** : qui vise à reproduire l'évolution de la répartition spatiale de la population de la ville Bogota à travers une période de plusieurs dizaines d'années [Gil-Quijano *et al.*, 2012; Quijano-Gil *et al.*, 2010] ;
6) **Vortexes** : pour la simulation d'écoulement du fluide [Tranouez, 2005; Tranouez *et al.*, 2006].

Ces exigences, sauf celles correspondant aux agents récursifs, correspondent aussi à celles de notre modèle dont l'objectif est l'étude de la diffusion du rat noir au Sénégal

simultanément sur plusieurs échelles d'espace et de temps. La seule différence est la manière dont ces exigences sont modélisées et implémentées.

Exigence/Modèle	Changement dynamique de comportement	Agent récursif	Interactions inter-niveaux	L'accès aux données inter-niveau
Tsunami évacuation	Oui	Oui		
BPH		Oui	Oui	Oui
RIVAGE	Oui	Oui	Oui	
Avascular	Oui	Oui		
SimulBogota		Oui	Oui	
Vortexes		Oui		
SimMasto	Oui		Oui	Oui

VIII.F Discussion sur la méthodologie dans GAMA

VIII.F.1 Méthode agent récursif

Par rapport à notre problématique, la méthode agent récursif [Gil-Quijano *et al.*, 2012; Lepagnot et Hutzler, 2009; Servat *et al.*, 1998b, 1998a; Tranouez *et al.*, 2006; Vo *et al.*, 2012] n'est applicable qu'à l'environnement. En effet l'espace sénégalais est discrétisé en des cellules représentant une topologie de type grille. Donc en considérant les cellules comme des agents récursifs, chaque cellule de l'environnement pourra contenir elle aussi une grille de cellules plus fines (diminuer l'échelle spatiale). Si on prend par exemple la grille de la Figure 12.1.a composée de 16 grandes cellules (4x4) et que l'on veuille avoir un nouveau niveau de détail plus fin dans la zone en jaune. Alors en utilisant les agents récursifs, au lieu d'une seule nouvelle grille représentant cette partie, on en aurait 4 *dissociées* (Figure 12.1.b).

Il faudrait alors mettre en place des procédures pour savoir, par exemple, que les cellules A, B, C et D sont voisines les unes des autres (voisinage de Moore, Figure 12.1.b). Car par rapport aux grilles dissociées, aucune des cellules A, B, C et D n'est voisine de l'autre (ce n'est pas sur la même grille). Ces procédures rendraient plus complexe la perception des agents de leur voisinage. Ce problème serait encore plus compliqué si la zone où l'on veut entrer plus en détail peut ne pas concerner des cellules entières du niveau supérieur (la partie en jaune de la Figure 12.2.a).

Au lieu de mettre en place des procédures qui peuvent être compliquées pour coupler, dans un niveau, les grilles dissociées à cause de la méthode agent récursif, une alternative serait d'avoir un deuxième modèle (sous-modèle ou niveau) pour la partie où l'on veut entrer plus en détail. Ce modèle n'ayant qu'une seule grille avec la granularité (échelle spatiale) que l'on veut (Figure 12.1.c et Figure 12.2.c). Il faudra ensuite (comme dans tous les cas, méthode agent récursif ou pas) mettre en place des moyens de communications entre niveau. Si nécessaire, on peut aussi mettre des procédures simples pour bien positionner les

environnements des différents niveaux par rapport à un repère absolu (Figure 13) (voir partie B.2, p. 70).

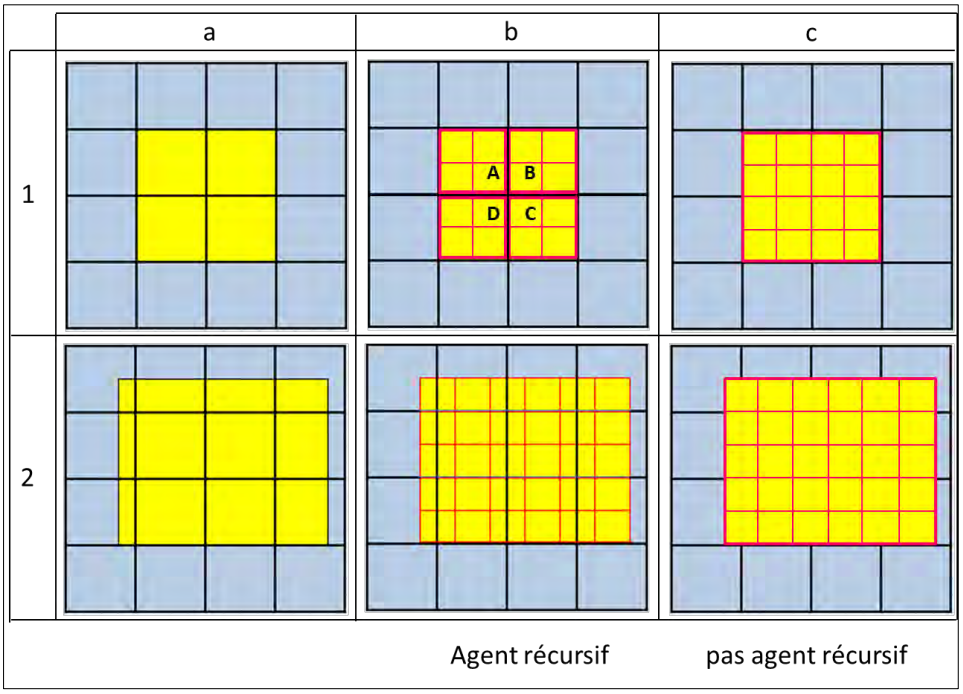

Figure 12 – La méthode agent récursif appliquée à l'environnement.
Avec cette méthode l'environnement à un niveau peut être composé de plusieurs grilles dissociées (1, b) au lieu d'une (1, c). Il faut ainsi mettre en place des procédures pour les coupler. Ces procédures peuvent être compliquées surtout avec la situation sur la partie 2 où la zone où l'on veut entrer plus en détail ne concerne pas que des cellules entières.

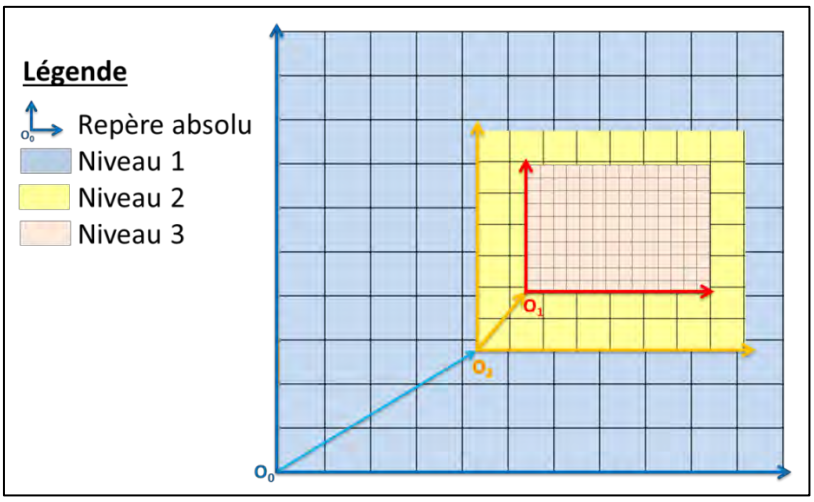

Figure 13 – Multi-échelle sans l'utilisation de la méthode agent récursif.
Chaque niveau a sa propre unique grille, contrairement à ce que l'on a avec la méthode agent récursif. Si nécessaire, on peut mettre des procédures simples pour bien positionner les grilles des différents modèles (niveaux) par rapport à un repère absolu.

VIII.F.2 Passage dynamique entre niveaux

Pour modéliser la dynamique de changement de niveau et/ou d'échelle et le comportement des agents au cours de la simulation, Vo *et al.* [2012] utilise deux opérations "capture" et "libération" ; d'autres noms ont été utilisés dans la littérature pour désigner des concepts similaires, y compris "émergence", "création de groupe" [Servat *et al.*, 1998b, 1998a], "agrégation/désagrégation" [Navarro *et al.*, 2011, 2013; Soyez, 2013] permettant à des individus d'un niveau de s'agréger ou de se désagréger vers un autre niveau (passage dynamique entre niveaux). En ce qui concerne les opérations "capture" et "libération" :

- un agent peut transformer des agents d'autres espèces en ses micro-agents en les capturant. Pour cela le modélisateur doit spécifier l'espèce ("species") que deviendra l'agent capturé et/ou si l'agent sera spatialement et temporellement capturé. C'est-à-dire si l'agent va évoluer ou pas dans l'environnement de l'agent capturant et si son exécution sera contrôlée ou pas par le scheduler de l'agent capturant ;
- un agent peut décider de libérer de manière appropriée certains de ses micro-agents. Lors de la libération, les agents libérés prennent comme macro-agent celui de l'agent libérant et récupèrent leur ancienne espèce qu'ils avaient juste avant d'être capturés par le macro-agent actuel.

Ces manières de faire permettent l'agrégation/désagrégation des agents pour passer dynamiquement d'un niveau à un autre. Cependant, elles ont été développées principalement pour la modélisation multi-niveau de systèmes représentant des hiérarchies constitutives avec des groupes d'agents ou de processus pouvant être combiné en une nouvelle unité avec des fonctions et propriétés émergentes. Ce qui ne correspond pas à notre configuration. Par rapport à la problématique de notre cas d'étudier, notre modèle a besoin de modéliser des agents transporteurs avec une liste de destinations possibles. Cette liste pourra contenir des destinations appartenant au monde dans lequel il est en train d'évoluer, mais aussi appartenant à d'autres mondes avec des échelles spatiales et temporelles différentes. Au lieu d'être capturé ou libéré par un monde (encore moins de s'agréger ou se désagréger), un agent transporteur lui-même doit être capable de choisir comme destination un endroit appartenant à un autre monde, se construire un itinéraire complet et précis et prendre le temps qu'il faut pour y aller (voir partie B.2, p. 70).

VIII.G Modélisation du temps

Pour modélisation la multi-échelle temporelle, Vo [2012] a introduit la notion de "Scheduling Information" permettant d'attacher une granularité temporelle à chaque type d'agent ("species"). Ce qui permet d'ordonnancer de manière non uniforme l'exécution d'agents appartenant à des populations différentes (une population est une "instanciation" d'un "species" dans la simulation et est responsable de la gestion des agents de ce "species"). Cet ordonnancement, pour un pas de simulation, se fait globalement en 3 étapes :

- en s'appuyant sur les informations temporelles associées à chaque "species", construire la liste de tous les agents qui doivent s'exécuter à partir de toutes les populations ;

- ensuite exécuter l'unité d'activité des agents de cette liste (ordonnancée par un unique ordonnanceur appelé "global scheduler", même si dans Vo *et al.* [2012] chaque agent possède aussi un scheduler) ;
- enfin mettre à jour les données de sorties et incrémenter l'horloge du simulateur.

Cette manière de faire me semble très pertinente et peut permettre, avec notre approche multi-monde, de prendre en compte au sein d'un monde l'exécution non uniforme d'unité d'activité de durées diverses des agents. Cependant, les auteurs n'ont pas décrit dans le détail comment ils ont procédé surtout pour la première étape avec la construction de la liste des agents qui doivent s'exécuter à un pas de simulation donné.

Contrairement à la modélisation à pas de temps discrets que nous adoptons dans ce travail, comme l'ont fait la plupart des plateformes de simulation multi-agent telles que NetLogo [Wilensky, 1999b], GAMA [Taillandier *et al.*, 2012], Repast [Collier *et al.*, 2003] et Cormas [Bousquet *et al.*, 1998], d'autre ont préféré la modélisation à événement discrets [Banks *et al.*, 1999, 2005; Guo et Tay, 2008; Müller, 2004, 2007; Steiniger *et al.*, 2012]. Parmi eux, Guo et Tay [2008] montrent la nécessité de modéliser la prise en compte d'échelles de temps hétérogènes dans les simulations à base d'agents et dénoncent le fait de n'avoir qu'une unique granularité temporelle pour tous les agents. Ils défendent ensuite, pour la prise en compte de l'hétérogénéité de la granularité temporelle des agents, l'utiliser de l'ordonnancement à événements discrets au lieu de l'ordonnancement uniforme à pas de temps discrets. Ils ont ainsi proposé un procédé assez détaillé pour cet ordonnancement en adaptant l'algorithme de Banks *et al.* (1999) sur l'ordonnancement de l'exécution des agents dans les simulations à événements discrets. Les auteurs ont enfin, appliqué leur approche aux systèmes complexes naturels (biologie) qui présente des hiérarchies constitutives et de la récursivité (agents composés d'autres agents), même s'ils n'ont pas utilisé ces termes.

VIII.H Conclusion

GAMA était la plateforme la plus avancée que nous avions trouvée, à l'époque où nous commencions cette thèse, être la plus proche comme outil de modélisation multi-agents et multi-échelle pour notre problématique. Cependant elle présentait quelques limites par rapport à notre problématique telles que :

- la représentation des niveaux avec la méthode agent *récursif* qui correspond surtout aux systèmes complexes de hiérarchie constitutive avec des regroupements d'agents pour former de nouvelles unités appartenant à d'autres niveaux. Ce qui ne correspond pas à notre problématique (voir VIII.C) ;
- le passage des agents d'un niveau à un autre avec les méthodes *capture* et *libération*. En effet dans notre modèle c'est l'agent qui doit choisir d'aller d'un niveau à un autre (voir VIII.C).

En plus de cela, deux des objectifs de cette thèse étaient *1)* de réutiliser les outils préexistants sur la modélisation des rats de la plateforme SimMasto (voir VI.B.3.b) afin d'éprouver sa robustesse et *2)* d'y ajouter des modules lui conférant la prise en compte de la

topologie de type graphe, l'approche orientée connaissance-événement et la modélisation multi-échelle.

Nous réitérons donc notre proposition qui consiste à mettre en place et à implémenter *i)* un meta-modèle multi-agents multi-niveau composé de sous-modèles (mondes) d'échelles spatiales et temporelles différentes (multi-monde et multi-échelle), *ii)* chaque monde pouvant être alimenté, sous forme d'événement, par des données provenant de la connaissance pluridisciplinaire (approche orientée connaissance-événement), *iii)* les agents pouvant délibérément choisir d'aller d'un monde à l'autre en y transférant de nouveaux éléments ou de l'activité (modèles emboités, qui s'alimentent mutuellement), et enfin, *iv)* la simulation d'une phase, avec un monde, permettant de déterminer certains éléments de la situation initiale pour la phase suivante simulée avec un autre monde (modèles en cascade). Ce que nous abordons dans la partie suivante.

Mais avant cela, nous précisons que, pour des raisons de simplicité, nous allons dans la suite de ce document, utiliser les termes "multi-monde" et "multi-échelle" au lieu du terme "multi-niveau et multi-échelle". En effet, dans le modèle que nous allons proposer, chaque niveau correspond à un monde qui correspond à une échelle. Ainsi, nous pouvons aussi utiliser les termes "échelle" ou "monde" à la place du terme "niveau".

Partie B
Modélisation/implémentation

Appréhender les risques d'émergence des anthropozoonoses liés à la diffusion du rat noir (cf. I, p. 1) est un objectif multi-échelle. La démarche retenue pour aborder le problème a consisté tout d'abord à scinder le travail de modélisation en trois sous-modèles (mondes) d'échelle spatiale et temporelle différentes (partie B.1, p. 42), avant de les emboiter dans un même modèle multi-monde et multi-échelle (partie B.2, p. 70).

B.1 : Modèles mono-échelles autonomes

I. Présentation

Dans cette partie nous divisons donc le travail de modélisation en trois phases d'échelles spatiales et temporelles différentes :

Phase 1 : échelle nationale-centennale : Cette première modélisation porte sur la colonisation à l'échelle du territoire sénégalais sur un siècle. Dans cette phase, l'accent est mis sur le transport des rats par les véhicules circulant dans le territoire ainsi que l'évolution des voies de circulation et des villes sur le long terme. Le modèle doit prendre en compte, à partir des connaissances expertes, les apports de rats depuis leurs foyers historiques et leur transport par route, voies navigables et rails via divers types de véhicules. Cette phase doit particulièrement permettre la calibration du processus de transport, essentiellement la probabilité de montée et descente des différents types de véhicules et celle de la dynamique de population de rats.

La simulation pour cette phase se tient sur une étendue temporelle d'un siècle (1910 - 2010) avec des pas de temps de 1 jour et sur l'étendue spatiale du territoire sénégalais digitalisé en des cellules de 7,5 km de côté.

Phase 2 : échelle régionale-décennale : Dans un deuxième temps et à partir des données produites lors de la première phase, on procède à la modélisation de la diffusion à l'échelle régionale sur les vingt dernières années et l'effet de modifications de l'environnement sur ce processus. Le cas d'étude retenu sur cette phase est l'impact de la construction d'une route bitumée entre Tambacounda et Kédougou à la fin des années 1990. La présence avérée de rats noirs à Kédougou depuis ce changement alors que cette ville n'en contenait pas constitue le point focal pour la calibration et la validation du modèle. Cette phase doit particulièrement permettre la représentation des communications (dont les flux d'entrée/sortie de la région) et des contextes sociaux et commerciaux ainsi que la calibration du modèle avec les données spécifiques récoltées par le projet CHANCIRA dans lequel s'est inséré ce travail.

La simulation pour cette phase se tient sur une vingtaine d'années (1990 - 2010) avec des pas de temps de 1 heure et sur un espace digitalisé en des cellules de 2 km de côté.

Phase 3 : échelle urbaine-mensuelle (échelle contact) : Enfin, à partir des données produites lors de la deuxième phase et à une échelle spatiale et temporelle locale, on aborde la modélisation de la propagation du rat noir au sein même des systèmes urbains en fonction des paramètres sociaux-environnementaux et de la configuration des espaces (gares routières, marchés, types d'habitation, connectivité, etc.). La ville de Kédougou constitue le cas type sur

lequel la modélisation des données disponibles est réalisée. Cette phase doit permettre de s'intéresser aux modalités de transmission des zoonoses par le rat à l'homme en fonction de la nature de l'environnement urbain.

La simulation pour cette phase se tient sur 1 mois avec des pas de temps de 5 minutes et sur un espace digitalisé en des cellules de 38 m de côté. D'où le nom d'*échelle urbaine-mensuelle*.

Il faut remarquer que les 3 phases représentent quasiment les mêmes phénomènes. Ce sont surtout les échelles spatiales et temporelles qui changent d'une phase à l'autre. En conséquence nous allons aborder le travail de modélisation de telle sorte qu'en faisant un modèle (ou sous-modèle), il y aura beaucoup d'éléments déjà développés qui pourrons être transposés au modèle suivant. Ce qui conduit à un effort de modélisation de plus en plus petit d'un modèle à un autre. Cependant rien n'empêche que l'on puisse représenter dans un sous-modèle des entités et des activités qui ne sont pas représentées dans d'autres sous-modèles. En effet les sous-modèles que nous allons représenter ici peuvent être considérés comme des modèles classiques complets, autonomes, qui peuvent fonctionner seuls. On les appelle des *mondes*.

L'objectif final de cette partie est d'emboiter les 3 *mondes* dans une même simulation et que ces *mondes* communiquent (passage d'agents d'un monde à un autre etc.). Ainsi le modèle de chaque monde doit prévoir le fait que les agents qu'il contient puissent s'adapter à un éventuel changement d'échelle spatiale et temporelle. Ceci sera donc pris en compte dans le modèle de la phase 1 (nationale-centennale) et sera transposé aux autres modèles.

Pour ce qui suit, nous retenons les appellations suivantes :

- Modèle *centennal* ou *monde 1* pour la phase 1 ;
- Modèle *décennal* ou *monde 2* pour la phase 2 ;
- Modèle *contact* ou *monde 3* pour la phase 3.

II. Modèle Centennal

Dans cette section, nous présentons la construction d'un modèle de monde permettant : *(i)* la représentation et l'évolution dans le temps de rats simulés qui peuvent monter et descendre de divers types de moyen de transports (bateaux, camions, trains) qui circulent sur divers types de voies (fleuves, rail, routes) reliant les villes du Sénégal. Avec les transporteurs qui choisissent aléatoirement leurs destinations à partir d'alternatives pondérées par la taille de la population des villes ; *(ii)* de passer des connaissances à la simulation multi-agents. C'est-à-dire permettant la prise en compte de l'évolution sur un siècle (des villes, des voies de transport, des zones de commerce, …) par l'utilisation et l'adaptation de l'approche orientée événement (cf. A.IV). Ce qui permet au simulateur de recevoir comme entrées les données de diverses thématiques (biologie, géographie, histoire…) organisées sous forme de chronogramme ; et *(iii)* permettant aux agents du monde de pouvoir s'adapter à un éventuel changement d'échelle spatiale et temporelle qui pourrait survenir en cours de simulation. Ce travail a fait l'objet de deux publications [Le Fur *et al.*, 2016b; Mboup *et al.*, 2015a].

II.A Modélisation

II.A.1 Architecture du modèle

Nous avons ainsi intégré dans le modèle de SimMasto les classes encadrées en rouge de l'architecture présentée sur la Figure 14. Une bonne partie de cette architecture étant déjà présentée (voir Le projet SimMasto), nous ne présenterons donc ici que les parties qui concernent directement le modèle de monde *centennal* :

C_HumanCarrier est une classe qui hérite de *A_Animal* et permet de créer des agents transporteurs. Un transporteur possède un véhicule de la classe *C_Vehicle* qui est un conteneur mobile et donc peut contenir des agents tels que son propriétaire et des rats (voir la section II.A.6). *C_Vehicle* peut prendre plusieurs types possibles (camion, train, bateau etc.) et chaque type dispose d'une vitesse et d'un type de voie (route, rail, fleuve) sur lequel il évolue. Ces informations sont sauvegardées dans des constantes (voir conversion de vitesse dans la section II.A.3.d). Le déplacement des véhicules s'effectue donc sur les topologies de type graphe dont la gestion est prise en compte grâce aux classes *C_SoilCellGraphed*, *C_LandscapeNetwork* et *C_Graph* que nous étudions à la section II.A.2.

Les inspecteurs (*A_Inspector*) se chargent de stocker et de restituer au besoin les données de la simulation (voir VII.B.3.b). Nous avons ainsi un inspecteur *centennal* pour cette étude de cas qui contient la liste des agents transporteurs (*C_HumanCarrier*, voir VI.B.3.b).

C_Chronogram est la classe qui se charge de contenir les événements (par ordre chronologique) permettant la création et les mises à jour de l'environnement. Nous y reviendrons dans les sections II.A.4 et II.B.3.

C_Calendar permet de connaitre la date courante du monde et l'arrivée des dates pour la mise à jour de l'environnement ; le calendrier s'incrémente à chaque pas de simulation. Cette incrémentation se fait avec les champs de *C_Calendar* que sont :

- *tickAmount_Ucalendar* (ex. 2) ;
- *tick_UcalendarField* (ex. heure) ;
- *tickAmountMultiplicator* (par défaut égal à 1) est un multiplicateur permettant d'ajuster le pas d'incrémentation du calendrier. Ce multiplicateur est égal à 1 en mono-échelle temporelle. Cependant il peut varier en multi-échelle temporelle (voir B.2.I.D).

Les valeurs par défaut de *tickAmount_Ucalendar* et de *tick_UcalendarField* sont celles de l'échelle temporelle du monde et celle de *tickAmountMultiplicator* est 1. Par exemple si l'échelle temporelle du monde est de 2 heures, son calendrier doit s'incrémenter de 2 heures à chaque pas de simulation du monde. Ainsi :

tickAmount_Ucalendar = 2 ;
tick_UcalendarField = heure ;
tickAmountMultiplicator = 1.

C_TimeAndSpaceConverter() est une classe accessible partout (*static*) et permettant de faire les conversions telles que :

- des distances avec des unités spatiales (ex. de kilomètre à mètre …) ;
- des durées avec des unités temporelles (ex. de jour en heure …) ;
- des durées en temps en des durées en tick (pas de simulation) ;
- des vitesses en distance par temps en des vitesses en distance par tick (ex. si 1 tick vaut 30 minutes alors 1 km/h correspond à 0,5 km/tick) ;
- des vitesses en distance par tick en des vitesses en nombre de cellules par tick (ex. si une cellule vaut 10 m alors 0,5 km/tick correspond à 50 cellules/tick) ;
- …

Cette classe est utile pour la gestion spatiale et temporelle qui fait l'objet des deux prochaines sections.

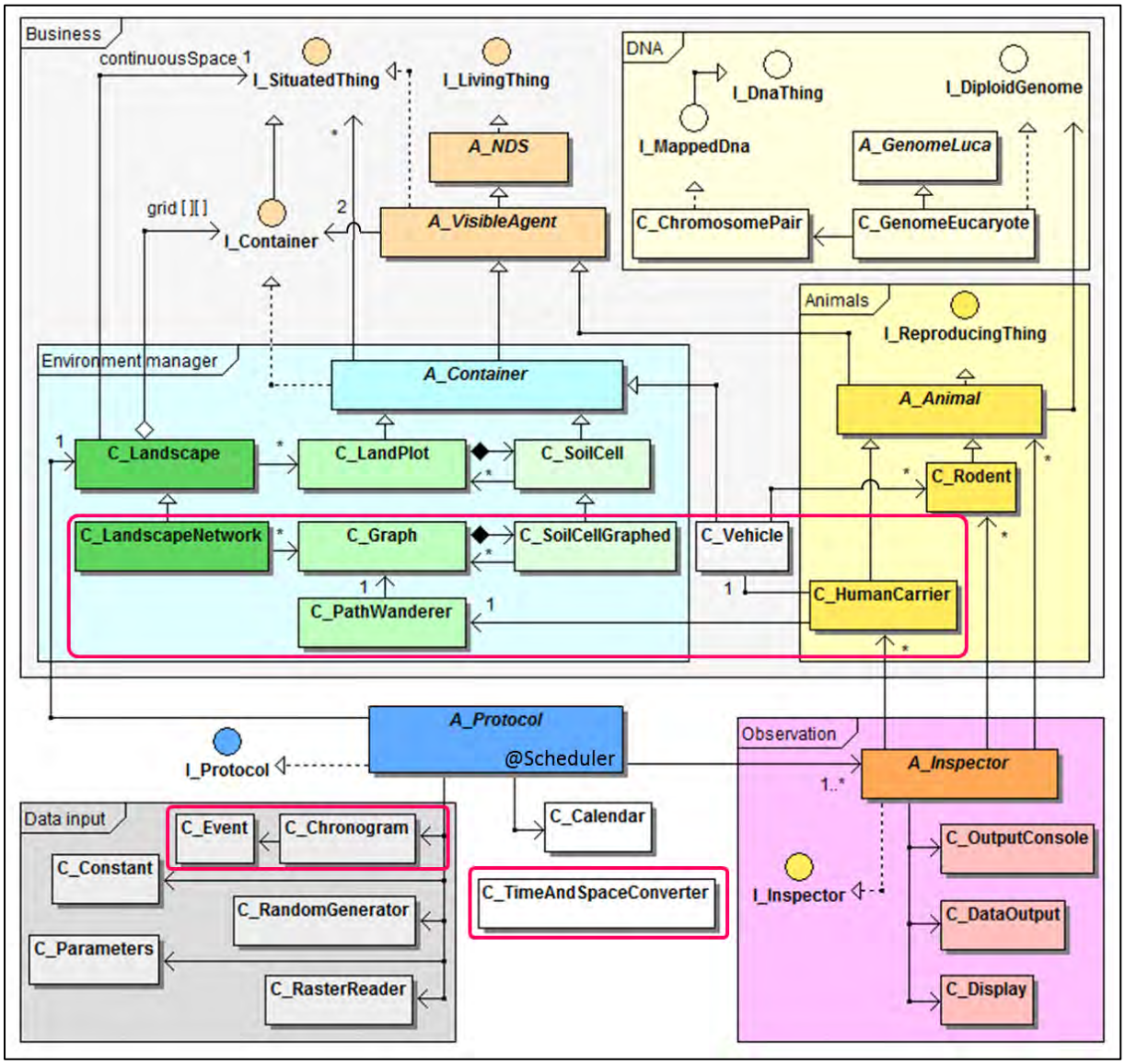

Figure 14 – Schéma UML du modèle.
Nomenclature des classes : I_ : Interface, A_ : classe Abstraite, C_ : Classe. Les classes encadrées en rouge sont nouvelles par rapport au schéma UML présenté dans VI.B.3.b.

II.A.2 Représentation et gestion de l'espace

L'espace est discrétisé en cellules élémentaires de même taille (voir Figure 15). Grâce au polymorphisme, ces cellules sont déclarées comme des *I_Container* (dans *C_Landscape*) et instanciées avec des *C_SoilCellGraphed* (dans *C_LandscapeNetwork* qui étend *C_Landscape*). Etant des *C_SoilCell*, les *C_SoilCellGraphed* peuvent contenir plusieurs informations permettant de connaître au besoin position, nombre d'agents, affinité pour les rongeurs, mais aussi type de terrain (route, rail, rivière, ville, et/ou bioclimat etc.). A la base, une *soilCell* ne pouvait avoir qu'un type de terrain et ne pouvait appartenir qu'à un seul *landPlot* à la fois. Ceci ne permet pas de représenter convenablement par exemple, une route et un rail (qui sont des *landPlot*) qui passent par une même *soilCell*. Pour remédier à cela, nous avons réadapté *C_SoilCell* pour lui permettre de pouvoir avoir plusieurs types de terrains et appartenir à plusieurs *landPlots* en même temps.

Les coordonnées d'une *soilCell* sur la grille et/ou sur l'espace continu étaient recalculées à chaque fois qu'un agent ou le gestionnaire d'espace avait besoin de la localiser. Ce calcul se faisait avec une complexité O(n*m) (avec n et m respectivement le nombre de lignes et de colonnes de la grille). En effet, ces coordonnées n'étaient pas connues a priori et il était nécessaire de parcourir la grille pour les retrouver : en testant à chaque itération si la *soilCell* courante de la grille correspond à celle dont on souhaite connaitre les coordonnées ; si c'est le cas, alors on récupère la ligne et la colonne correspondantes et ensuite on les multiplie par la taille des *soilCell* pour obtenir les coordonnées sur l'espace continu. Nous avons ainsi optimisé cette manière de localiser les *soilCell* sur la grille et sur l'espace continu en ajoutant à la classe *C_SoilCell* des champs permettant aux *soilCell* de sauvegarder leurs coordonnées discrètes et en nombres réels dès leur création puisqu'elles sont immobiles. De ce fait, elles sont localisées sur la grille et sur l'espace continu avec une complexité constante O(1).

Nous avons ensuite étendu *C_SoilCell* en *C_SoilCellGraphed* particulièrement pour la prise en compte des topologies graphes (*C_Graph*) composées de *C_SoilCellGraphed*. Ces dernières peuvent elles aussi appartenir à plusieurs graphes de type différents comme par exemple une route et un chemin de fer qui passent par une même cellule (voir Figure 15).

Comme décrit précédemment un ensemble de cellules élémentaires contiguës de même type est un *C_LandPlot*. Cette classe permet la délimitation et l'identification des différentes zones de l'environnement (champs, villes, routes, chemins de fer, rivières, bassins arachidiers etc.) ; ce qui est utile par exemple pour les agents transporteurs qui ne doivent pas sortir d'une zone donnée (*e.g.* les transporteurs qui font toute leur activité dans le bassin arachidier). Une cellule pouvant appartenir à plusieurs *C_LandPlots* (de types différents) on peut alors représenter plusieurs types de terrains à un même endroit. Par exemple une route passant sur une zone du bassin arachidier qui à son tour se trouve dans une ville.

A partir des *C_landPlots* de type voies de transports, les graphes (*C_Graph*) sont construits, ce qui garantit leur connexité (voir la section II.A.5.b, p. 60).

La classe *C_LandscapeNetwork* qui étend *C_Landscape* (Figure 14) se charge alors de la construction et de la gestion globale de l'environnement ainsi que de mettre en relation les

agents et l'espace au cours de la simulation (*e.g.* elle déréférence un agent d'une position et le référence à une autre au cours du déplacement de l'agent). Ainsi, cette classe contient :

i. l'espace continu qui représente une topologie où les agents peuvent se localiser (en coordonnées en nombres réels) et se déplacer,
ii. une matrice (grille) de *I_Container* qui sera instanciée avec des *C_SoilCellGraphed* représentant la discrétisation de l'environnement, la taille des cellules dépend du niveau de détail que l'on a choisi. La grille représente une autre topologie permettant aux agents de se localiser (en coordonnées discrètes), se déplacer mais aussi de percevoir facilement leur voisinage (voisinage de Moore, von Neumann),
iii. des listes et des procédures permettant détection, construction et stockage des *landPlots* représentant les différentes zones de l'espace et des *graphes* représentant les voies de communication (route, rail et rivière), une nouvelle topologie où se déplacent les véhicules.

Nous appelons l'échelle ou la granularité spatiale d'un monde la taille des cellules de la grille (TCG) de l'environnement de ce monde.

Cette échelle dépend donc du niveau de détail choisi. Ainsi pour cette étude de cas dont seule nous intéresse la diffusion des rats d'une ville à une autre par le transport humain, la taille des cellules est de 7,5 km de coté (Figure 15).

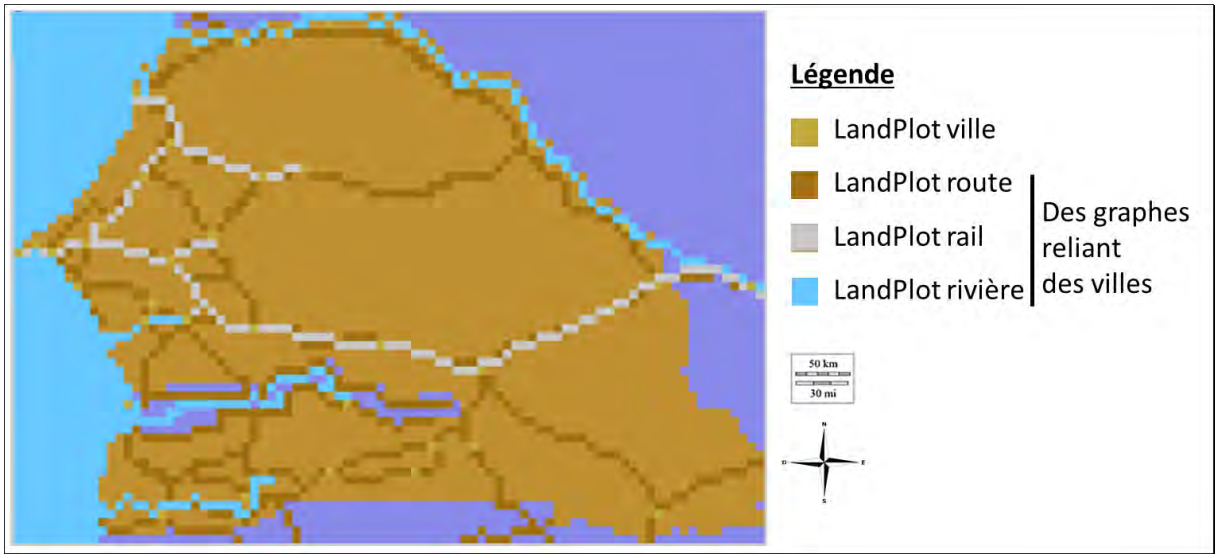

Figure 15 – Discrétisation de l'espace sénégalais au sein du simulateur.
Un carré unitaire correspond à des s*oilCellGraphed*, une surface homogène correspond à un *LandPlot* ; les routes, voies ferrées et fleuves sont aussi formalisés sous forme de *graphes*.

II.A.2.a Prise en compte d'un éventuel changement d'échelle spatiale

Contrairement aux agents transporteurs qui peuvent choisir une destination lointaine (à partir d'une liste de destinations élaborée dès leur création), les agents rats ne peuvent se déplacer et/ou agir que sur les objets (destinations) et agents qui se trouvent dans leur voisinage. En effet les rats perçoivent leur environnement proche suivant un rayon de

perception fixe (R) : à partir de R ils récupèrent les cellules de l'environnement qui intersectent leur disque de perception (disque de rayon R et centré sur l'agent, voir Figure 16).

L'intérêt de ceci est que pour un type d'agent donné, le rayon de perception est fixe mais l'ordre du voisinage de Moore (par exemple) ne l'est pas, il dépend de l'échelle spatiale du monde (taille des cellules de la grille, voir Figure 16). Cette manière de faire de SimMasto est donc compatible avec une modélisation multi-échelle spatiale. C'est-à-dire qu'il n'est pas nécessaire de reprogrammer un type d'agent ou de changer le nombre de cellules qu'il peut percevoir quand on change l'échelle spatiale. Ceci est valable dans le cas où le changement se fait au sein d'un même monde mono-échelle ou dans le cas où un agent change d'échelle spatiale en changeant de monde (voir B.2.I.C).

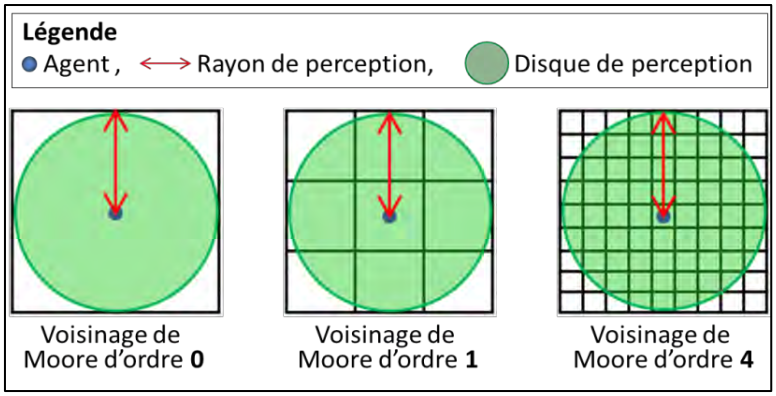

Figure 16 – Le nombre de cellules perçues dépend de la taille des cellules de la grille.

II.A.3 Gestion du temps

Dans la modélisation à base d'agents, le *temps* est une organisation structurée par un référentiel temporel (ou temporalité) dans lequel s'inscrivent toutes les entités dynamiques d'un modèle [Langlois, 2009]. En effet une simulation est un modèle plongé dans le *temps* [Coquillard et Hill, 1997]. Ce dernier est donc représenté (du moins pour les simulations en temps discret que nous réalisons) par un référentiel monodimensionnel défini par une origine, une unité de temps, un pas de temps élémentaire et un ordonnanceur élémentaire (scheduler).

Nous appelons l'échelle ou la granularité temporelle d'un monde la durée du pas de simulation de ce monde.

Nous appelons tick monde le pas de simulation pour un monde.

L'échelle temporelle est donc la durée du tick monde que nous abrégeons par DTM.

Notre objectif ici est, au sein d'un monde, *1)* de coordonner l'exécution non uniforme d'unités d'activité, de durées diverses, d'agents ; *2)* de faire de telle sorte qu'on ait plus besoin de changer la durée de l'unité d'activité des agents après un changement d'échelle temporelle.

II.A.3.a Prise en compte en compte un éventuel changement temporel

Pour un monde (mono-échelle), l'unité d'activité des agents est souvent programmée pour une durée fixe. De telle sorte qu'il est nécessaire de les reprogrammer si on veut changer

l'échelle temporelle du monde. Ainsi notre premier travail en ce qui concerne le temps était de faire de telle sorte que les agents du monde puissent s'adapter à un changement temporel. C'est-à-dire qu'il ne soit plus nécessaire de reprogrammer leur unité d'activité quand on voudrait changer l'échelle temporelle. Ceci est valable dans le cas où le changement d'échelle temporelle se fait au sein d'un même monde ou dans le cas où c'est l'agent qui change d'échelle temporelle à l'issue d'un changement de monde (voir B.2.I.D).

Pour mettre en œuvre cela, c'est-à-dire permettre aux agents de pouvoir s'adapter à un éventuel changement d'échelle temporelle, il faut prendre en compte les deux aspects suivants :

1- il faut que l'unité d'activité des agents puisse dépendre du temps. En d'autres termes, les agents doivent être capables d'adapter leur unité d'activité si la durée du tick monde s'allongeait ou raccourcissait. Nous pouvons prendre par exemple le cas des agents transporteurs pour cette étude. En effet l'unité d'activité de ces agents consiste à ne faire que de longs déplacements (après avoir choisi une destination qui ne coute pas du temps). Puisque l'on se déplace avec une vitesse et que cette dernière est une distance divisée par une durée, alors il est possible de contrôler la distance à parcourir à partir de la durée disponible (*DTM*). Par exemple si pour un transporteur la vitesse (moyenne) est de 60 km/h, alors la distance qu'il peut parcourir à chaque pas de simulation, dans un monde, doit être :
   - 60 km si le pas de temps est de 1 h ;
   - 30 km si le pas de temps est de 30 minutes ;
   - 120 km si le pas de temps est de 2 h ;
   - …

   Ces agents peuvent donc avoir la main à chaque pas de simulation, il suffit seulement que leur unité d'activité soit programmée de telle sorte qu'elle puisse dépendre du temps. Cependant, ceci n'est pas facile pour les unités d'activité de tout type d'agent. Ce qui nous amène au point suivant.
2- pour les agents dont l'unité d'activité est programmée pour une durée fixe, alors il revient à l'ordonnanceur du monde d'adapter la fréquence à laquelle il donne la main à ces agents pour qu'ils exécutent leur unité d'activité. Cette fréquence sera en fonction de la durée de leur unité d'activité et de la durée du pas de simulation du monde où ils se trouvent. Nous pouvons prendre par exemple le cas des agents rats pour cette étude. Si on suppose que leur unité d'activité est programmée pour une durée fixe de 1h, alors la fréquence à laquelle l'ordonnanceur du monde donne la main aux agents de type rat pour qu'ils réalisent leur unité d'activité sera :
   - 1 fois à chaque tick monde si la *DTM* vaut 1 h ;
   - 1 fois à chaque 2 ticks monde si la *DTM* vaut 30 minutes ;
   - 2 fois à chaque tick monde si la *DTM* vaut 2 h ;
   - 1,5 fois à chaque tick monde (ou 15 fois chaque 10 ticks monde) si la *DTM* vaut 40 minutes ;
   - …

Dans la section suivante nous montrons comment nous avons mis en œuvre les idées développées dans les deux points précedents.

II.A.3.b Agents dont l'unité d'activité peut dépendre du temps (transporteur)

Pour que ce type d'agent puisse s'adapter à un changement temporel, il faut convertir sa vitesse de distance par temps en distance par tick monde et la recalculer à chaque fois que l'échelle temporelle change :

Conversion de la vitesse en distance par tick monde:

On commence par unifier les unités temporelle. On peut ainsi convertir les unités temporelles en milliseconde (ms)

Soient :

- V_{u_ms} la vitesse en distance par milliseconde (ex. *60 km/h = 60km/(m × ms)*) ;
- $V_{u\_tickMonde}$ la vitesse en distance par tick monde (ex. *60 km/tickMonde*) ;
- $n \times ms$ la durée du tick monde (*1* $tickMonde = n \times ms$).

Alors

$$V_{u\_ms} = \frac{distance}{temps} = \frac{distance}{m \times ms}, avec\ (temps = m \times ms)\ (1)$$

$$1\ tickMonde = n \times ms \Rightarrow\ 1\ ms = \frac{1\ tickMonde}{n}\ (2)$$

En remplaçant $1\ ms$ de (2) dans V_{u_ms} de (1), alors on fait disparaitre le temps et fait apparaitre le tick monde dans la formule de la vitesse , ce qui donne la vitesse en tick monde :

$$(1)\ et\ (2)\ \Rightarrow\ \boldsymbol{V_{u\_tick} = \frac{distance \times n}{m}/tickMonde}$$

Exemple illustratif

Soient :

$V_{u\_ms} = \frac{60km}{3\ 600\ 000\ ms}$: $distance = 60km, m = 3\ 600\ 000$

Alors :

si $1tick = 1h : n = 3\ 600\ 000 \Rightarrow V_{u\_tick} = 60\ km/tick$

si $1tick = 2h : n = 7\ 200\ 000 \Rightarrow V_{u\_tick} = 120\ km/tick$

si $1tick = 0{,}5h : n = 1\ 800\ 000 \Rightarrow V_{u\_tick} = 30\ km/tick$

Suivant cette méthode de conversion, nous avons mis dans la classe *C_TimeAndSpaceConverter* une méthode *speed_ToDistanceByTickMonde* permettant de convertir les vitesses en distance par tick. Cette méthode prend en paramètre : l'échelle temporelle (DTM) et l'échelle spatiale (TCG) du monde ainsi qu'une vitesse (la vitesse d'un transporteur par exemple). Elle retourne un nombre (dont l'unité est celle de l'échelle spatiale par tick).

II.A.3.c Agents dont l'unité d'activité est programmée pour une durée fixe (rat)

Contrairement à l'autre type d'agent c'est l'ordonnanceur qui doit, cette fois-ci, adapter la fréquence d'exécution des agents en fonction de la durée du tick monde et de la durée à laquelle l'unité d'activité des agents a été programmée. Ainsi, on ne peut avoir que quatre situations possibles ; selon que la durée de l'unité d'activité de l'agent est :

a) un multiple de la DTM ;
b) supérieur à la DTM sans être son multiple ;
c) un diviseur de la DTM ;
d) inférieur à la DTM sans être son diviseur.

Pour que l'ordonnanceur puisse prendre en compte ces quatre cas de figure, nous proposons de les étudier un à un avec des exemples à l'appui. Nous allons nous baser ainsi sur l'exemple du Tableau 1 ci-dessous correspondant respectivement aux situations a), b), c) et d). Nous allons aussi utiliser les abréviations suivantes :

DUA_{Ai} : la durée de l'unité d'activité pour une liste d'agents *Ai* ;

DTM : la durée du pas de simulation pour un monde (durée du tick monde) ;

N1 : le nombre de tick monde au bout duquel les agents d'une liste *Ai* reçoivent la main pour exécuter leur unité d'activité. On va utiliser *N1* quand DUA_{Ai} est supérieure à *DTM*.

N2 : le nombre de fois où les agents d'une liste *Ai* ont la main au sein d'un tick monde pour exécuter leur unité d'activité. On va utiliser *N2* quand DUA_{Ai} inférieure à *DTM*.

Pour faire simple nous allons utiliser le terme tick au lieu de tick monde pour le reste de ce chapitre.

Liste d'Agent	**A_1**	**A_2**	**A_3**	**A_4**
$\boldsymbol{DUA_{Ai}}$	DUA_{A1} = 12h	DUA_{A2} = 8h	DUA_{A3} = 3h	DUA_{A4} = 4h
DTM = 6 h				

Tableau 1 – Durée de l'unité d'activité des agents des listes Ai.

Situation où DUA_{Ai} est un multiple de DTM

Nous prenons l'exemple de la liste A_1 :

$DUA_{A1} = 12h$;
$N1 = DUA_{A1} / DTM = 2.$

Les agents de la liste A_1 doivent donc avoir la main *1 fois* tous les *2 ticks* (*N1 fois*), c'est-à-dire quand la valeur du *compteur de tick*s est un multiple de *N1*. Nous pouvons illustrer ceci avec la Figure 17 suivante :

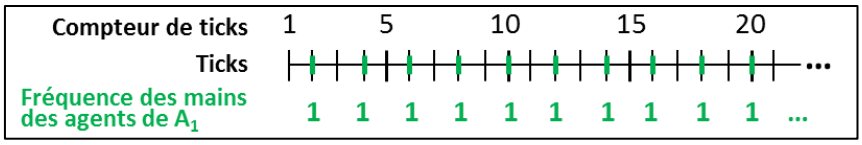

Figure 17 – Situation où DUA_{Ai} est un multiple de DTM.
On prend par exemple le cas où $N1 = 2$ (DUA_{A1} / DTM). Dans ce cas les agents de la liste A_1 ont la main 1 fois (en vert) tous les *2* ticks. C'est-à-dire quand la valeur du *compteur de tick*s est un multiple de *2*.

Nous proposons l'Algorithme 2 (Partie II) pour mettre en œuvre cet ordonnancement.

Situation où DUA_{Ai} est supérieure à DTM mais n'étant pas son multiple

Nous prenons l'exemple de la liste A_2 :

$DUA_{A2} = 8\ h$;
$N1 = DUA_{A2} / DTM = 1,3$.

Les agents de la liste A_2 doivent donc avoir la main *1 fois* tous les *1,3 ticks*. En d'autres termes, puisque le *tick* est discret, ils doivent avoir la main *10 fois* tous les *13 ticks* (pour être encore plus précis on peut aussi dire *100 fois* tous les *133 ticks* etc.). Nous allons donc dire que les agents de A_2 prennent la main *1 fois* tous les *2 ticks* (arrondi par excès de DUA_{A2} / DTM) au lieu de *1,3 ticks* (DUA_{A2} / DTM). Par contre arrivés à un certain nombre de *ticks*, que nous notons *B*, ils prennent la main tous les *tick* (au lieu de tous les *2 ticks*) pour pouvoir arriver à avoir *10 fois* la main à la fin des *13 ticks* (voir Figure 18).

Détermination de *B*

Soient :
N1 = arrondi par excès de DUA_{A2} / DTM, ($N1 = 2$ avec l'exemple) ;
M = arrondi par défaut de $DUA_{A2} \times 10 / DTM$, ($M = 13$ avec l'exemple) .

Alors :
$B = (M - 10) / (N1 - 1)$, ($B = 3$ avec l'exemple).

Preuve (pour une meilleure compréhension on peut se baser sur la Figure 18 ci-dessous) :

Si à partir d'un certain nombre de ticks, 1 par exemple, les agents de A_2 ont la *main B fois* et qu'ils doivent en avoir *10 fois* quand on arrive à *M ticks*, alors il leur reste *10 - B mains* à raison de *1 main* par tick. Au même moment, il reste à la simulation elle aussi $M - (B \times N1)$ *ticks* pour arriver à *M ticks*. Ainsi nous pouvons dire que :

$$10 - B = M - (B \times N1) \Rightarrow B = (M - 10) / (N1 - 1)$$

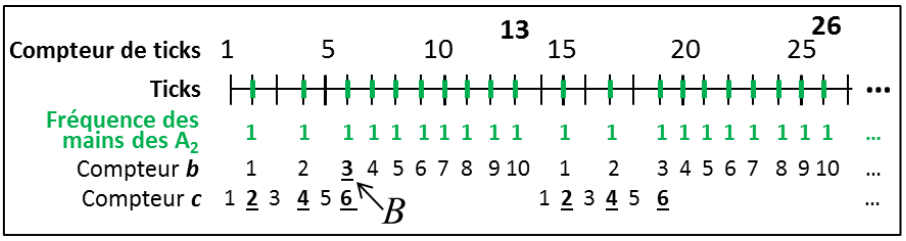

Figure 18 – Situation où DUA_{Ai} est supérieure à DTM mais n'étant pas son multiple.
On prend par exemple le cas où *N1 = 2* (arrondi par excès de DUA_{A2} / DTM). Dans ce cas les agents de la liste A_2 ont la main 1 fois (en vert) tous les 2 ticks (*N1 ticks*). Après avoir eu *B fois* la main, la fréquence change, ils ont alors la main 1 fois tous les tick pendant *B – 10* ticks.

Pour mettre en place un algorithme (Algorithme 2, Partie III) permettant de mettre en œuvre cette ordonnancement, nous aurons besoin, en plus de *N1* et de *B*, de deux compteurs *b* et *c* :

- *b* : permet de savoir si on est avant ou après *B* et si on a atteint *10*. Si *b* est inférieur ou égal à *B* alors les agents ont la *main 1 fois* tous les *N1 tick* ; et si *b* est supérieur à *B* alors les agents ont la *main 1 fois* tous les *tick*. Ce compteur *b* s'incrémente par pas de *1* à chaque fois que les agents ont la main et est réinitialisé à *1* quand il atteint *10* ;
- *c* : permet de connaitre les multiples de *N1* quand *b* est inférieur ou égal à *B* (car, pour ce cas-ci, le compteur de ticks ne marche pas pour déterminer les bons multiples de *N1*). Le compteur *c* s'incrémente par pas de *1* à chaque tick à condition que *b* soit inférieur à *B*. Il est réinitialisé à *1* quand *b* atteint *10*.

Situation où DUA_{Ai} est un diviseur de DTM

Nous prenons l'exemple de la liste A_3 :
$DUA_{A3} = 3h$;
$N2 = DTM / DUA_{A3} = 2$.

Les agents de la liste A_3 doivent donc avoir la main *2 fois* (*N2 fois*) à chaque *tick*. Nous pouvons illustrer ceci avec la Figure 19 suivante :

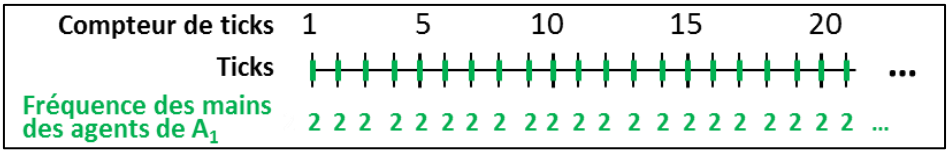

Figure 19 – Situation où DUA_{Ai} est un diviseur de DTM.
On prend par exemple le cas où *N2 = 2* (DTM / DUA_{A3}). Ici les agents de la liste A_3 ont la main *2 fois* (en vert) à chaque *tick*.

On peut voir ci-dessous l'Algorithme 2 (Partie IV) qui permet de mettre en œuvre cet ordonnancement.

Situation où DUA_{Ai} est inférieure ou égal à DTM mais n'étant son diviseur.

Nous prenons l'exemple de la liste A_4 :

$DUA_{A4} = 4\ h$;

$N2 = DTM / DUA_{A4} = 1{,}5$.

Les agents de la liste A_4 doivent donc avoir la main *1,5 fois* à chaque *tick*. En d'autre termes, puisque le *tick* est discret, ils doivent avoir la main *15 fois* tous les *10 ticks*. Nous allons donc dire que les agents de A_2 prennent la main, au lieu de *1,5 fois* (DTM / DUA_{A4}), mais *2 fois* (arrondi par excès de DTM / DUA_{A4}) à chaque *tick*. Par contre arrivés à un certain nombre de *ticks*, que nous notons *B*, ils prennent la main *1 fois* par *tick* pour en arriver à *15* mains au bout de *10 ticks*.

Détermination de *B*

Soient :

N2 = arrondi par excès de DTM / DUA_{A4}, (*N2 = 2* avec l'exemple) ;

M = arrondi par défaut de $DUA_{A2} \times 10 / DTM$, (*M = 15* avec l'exemple) .

Alors :

$B = (M - 10) / (N2 - 1)$, (*B* = 5 avec l'exemple).

Preuve (pour une meilleure compréhension on peut se baser sur la Figure 20 ci-dessous) :

Si à partir d'un certain nombre de ticks, 1 par exemple, les agents de A_4 ont la *main N2* × *B fois* et qu'ils doivent en avoir *M fois* en *10 ticks*, alors il leur reste *M - N2* × *B mains* à raison de *1 main* par *tick*. Au même moment, il reste à la simulation *10 - B ticks* pour arriver à *10 ticks*. Ainsi nous pouvons dire que :

$$M - (B \times N2) = 10 - B \Rightarrow B = (M - 10) / (N2 - 1)$$

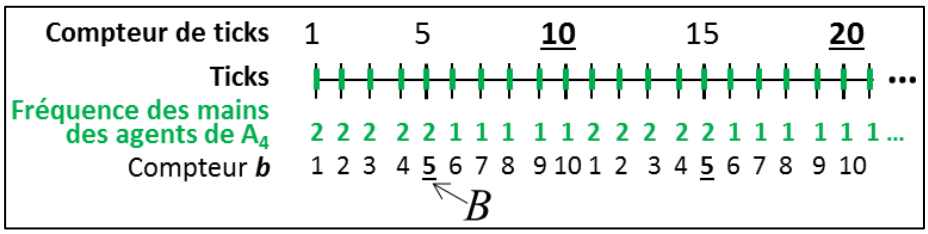

Figure 20 – Situation où DUA_{Ai} est inférieure ou égal à DTM mais n'étant son diviseur. On prend par exemple le cas où *N2 = 2* (arrondi par excès de DTM / DUA_{A4}). Les agents de la liste A_4 ont ici la main à chaque tick : *2 fois* (*N2* fois) quand *b* est inférieur ou égal à *B* et *1 fois* quand *b* est supérieur à *B*. Ce qui leur permet d'avoir *M fois* (*15 fois*) la main tous les *10 ticks*.

Pour mettre en place un algorithme (Algorithme 2, Partie V) permettant de mettre en œuvre cet ordonnancement, nous aurons besoin ici aussi, en plus de *N2* et de *B*, d'un compteur *b* :

b : permet de savoir si on est avant ou après *B* et si on a atteint *10*. Si *b* est inférieur à *B* alors les agents de A_4 ont la *main 2 fois* (*N2 fois*) à chaque *tick* ; et si *b* est supérieur à *B*

alors les agents ont la *main 1 fois* à chaque *tick*. Ce compteur s'incrémente par pas de *1* à chaque fois que les agents ont la main et recommence à *1* quand il atteint *10*.

En ce qui concerne les deux dernières situations c) et d), où un agent peut avoir plusieurs fois la main à un tick, l'ordonnanceur (décrit ci-après) ne lui donne pas, de manière consécutive, plusieurs fois la main pour qu'ils réalisent consécutivement toutes ses unités d'activité pour ce tick. Mais plutôt il recence toutes les listes d'agents de ce genre, puis il donne une fois la main à tous les agents de ces listes, ensuite il leur redonne la main une deuxième fois ainsi de suite. Si les agents d'une liste épuisent le nombre de fois pour lesquelles ils doivent avoir la main, alors ils laissent les autres continuer, jusqu'à ce que tous les agents aient la main le nombre de fois qu'ils doivent l'avoir pour ce tick. Ce qui permet de bien mélanger l'exécution des unités d'activités des agents dans le cas où la *DUA* de certains agent est plus petite que la *DTM* du monde. Ceci est pris en compte dans la dernirère partie de l'algorithme (Algorithme 2, Partie VI) de la fonction *ordonancerActivitesAgents ()*.

Classe ordonnanceur

Pour ordonnancer la fréquence d'exécution des unités d'activités des agents suivant les quatre situations présentées précédemment, nous avons mis en place pour chaque monde un ordonnanceur (classe C_Scheduler, voir I.B, p. 75) disposant de :

- quatre tableaux associatifs permettant d'associer les listes d'agents aux informations nécessaires pour ordonnancer l'exécution des unités d'activités des agents qu'ils contiennent. Ces tableaux associatifs contiennent donc des listes d'agents (*listeAgent*) associées à des tableaux d'entiers (*tableauInfos*) dont le contenu dépend des quatre situations précédemment présentées:
  a) *multiple_DTM_ListeAgent* : quand *DUA* est un multiple *DTM*. *tableauInfos* contient *N1* et *DUA* ;
  b) *non_multiple_DTM_ListeAgent* : quand *DUA* est supérieure à *DTM* mais n'étant pas son multiple. *tableauInfos* contient *N1*, *B*, *DUA* et les compteurs *b* et *c* ;
  c) *diviseur_DTM_ListeAgent* : quand *DUA* est un diviseur *DTM*. *tableauInfos* contient *N2* et *DUA* ;
  d) *non_diviseur_DTM_ListeAgent* : quand *DUA* est inférieure à *DTM* mais n'étant pas son diviseur. *tableauInfos* contient *N2*, *B*, un compteur *b* et *DUA*.

  DUA est concervée pour chaque liste d'agents, elle servira au changement d'échelle temporelle d'un monde en cours de la simulation (voir la section II.A.3.d) ;
- un champs *compteurTickMonde* permettant de compter les ticks monde ;
- un champs *DTM* qui contient la durée du pas de simulation ;
- une fonction *ajoutListeAgent (listeAgent, DUA)* permettant de renseigner les tableaux associatifs cités ci-dessous. Cette fonction prend en paramètre une liste d'agents (une référence de *listeAgent*) et la durée de l'unité d'activité (*DUA*) des agents de cette liste ;
- une fonction *ordonancerActivitesAgents()* qui permet de donner la main aux agents avec les bonnes fréquences ;

- une fonction *changementDeLaDTM()* qui permet de recalculer les éléments des tableaux *tableauInfos* après un changement d'échelle temporelle du monde en cours de simulation (voir section II.A.3.d, p. 56) ;
- une fonction *lectureDesParametres()* qui permet de lire les paramètres temporels. Ces derniers ne sont pris en compte qu'en début de simulation et quand ils sont modifiés (voir II.A.3.d, p. 56) en cours de simulation.

Mise en œuvre de l'ordonnancement

Pendant le développement du modèle, il suffit donc de mettre autant de listes (*listeAgent*) que d'agents ayant des *DUA* différentes. Ensuite il faut renseigner les tableaux associatifs présentés ci-dessus en utilisant la fonction *ajoutListeAgent (listeAgent, DUA)* en lui donnant en paramètre, à chaque appel, une liste d'agents (*listeAgent*) et son *DUA* correspondant. Cette fonction sera donc appelée autant de fois qu'il y aura de listes d'agents et à chaque fois que la *DTM* du monde change (voir la section II.A.3.d, p. 56).

Après cela, c'est la fonction *ordonancerActivitesAgents()* qui est exécutée à chaque pas de simulation par l'ordonnanceur interne de Repast Simphony (avec une annotation @Scheduler). Ainsi elle ordonnance pendant toute la simulation la manière de donner la main aux agents, suivant la durée du tick et celle de leur unité d'activité, afin qu'ils les réalisent.

Il faut noter que l'ordonnanceur ordonnance l'unité d'activité des agents indépendamment de ce qui se passe sur les listes d'agents (*listeAgent*). Il ne fait que les parcourir. Il ne peut ni y ajouter des agents, ni en supprimer. Ainsi ces listes et leur gestion (ajout, suppression d'agents) restent du ressort des inspecteurs comme c'était par exemple le cas avec SimMasto avant l'intégration de notre travail. Ce qui rend plus facile le travail et garantit la compatibilité ascendante.

Nous avons présenté dans l'annexe B (Algorithme 1 et Algorithme 2) les algorithmes des fonctions *ajoutListeAgent()* et *ordonancerActivitesAgents()*.

II.A.3.d Changer l'échelle temporelle en cours de simulation

Après avoir préparé les agents et l'ordonnanceur à un éventuel changement temporel, nous pouvons à présent changer la durée du tick monde (échelle temporelle) en cours de simulation du monde mono-échelle, ce qui est utile pour un observateur devant son écran (échelle d'observation) [Ratzé *et al.*, 2007]. Pour cela nous avons fait de telle sorte que certains paramètres du modèle (dans l'onglet *Parameters* Figure 21) puissent être lus en cours de simulation (avec la fonction *lectureDesParametres()*). Parmi eux nous avons mis les paramètres temporels. Ainsi pour changer l'échelle temporelle du monde on change la durée du tick dans ces paramètres (voir Figure 21). Ce qui modifie la valeur du champ *DTM* de l'ordonnanceur et des champs d'incrémentation du calendrier. Enfin on réadapte *(i)* la valeur des champs dépendant de la durée du tick pour les agents dont l'unité d'activité n'est pas programmée pour une durée fixe, par exemple reconvertir les vitesses des véhicules en

distance par nouvelle durée du tick, voir II.A.3.b) ; et/ou *(ii)* les fréquences d'exécution des agents dont l'unité d'activité est programmée pour une durée fixe (voir II.A.3.c).

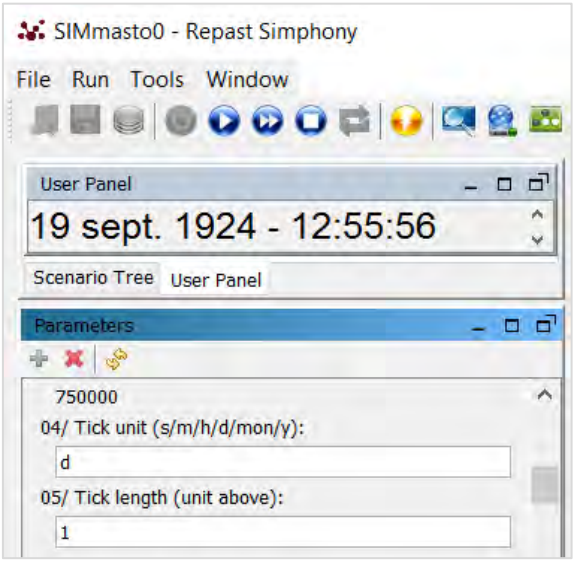

Figure 21 – Paramètres temporels utilisés dans SimMasto.
s: second, m: minute, h: hour, d: day, mon: month, y: year.

Reconvertir les vitesses en distance par tick pour les agents transporteurs

Dans les classes contenant les constantes du modèle sur la Figure 14, nous avons un tableau associatif, *VEHICLE_SPECS,* contenant pour chaque type de véhicule les propriétés telles que vitesse, type de voie :

VEHICLE_SPECS["truck"] = String[]{"60", "km/h", "road"};
VEHICLE_SPECS["train"] = String[]{"20", "km/h", "rail"};
VEHICLE_SPECS["boat"] = String[]{"15", "km/h", "river"};

Chaque *C_Vehicle* a aussi un champ *speed_UdistanceByTick* et un champ *type* (pouvant être *truck*, *train*, *boat*).

Ainsi si un changement temporel se produit, le protocole demande aux inspecteurs de mettre à jour la vitesse en tick des agents concernés. Il récupère ainsi pour chaque véhicule, à partir de son type, la vitesse avant conversion se trouvant dans *VEHICLE_SPECS* ; et avec la méthode *speed_ToDistanceByTickMonde* de *C_TimeAndSpaceConverter* il réalise la conversion en prenant en compte la nouvelle échelle temporelle.

Réadapter la fréquence d'exécution des agents à chaque pas de simulation

Cette réadaptation se fait en trois étapes : *(i)* faire des copies des quatre tableaux associatifs concernant l'ordonnancement, *(ii)* les vider et *(iii)* à partir des copies les reconstruire en utilisant de nouveau la méthode *ajoutListeAgent()*. Nous avons présenté dans l'annexe B l'algorithme (Algorithme 3) correspondant, du nom de *changementDeLaDTM()*.

II.A.4 Approche orientée connaissances-événements

Comme déjà dit, ce travail a pour vocation d'utiliser des connaissances multidisciplinaires dans un modèle formel qui permet de les intégrer et de les restituer aux thématiciens pour fournir de nouveau éclairages sur la façon dont s'articule les différents thématiques.

Ainsi, pour que les différents chercheurs biologistes, écologues, géographes puissent intégrer en temps utile dans la simulation tout type de connaissance liée à leur domaine d'expertise, nous avons choisi une approche orientée événement (cf. A.IV). Nous avons ainsi mis en place un chronogramme qui est une structure de données normalisée permettant d'introduire tout type de données constituant la transcription des connaissances multidisciplinaires. Le chronogramme est un fichier au format *csv* composé de lignes et de colonnes (Tableau 2) dans lequel chaque ligne ou bloc de lignes représente un événement complet. Le chronogramme a quelques contraintes à respecter :

- il est nécessairement chronologique et répond et par ordre, aux trois questions : *Quand* (jj/mm/aaaa), *Où* (x, y) et *Quoi* (nom de l'événement). Un complément d'information (permettant de préciser comment, combien, spécificité) peut être optionnellement fourni (Tableau 2) ;
- les mots utilisés doivent être bien unifiés (par exemple éviter de mettre à la fois St-Louis et Saint-Louis quand il signifie la même chose) ;
- ne pas mettre à une même date une instruction de création d'un événement *e* et une instruction qui utilise le même événement *e* (voir la section II.A.4.b) ;
- d'autres contraintes concernant la partie complément d'information (*VALUE1* et *VALUE2*) sont aussi à respecter (voir la section II.B.3.a).

Quand	Où		Quoi	Complément d'information	
DATE	X	Y	EVENT	VALUE1	VALUE2
…	…	…	…	…	…

Tableau 2 – Format du chronogramme.

II.A.4.a Stockage et parcours des données

Chaque chercheur communique ses connaissances et/ou ses données qui sont transcrites et/ou mises en forme et intégrées au chronogramme. Les classes *C_Event* et *C_Chronogram* (Figure 14) sont les classes de base permettant le stockage, le parcours et le traitement des événements du chronogramme. *C_Chronogram* copie les lignes du chronogramme, construit des objets événements avec la classe *C_Event* et les stocke dans la collection (au sens POO) *dataChrono* (un champ de *C_Chronogram)*. *C_Chronogram* dispose aussi d'un champ *nextDate* permettant de garder la date pour le ou les prochains événements en attente. Durant la simulation, la collection *dataChrono* n'est parcourue que si la date de la simulation correspond à celle dans *nextDate*. Chaque type d'événement dispose de son traitement spécifique réalisé par le protocole (celui du *centennal*, voir la section A.IV). Les instructions (sous forme d'événements historiques) de ce chronogramme permettent de construire et de mettre à jour l'environnement.

II.A.4.b Mise à jour de l'environnement à partir des événements

A chaque pas de temps, le protocole compare donc la date de la simulation à la date dans *nextDate* de *C_Chronogram*. Si les dates concordent l'environnement est mis à jour en deux étapes :

- la première étape est une mise à jour élémentaire. Elle consiste à parcourir ligne par ligne le tableau *dataChrono* jusqu'à ce que les événements pour cette date soient épuisés. Pour chaque ligne lue, l'environnement est mis à jour :
  - soit par modification des types d'un *SoilCellGraphed* (ex: construction d'une route passant par cette cellule) ;
  - soit par ajout à un endroit donné (ex. les comptoirs coloniaux) d'une population de rats. Ceci concerne le début de simulation, après cela les rats par exemple se reproduisent et meurent de façon autonome (la démographie des rongeurs est incluse dans le modèle) ;
  - soit par augmentation ou diminution à un endroit donné, du nombre de camions, de trains ou de bateaux dans les parcs de véhicules. En effet les véhicules ne se reproduisent pas, la fluctuation de leur nombre dans temps est enregistrée dans le chronogramme ;
  - après cela la date du prochain événement remplace celle dans *nextDate* de *C_Chronogram* ;
- la deuxième étape consiste à intégrer les mises à jour élémentaires de l'environnement pour tous les événements reçus à cette date. C'est-à-dire la mise à jour des différentes zones de l'espace en intégrant les cellules mises à jour concernées. Par exemple prolongation, raccourcissement ou apparition d'une route, d'un bassin arachidier, d'une ville etc. Cette mise à jour est directement prise en compte par les agents (voir la section II.B.3.b, p. 64).

La division en deux étapes de la mise à jour de l'environnement explique la nécessité de la contrainte du chronogramme : ne pas mettre à une même date une instruction de création d'un événement *e* et une instruction qui utilise le même événement *e*.

II.A.5 Le déplacement des agents transporteurs

Chaque type de transporteur connait sa vitesse moyenne (déterminée par les chercheurs) et le type de graphe sur lequel il doit se déplacer : les camions sur les routes, les trains sur les rails et les bateaux sur les rivières (ces informations n'ont donc pas besoin d'être dans le chronogramme, elles sont dans les constantes, *VEHICLE_SPECS*).

Après création d'un agent transporteur sous l'ordre du chronogramme, il récupère à partir de *VEHICLE_SPECS* le graphe correspondant à sa spécialité comprenant sa cellule courante (car un agent transporteur ne peut être créé que sur des cellules appartenant aux graphes où il doit évoluer). Le transporteur élabore ainsi, à partir de ce graphe, une liste de destinations possibles (toutes les villes reliées à ce graphe) ou il utilise la liste de destinations que l'on lui a éventuellement spécifiée dans le chronogramme (voir la section II.B.3.a).

II.A.5.a Choix de la destination

Un agent transporteur connait le graphe sur lequel il évolue et la liste de villes (reliées au graphe) où il peut aller. Lors de sa phase de délibération, le transporteur choisit aléatoirement une ville de sa liste de destinations, ce choix est pondéré par la taille de la population de la ville au moment du choix (car cette taille est mise à jour au fil de la simulation avec les données du chronogramme). C'est-à-dire comme dans la réalité, plus une ville est peuplée, plus les transporteurs s'y rendent (modèle gravitaire).

Pour mieux expliquer comment nous avons procédé prenons l'exemple du Tableau 3 contenant des villes avec des tailles de population, les effectifs cumulés croissants de ces tailles et la probabilité pour qu'un véhicule s'y rende. Les transporteurs doivent donc aller à toutes les villes, mais plus fréquemment à la ville 3 puis à la ville 4 etc. (puisque les probabilités sont plus élevées). Pour cela, nous avons élaboré un algorithme qui se présente de la façon suivante (Tableau 3) :

Soit *popTotal* la somme des tailles des populations (95 ici). A leur phase de délibération, les transporteurs choisissent un nombre aléatoire *NA* (entier) compris entre 1 et *popTotal*. Si *NA* est compris entre 1 et 5, *i.e.*, 5 possibilités sur 95 soit 5,26%, alors la ville 1 est choisie ; si *NA* est supérieur à 5 et est inférieur ou égal à 12, i.e. 7 possibilités sur 95 soit 7,37%, alors la ville 2 est choisie et ainsi de suite. Pour une optimisation spatiale, seul le tableau contenant les effectifs cumulés croissants des tailles de population est stocké dans le simulateur. Ce tableau est construit à partir des données du chronogramme.

Numéro ville	**1**	**2**	**3**	**4**	**5**
Taille population	5	7	50	30	3
Effectifs cumulés croissants	5	12	62	92	95
Probabilité d'être choisie	5,26%	7,37%	52,63%	31,58%	3,16%

Tableau 3 – Exemple de liste de villes avec leur taille de population.

II.A.5.b Construction du plus court chemin pour aller à la destination choisie

La Figure 22 illustre l'approche retenue pour la construction et le parcours d'un type de chemin reliant deux villes. Chaque case (cellule) représente un *SoilCellGraphed*, la partie grisée constitue un *LandPlot* appréhendé ici comme un graphe. En effet construire les graphes à partir des *landPlots* nous assure que les graphes sont connexes (existence d'un chemin entre chaque paire de nœuds). Dans l'algorithme développé chaque cellule constitue un nœud du graphe et entre deux nœuds successifs il y a une arête. De cette façon, une route constitue un ensemble de nœuds et d'arêtes de longueurs égales (taille d'une *C_SoilCellGraphed*) au lieu d'une seule arête reliant deux nœuds villes. Cette approche rend possible et suffisante l'utilisation de graphes non étiquetés. L'algorithme de Dijkstra [1959] pour les graphes non étiquetés détecte alors le plus court chemin et renvoie le chemin A (description et test de l'algorithme : [Mboup, 2012]). Des lourdeurs dues à l'utilisation de l'algorithme de Dijkstra ont été constatées lors des simulations. Cela est dû du fait que *1)* cet algorithme explore tous les nœuds d'un graphe pour élaborer le plus court chemin entre deux d'entre eux ; et que *2)* cet algorithme est sollicité par tout type de transporteurs circulant sur divers types de voies

(graphes) de transports (fleuves, rails, routes) sur l'ensemble du territoire sénégalais pendant un siècle. Ce qui implique une grande quantité de chemins à construire et nécessite un grand temps de calcul qu'il faut impérativement diminuer. Ainsi, nous avons proposé en annexe C une optimisation de l'utilisation de l'algorithme de Dijkstra. Ce qui a fait l'objet d'une publication [Mboup *et al.*, 2015b].

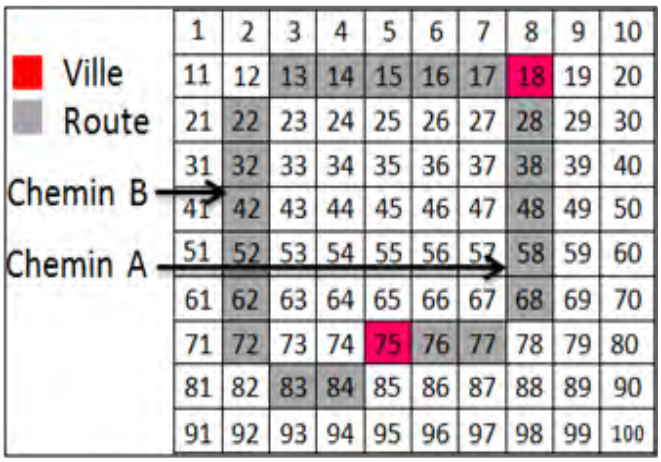

Figure 22 – Un exemple de grille avec un graphe de deux routes reliant deux villes.

II.A.5.c Déplacement sur le chemin construit

Par défaut un objet de type *Animal* se déplace de façon rectiligne du *soilCell* où il se trouve au *soilCell* où il veut aller. Après qu'un agent transporteur se soit construit un chemin, il lui suffit alors de se donner successivement comme destinations (intermédiaires) les cellules de son chemin, jusqu'à terme. Enfin entre deux cellules successives, il utilise le processus de déplacement de base de *Animal* [Baduel *et al.*, 2009].

II.A.6 Diffusion des rats via les véhicules

Puisque le rat noir est un rongeur particulier, dit commensal[11], nous avons étendu *C_Rodent* en *C_RodentCommensal* pour prendre en compte les aspects qui lui sont spécifiques. En tant que *C_Animal*, un agent rat commensal peut donc choisir une destination et s'y rendre [Baduel *et al.*, 2009]. Pour cela il peut percevoir son environnement sur un rayon paramétrable, et recenser les objets qui s'y trouvent. Si dans les objets recensés il y a un véhicule parqué alors le rat commensal, suivant une probabilité paramétrable, a le choix de monter ou pas dans le véhicule. Si le transporteur arrive à destination tous les rats descendent du véhicule. Ainsi les rats diffusent dans le pays à travers les véhicules. On note que les rats sont transportés sans conséquence pour les véhicules ou les transporteurs (a priori conformément à ce qui se passe dans la réalité). En plus de cela, une fois transporté, un rat ne remonte plus dans un véhicule du fait que c'est un phénomène rare et que ceci l'empêche de remonter dans le véhicule qui vient de le décharger.

[11] C'est-à-dire il vit associé à l'homme

II.B Implémentation et vérification

II.B.1 Gestion de l'espace

Nous avons implémenté et testé la gestion de l'espace et la prise en compte, par les agents, d'un éventuel changement d'échelle spatiale. Cette dernière (la taille des cellules de la grille *i.e.,* le niveau de détail avec lequel l'espace est discrétisé) peut être changée d'une simulation à une autre sans avoir à reprogrammer l'unité d'activité des agents. En effet les agents ne perçoivent pas directement leur voisinage suivant un ordre *n* fixe de cellules (*n* : l'ordre du voisinage de Moore). Ils calculent leur *n* en fonction de leur rayon de perception (qui est fixe) et de l'échelle spatiale du monde dans lequel ils se trouvent. Ceci sera valable au cas où un agent change de monde et donc d'échelle spatiale au cours d'une simulation de modèle multi-échelle (voir partie B.2, p. 70).

II.B.2 Gestion du temps

Nous avons aussi implémenté et testé la prise en compte d'un éventuel changement d'échelle temporelle : *(i)* en mettant en place un ordonnanceur permettant (avec une méthode *ordonancerActivitesAgents*) d'ordonnancer à des fréquences différentes l'exécution des unités d'activité des agents. Ces fréquences sont calculées (avec une méthode *ajoutListeAgent()*) en fonction de la durée de l'unité d'activité de chaque type d'agent et de l'échelle temporelle du monde où ils se trouvent. Ces fréquences sont recalculées à chaque fois que l'échelle temporelle du monde change. Cet ordonnancement est applicable à tout type d'agent. Cependant, pour les agents dont l'unité d'activité n'est pas une durée fixe (*i.e.* peut dépendre du temps), en particulier les transporteurs, nous avons aussi *(ii)* implémenté les méthodes qui permettent de convertir leur vitesse en distance par durée du tick monde et de les convertir à chaque fois que l'échelle temporelle du monde change en cours de simulation ou d'une simulation à une autre. Ceci sera valable au cas où l'agent change de monde et donc d'échelle spatiale au cours d'une simulation de modèle (voir partie B.2, p. 70).

II.B.3 Approche orientée connaissances-événements

II.B.3.a Elaboration du chronogramme

Pour élaborer le chronogramme nous avons recensé toutes les données, des différents thématiciens du projet, permettant la construction et la mise à jour de l'environnement sur un siècle. Ces données sont remises dans le chronogramme en respectant le format cité dans la section II.A.4. Pour cette partie (mono-échelle), nous avons eu à mettre en œuvre une dizaine de types d'instructions dont un exemple est présenté ci-après (Tableau 4) :

	DATE	X	Y	EVENT	VALUE1	VALUE2
1	05/01/1909	0	34	city	Dakar	
2	05/01/1909	0	35	city		
3	05/01/1909	1	34	city		
4	05/01/1909	1	35	city		
5	06/01/1909	0	34	population	Dakar	284.830
6	07/01/1909	0	34	rail		
7	07/01/1909	…	…	rail		
8	01/01/1914	0	34	train	3	city:Saint-Louis,Dakar
9	02/01/1923	0	34	road		
10	02/01/1923	…	…	road		
11	11/01/1923	0	34	rats	100	
12	01/01/1924	1	35	train	-1	city:Saint-Louis,Dakar
13	01/01/1924	0	34	truck	1.000	
14	…	…	…	…	…	…
15	01/01/1950	7	31	GNT-HEAVY		
16	01/01/1950	…	…	GNT-HEAVY		
17	02/01/1950	7	31	truck	30	area-type:GNT-HEAVY
18	…	…	…	…	…	…

Tableau 4 – Extrait de chronogramme.

- ligne 1, 2, 3 et 4 : après démarrage de la simulation en 1910, cette instruction correspond à un événement qui se passe le *05/01/1909* et qui consiste à prendre en compte l'existence et la position de la ville de *Dakar* couvrant 4 cellules (Figure 15). Les événements dont la date vient avant celle du début de la simulation sont tous exécutés en début de simulation ;
- ligne 5 : affecter à la ville de *Dakar* une taille de population de *284.830* personnes juste après sa création. Ceci est utile pour le choix des destinations des transporteurs suivant la grille gravitaire ;
- ligne 6 et 7 : création d'un chemin de fer qui s'étend sur plusieurs cellules y compris des cellules de *Dakar* et de *Saint-Louis*. Cet événement se passe le *07/01/1909* ;
- ligne 8 : instruction prise en compte le *01/01/1914*. Elle consiste à créer 3 agents conducteurs de train (sur une cellule de type *rail*) qui ne pourront aller que sur les villes de *Saint-Louis* et *Dakar* (*city:Saint-Louis,Dakar*). Ces transporteurs évoluent sur le graphe de type rail auquel la cellule (*0*, *34*) appartient ;
- ligne 9 et 10 : cet événement correspond à l'apparition de la route bitumée en *1923*. Cette instruction consiste donc à la construction d'une route qui s'étend sur plusieurs cellules ;
- ligne 11 : arrivée de *100 rats* à *Dakar* (un des comptoirs colonial du Sénégal) en *1923* ;
- ligne 12 : un train (-1) parmi ceux qui font le trajet *Saint-Louis-Dakar* (*city:Saint-Louis,Dakar*) doit être supprimé en *1924* ;

- ligne 13 : A cette même date (*01/01/1924*), *1.000* agents transporteurs de type camion doivent être créés sur la cellule (*0*, *34*). Après cette création, chaque transporteur détecte à partir de sa cellule courante le graphe dans lequel il peut évoluer. En effet les agents transporteurs sont créés sur une cellule appartenant au graphe sur lequel ils doivent effectuer leur transport. Puisque la liste de destinations n'est pas spécifiée dans la colonne *VALUES2* du chronogramme, alors ces transporteurs vont prendre comme destination possible la liste de toutes les villes reliées à leur graphe ;
- … ;
- ligne 15 et 16 : le simulateur ajuste par création ou par mise à jour l'étendue de la partie bassin arachidier intensément fréquenté (*GNT-HEAVY*, avec GNT: "Ground Nut Trade") ;
- ligne 17 : après la création ou la mise à jour du bassin arachidier, cette instruction permet d'y créer *30* camions qui réalisent leurs transports en son sein comme l'indique l'information *area-type:GNT-HEAVY* ;
- …

II.B.3.b Intégration du chronogramme dans le simulateur

Nous avons ainsi implémenté l'approche orientée connaissance-événement, et pour valider sa prise en compte dans le simulateur, nous avons testé un à un et étape par étape tous les types d'instructions. Nous avons ensuite englobé toutes les données dans un chronogramme complet (10.597 lignes). Nous avons lancé une simulation à partir de l'année 1910 jusqu'en 2010 avec un pas de simulation (tick) correspondant à un jour simulé (FIGURE 23). L'environnement se construit progressivement à partir des données du chronogramme (prolongement et bitumage des routes, nouvelle ville, progression du bassin de l'arachide etc.). Tout type de véhicule parvient à passer par son type de route et par le plus court chemin tout en prenant en compte l'évolution de l'environnement. Par exemple si une nouvelle ville ou une nouvelle route est construite alors les agents transporteurs concernés l'apprennent et la prennent en compte pour s'y rendre (ville), ou pour y passer (plus court chemin). Le modèle prend bien en compte les transporteurs ne devant faire un trajet qu'entre deux ou un certain nombre de villes, se cantonnant à une zone (*e.g.*, un bassin arachidier) ou choisissant parmi toutes les villes reliées à leur graphe.

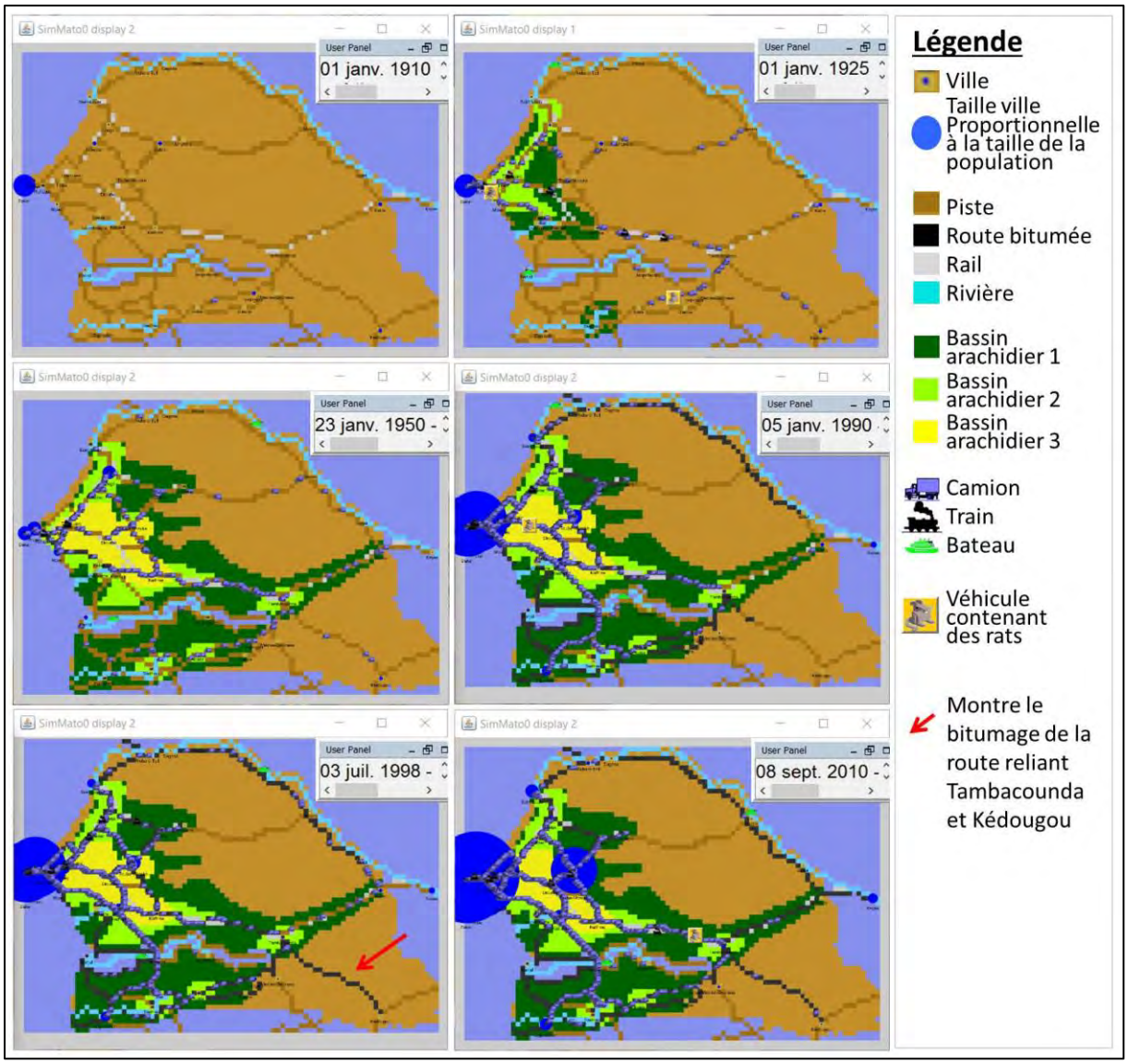

Figure 23 – Évolution de l'environnement et des voies de transport.

Les bassins arachidiers ainsi que les voies de transport peuvent être superposés à certains endroits et donc les uns peuvent cacher les autres. Ainsi, l'affichage se fait suivant un ordre de priorité : transporteurs, villes, rivières, pistes/routes bitumées, rails. L'activité des transporteurs est plus dense (il y a plus de transporteurs) dans le bassin arachidier 3 suivi du bassin arachidier 2 et ensuite du bassin arachidier 1.

II.B.4 Choix de la destination suivant la grille gravitaire

Nous avons implémenté l'algorithme que nous avons proposé pour le choix des destinations suivant la grille gravitaire. Pour valider le fait que plus une ville est peuplée, plus les transporteurs s'y rendent, nous avons ainsi élaboré un chronogramme de test contenant 11 villes du Sénégal avec des tailles de populations arbitraires (Tableau 5). Pour chaque ville nous avons calculé la probabilité pour qu'un transporteur s'y rende (probabilité réelle) : taille de la population d'une ville divisée le cumule des tailles des populations des villes (plus une ville est peuplée plus les transporteurs s'y rendent). Avec le chronogramme de test, nous avons lancé une simulation sur 5 ans avec une dizaine de transporteurs en comptant le nombre

de visites pour chaque ville. A la fin de la simulation, pour chaque ville nous avons recalculé la probabilité à laquelle un transporteur s'y est rendu (probabilité simulée) : nombre de visites pour chaque ville divisé par le nombre total de visites pour toutes villes. Les résultats sont présentés dans le Tableau 5. Le test de Student (P(T<=t) bilatéral = 1) montre que les probabilités réelles et les probabilités simulées sont statistiquement les mêmes.

Après avoir choisi une destination suivant la grille gravitaire, le transporteur se construit un chemin (un plus court ou un meilleur chemin) entre sa position courante et sa destination et ensuite le parcourt.

Ville	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**V8**	**V9**	**V10**	**V11**
Population	100	41	37	33	25	21	0	29	17	13	9
Proba réelle	0,31	0,13	0,11	0,10	0,08	0,07	0,00	0,09	0,05	0,04	0,03
Proba simulée	0,30	0,13	0,11	0,09	0,08	0,07	0,00	0,10	0,05	0,04	0,03

Tableau 5 – Tableau pour tester le choix de destination suivant une grille gravitaire. Proba : probabilité pour qu'un transporteur se rende dans une ville. Le test de Student (P(T<=t) bilatéral = 1) montre que les probabilités réelles et les probabilités simulées sont statistiquement les mêmes.

II.B.5 Diffusion des rats via les véhicules

Les objets rats de la classe *C_RodentCommensal* parviennent à percevoir les véhicules, y montent et se font transporter d'une ville à une autre. Pour tester la sensibilité de la diffusion des rats au nombre de transporteurs et de rats dans les villes, nous avons construit arbitrairement un autre chronogramme de test de sorte à varier le nombre de véhicules et de rats dans le temps (Figure 24) : Les pics de population correspondent à l'ajout arbitraire de 500 rats chaque année dans la simulation (proxy pour l'apport de rats par bateaux occidentaux). La population de rats décroit du fait que les rats dans cette simulation de test ne se reproduisent pas et meurent après un certain temps (c'est la sensibilité à la diffusion qui est testée et non pas la reproduction des rats). Avant 1926, les seuls transports par fleuve et rail ne permettent qu'une diffusion limitée des rongeurs dans le pays. L'augmentation du nombre de transporteurs par route en 1926 conduit à une augmentation sensible du nombre de rats transportés et leur diffusion des villes où leur population est forte vers celles où il n'y en avait pas.

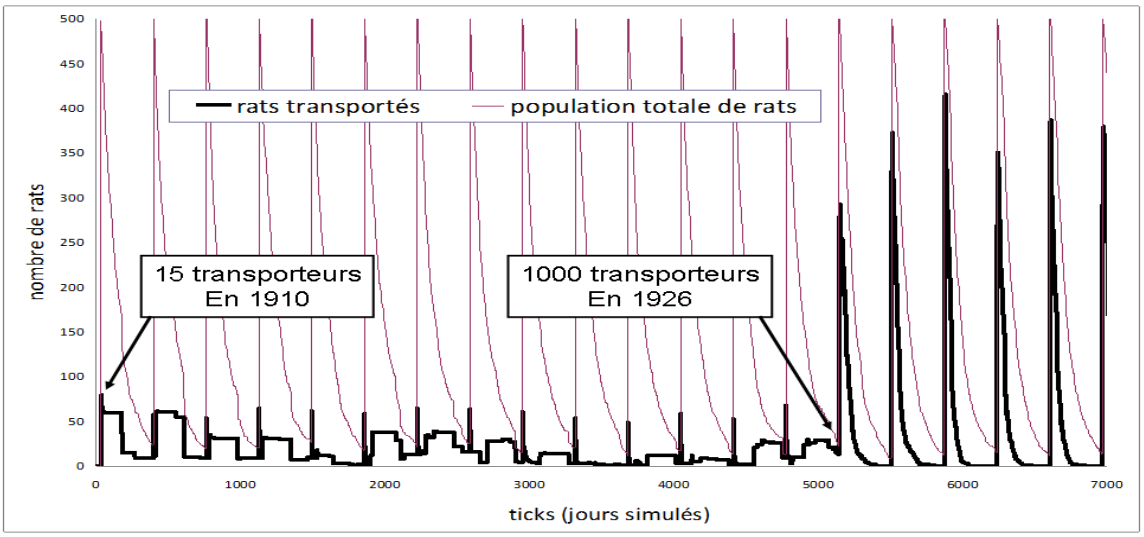

Figure 24 – Evolution du nombre de rats transportés simulée sur un siècle.
La figure montrant l'effet du nombre de transporteurs sur la diffusion des rats. Pour le test, la probabilité (paramétrable) des rats de monter dans un véhicule s'il le perçoit se situe entre 90 et 100%.

II.C Conclusion

Nous avons montré comment nous avons construit et évalué un modèle de monde multi-agents orienté connaissance et événement. Un modèle permettant de simuler le trafic routier, fluvial et ferroviaire sur l'ensemble du Sénégal avec une granularité spatiale et temporelle dépendant du niveau de détail que l'on choisit. Les agents peuvent s'adapter à un changement d'échelle spatiale et/ou temporelle d'une simulation à une autre et au sein d'une même simulation. En effet on peut, en cours de simulation pour un monde, changer l'échelle temporelle et, dans le cas du modèle multi-échelle, les agents pourront changer de monde et donc d'échelles spatiale et temporelle. Le modèle mono-échelle comprend un ensemble flexible d'objets transporteurs humains avec divers types véhicules. Ces derniers circulant d'une ville à une autre sur différents types de voies en empruntant les plus courts chemins possibles. Les rats peuvent monter, descendre et se faire ainsi transporter d'une ville à l'autre par les véhicules et tout ceci en prenant en compte l'évolution de l'environnement et des flux de véhicules.

Ce simulateur peut à présent être réutilisé pour d'autres études présentant des phénomènes similaires. Il suffira de *(i)* définir un nouveau protocole, *(ii)* changer le contenu du chronogramme (qui permet la construction de l'environnement, ses mises à jour et la création des agents qui y évoluent) *(iii)* définir les nouvelles échelles spatiale et temporelle correspondant au niveau de détail dans lequel s'inscrit le nouveau phénomène que l'on veut étudier et enfin *(iv)* faire des analyses de sensibilité des paramètres clés du modèle.

III. Modèle Décennal (Transposition de Centennal)

Dans cette deuxième phase nous passons à la modélisation de la diffusion actuelle du rat noir à l'échelle régionale, sur les vingt dernières années et l'effet de modifications de l'environnement sur ce processus (voir la section B1.Phase 2). On peut ainsi noter que pour cette phase, nous pouvons simplement et aisément réutiliser le simulateur que nous avons présenté précédemment. Il nécessaire pour cela :

1. de construire un nouveau chronogramme pour cette étude. Un chronogramme qui intègre des aspects qui n'apparaissaient pas dans cette zone (de Tambacounda et de Kédougou) à l'échelle *centennale* tels que : de nouvelles localités (appréhendées comme des villes), de nouvelles routes, de nouvelles zone de commerce ou d'intérêts (filière coton, le développement de l'orpaillage, etc.) ;
2. d'adopter une nouvelle échelle spatiale, c'est-à-dire la granularité de la grille de l'environnement (cellules de 2 km de côté au lieu de 7,5 km dans le modèle *centennal*), initialisé avec un fichier raster et mis à jour avec les instructions du chronogramme ;
3. d'adopter une nouvelle échelle temporelle (durée de pas de simulation de 1 heure). Avec une simulation qui commence en 1990 pour une étendue temporelle d'une vingtaine d'années.

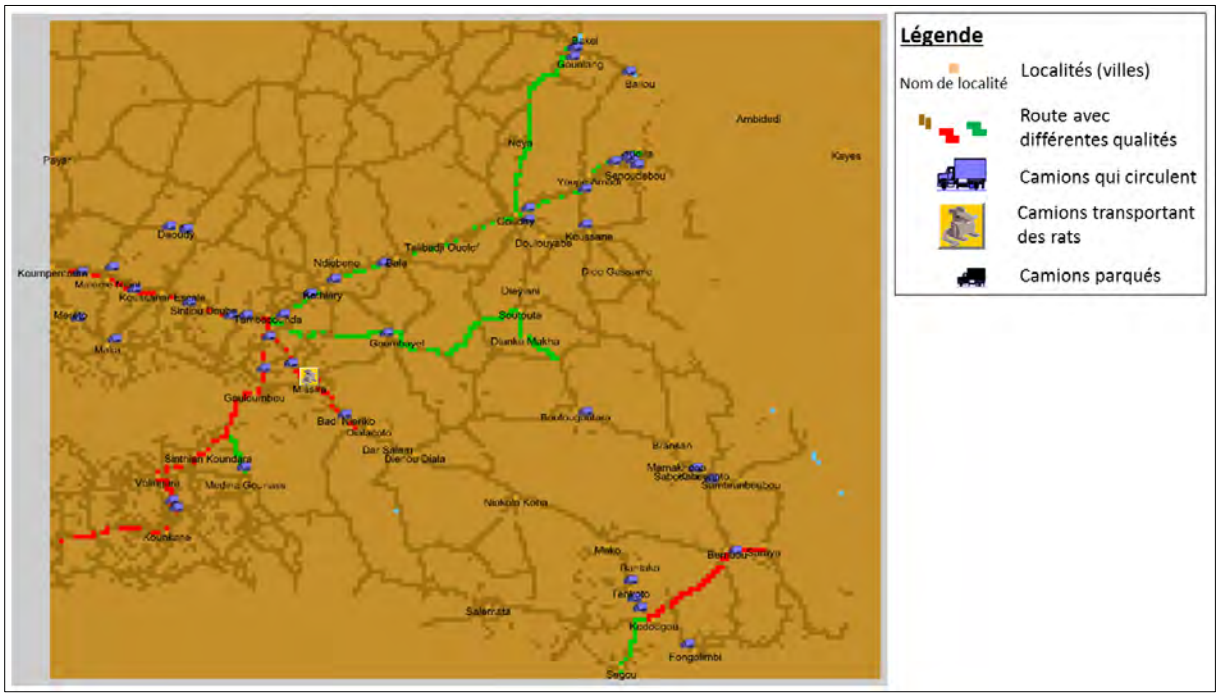

Figure 25 – Simulation du modèle décennal.

Ce modèle permet particulièrement la représentation des communications (dont les flux d'entrée/sortie de la région) et des contextes sociaux et commerciaux (voir Figure 25). Cependant, nous n'avons pas pu disposer de toutes les données nécessaires (*e.g.,* concernant la filière coton, le développement de l'orpaillage etc.) pour bien alimenter le chronogramme et donc le modèle afin de passer à sa calibration avec les données spécifiques.

IV. Modèle Contact (Transposition de Centennal)

Dans cette troisième phase nous passons à la modélisation et la simulation de la diffusion du rat noir au sein même des systèmes urbains en fonction des paramètres sociaux-environnementaux et de la configuration des espaces (gares routières, marchés, types d'habitation, connectivité, etc.). Comme avec la deuxième phase on peut ici aussi réutiliser le modèle *centennal*. Il est nécessaire pour cela :

1. de construire un nouveau chronogramme contenant les données nécessaires à la construction et à la mise à jour de l'environnement de simulation pour cette phase ;
2. d'adopter une nouvelle échelle spatiale, c'est-à-dire discrétiser l'espace de la ville de Kédougou (Figure 26) en des cellules de 38 m de côté, initialisé avec un fichier raster et mis à jour avec les instructions du chronogramme ;
3. d'adopter une nouvelle échelle temporelle avec une durée de pas de simulation de 5 minutes, une simulation qui commence en 2010 pour une étendue temporelle d'un mois.

Figure 26 – Simulation du modèle Contact.

Le modèle *centennal* étendu ici ne permet pas d'étudier les modalités de transmission des zoonoses par le rat en fonction de la nature de l'environnement urbain. Il aurait été nécessaire d'identifier et d'intégrer dans le modèle, pour cette phase, les facteurs clés de la transmission de maladie de rat à l'homme : mettre de nouveaux agents tels que des moustiques qui pourraient s'impliquer à la transmission de maladie de rats à l'homme. Le modèle transposé à cette échelle permet cependant d'étudier l'approche multi-échelle.

Nous pouvons à présent envisager d'emboiter ces trois phases d'échelles spatiales et temporelles différentes dans un même modèle multi-échelle. Ce qui fait l'objet du chapitre suivant.

B.2 : Modèle multi-échelle

Dans cette section nous proposons un meta-modèle générique pour la modélisation à base d'agents de système géographique multi-échelle alimenté par des données (sous forme d'événements historiques) permettant la construction et la mise à jour de l'environnement au fil du temps. Nous proposons ainsi un meta-modèle multi-monde orienté connaissances-événements (données d'entrées historiques). Les mondes sont emboités, interagissent entre eux et disposent, chacun, d'une échelle spatiale et d'une échelle temporelle qui leurs sont propres. Ainsi ce meta-modèle permet particulièrement d'emboiter dans une même simulation les 3 mondes décrits précédemment et permet la communication entre ces mondes. En effet le modèle de monde prévoit le fait que les agents puissent passer d'un monde à l'autre et s'adapter au changement d'échelle spatiale et/ou temporelle associé. Ce travail a fait l'objet de la publication Mboup *et al.* [2017].

I. Projet de modélisation multi-échelle

Toute modélisation géographique se situe à une certaine échelle d'espace et de temps se caractérisant par une extension (un domaine d'espace et de temps) et une granularité (discrétisation) [Langlois, 2009]. Pour établir un meta-modèle multi-monde multi-échelle pour la modélisation géographique, il faut donc retourner sur les notions fondamentales de la modélisation à base d'agents que sont, l'environnement (avec l'échelle spatiale), l'ordonnanceur (avec l'échelle temporelle) et l'agent.

I.A Définitions

I.A.1 Environnement et multi-échelle spatiale

Comme nous l'avons précisé dans la section de modélisation mono-échelle (B1.II.A.2, p. 46), nous appelons échelle ou granularité spatiale la taille des cellules de la grille (TCG) d'un environnement.

Afin d'élaborer un modèle multi-échelle, pour ce qui concerne l'espace, nous avons choisi de mettre en place un espace continu universel et plusieurs grilles de granularités différentes (voir A.VII.B.3.b, p. 20 et A.VIII.F.1, p. 36). Chaque grille, représentant l'environnement pour un monde, constitue une discrétisation d'une carte géographique qui représente une partie du domaine global que l'on veut étudier (Figure 27).

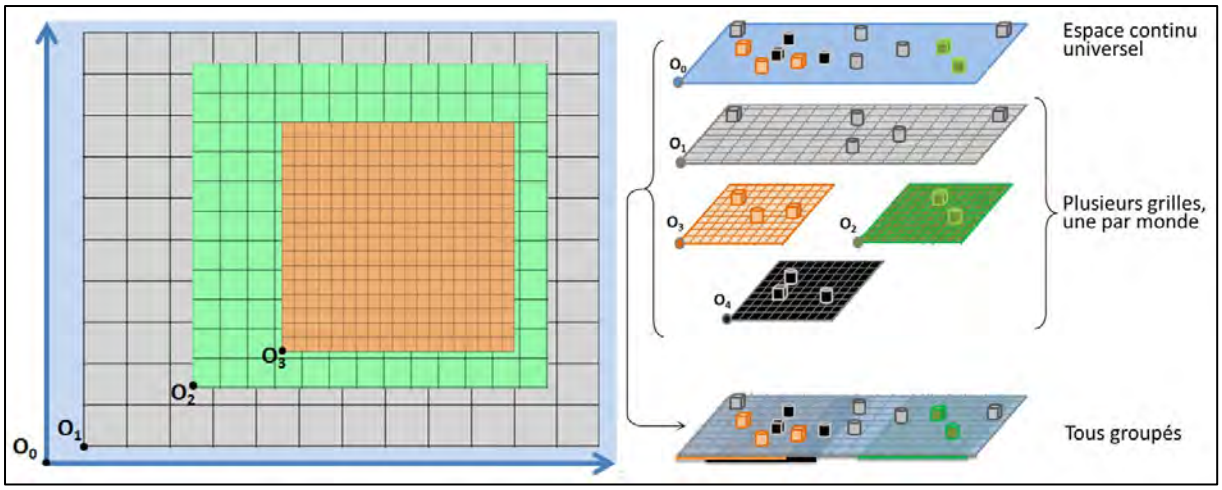

Figure 27 – Principe de construction du modèle multi-échelle spatiale.
Les cubes et cylindres représentent des agents. Ces derniers sont tous référencés dans l'espace continu et dans la grille du monde où ils se trouvent. Toutes les grilles appartiennent à un même plan, celui de l'espace continu (voir I.C, p. 76).

I.A.2 Multi-échelle temporelle

A un référentiel d'espace il est nécessaire d'associer un référentiel de temps qui lui est compatible en termes de discrétisation et d'extension [Langlois, 2009]. Un modèle multi-échelle temporelle implique donc de disposer d'une échelle temporelle pour chaque échelle spatiale. Nous rappelons ainsi que nous appelons échelle ou granularité temporelle la durée du pas de simulation (DPS) pour un monde. Nous appelons "tick monde" ce pas de simulation.

I.A.3 Monde et multi-monde

Nous considérons qu'un monde est un modèle classique complet, autonome, qui peut fonctionner seul et peut interagir avec d'autres mondes. Ainsi un monde :

- est associé à l'espace continu (qui est universel) muni de la métrique euclidienne ;
- dispose de son propre environnement discrétisé en une grille dont la taille des cellules représente l'échelle spatiale du monde. La grille est munie de la métrique de Manhattan (pour le voisinage de von Neumann) et/ou de la métrique du max (pour le voisinage de Moore) ;
- dispose d'un pas de simulation (tick monde) dont la durée représente l'échelle temporelle du monde et un *ordonnanceur* (voir la section I.A.3.a suivante) dont la fonction est de donner la main aux agents du monde afin qu'ils réalisent leur unité d'activité (Figure 29a).

Un modèle multi-monde multi-échelle (Figure 28 et Figure 29b) *correspond donc dans ce travail à plusieurs mondes d'échelles spatiales et temporelles différentes et pouvant interagir.*

La gestion de plusieurs échelles temporelles simultanées est réalisée par un ordonnanceur général que nous étudions dans les sections I.A.4, p. 74, I.D, p. 79 et II.B, p. 93. Tandis que l'interaction entre les mondes est étudiée dans les sections I.E, p. 86 et II.C, p. 93.

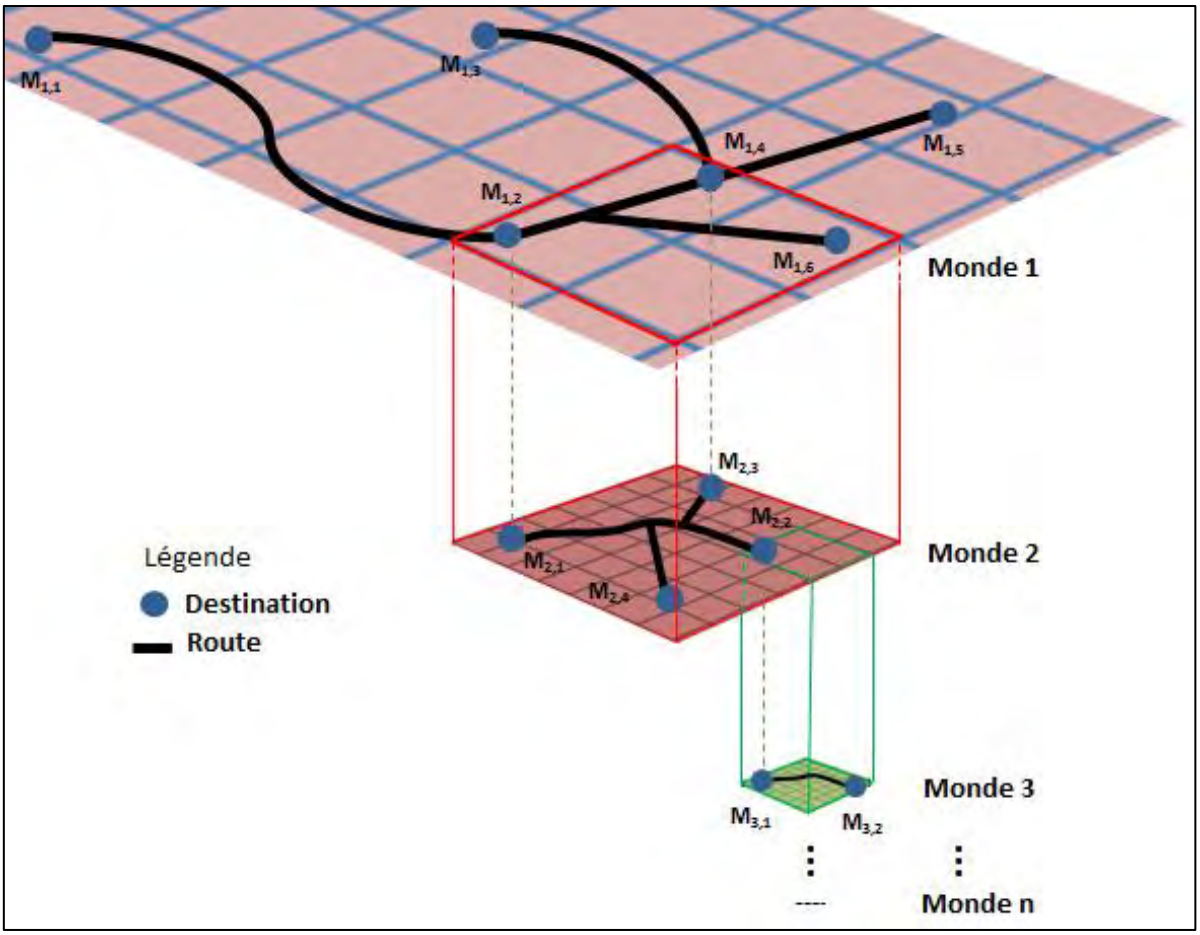

Figure 28 – Représentation 3D d'un multi-monde à plusieurs échelles spatiales.
Les mondes sont emboités et se trouvent tous sur le même plan. Nous ne les avons mis visuellement sur des plans différents que pour avoir une meilleure représentation des niveaux de détails et des différentes échelles spatiales. Nous abordons l'aspect mondes emboités au point I.A.3.a suivant et à la section I.E.

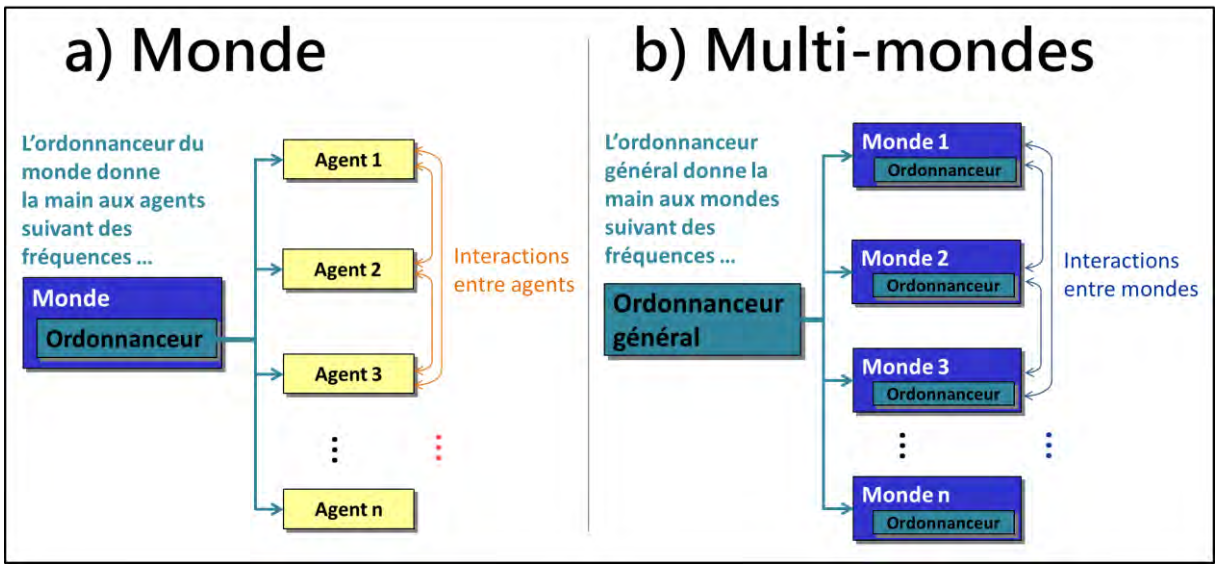

Figure 29 – Monde et multi-monde à plusieurs échelles temporelles.
L'interaction entre mondes est une composante essentielle abordée au point I.E et II.C.

I.A.3.a Mondes emboités

Un monde peut aussi être un modèle représentant une partie d'un autre monde au sein de laquelle on souhaite entrer plus en détail tout en prenant en compte le fait que les mondes s'alimentent mutuellement. Dans ce cas de figure les mondes sont dit emboités (voir Figure 30 ci-dessous).

I.A.3.b Mondes en cascade

Tous les mondes dans une simulation multi-monde peuvent démarrer en même temps. Cependant, on peut aussi avoir une simulation multi-monde en cascade. C'est-à-dire, il peut arriver qu'une simulation pour un monde (monde 1) démarre avant les autres ; après un certain temps de simulation de ce premier monde et à partir des données qu'il produit sur une zone, une deuxième simulation (avec une échelle plus petite) commence (monde 2) ; après un certain temps de simulation du deuxième monde et à partir des données qu'il produit sur une zone, une troisième simulation (avec une échelle encore plus petite) commence (monde 3) et ainsi de suite (voir Figure 30 ci-dessous).

Dans notre étude de cas, les modèles sont en cascade et emboités (voir I, p. 42).

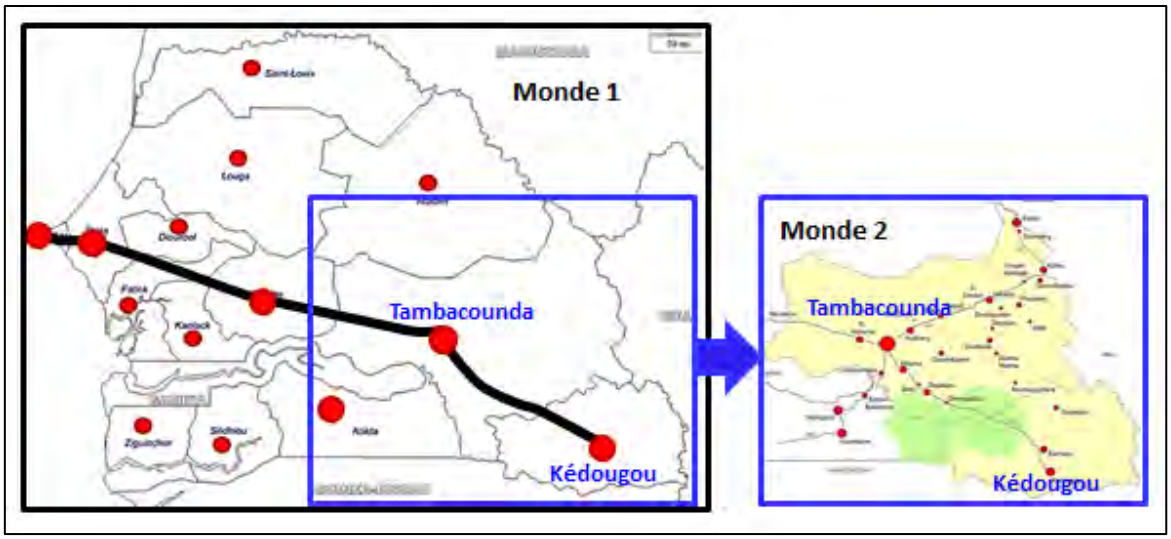

Figure 30 – Mondes emboités en cascade.
Avec cet exemple, la partie encadrée en rouge du monde 1 est représentée avec une échelle spatiale plus petite et un niveau de détail plus élevé au sein du monde 2. Les deux mondes s'allimentent mutuellement ; la simulation du monde 1 débute en 1910 (phase 1) alors que celle du monde 2 débute quelques années plus tard (en 2000) à partir des données produites par la phase 1.

I.A.4 Ordonnanceur et ordonnanceur global

Comme nous l'avons vu à la section II.A.3.c, p. 51, l'ordonnanceur du monde suivant des fréquences, donne la main aux agents de son système pour qu'ils réalisent leurs unités d'activité (Mono-échelle temporelle, Figure 29a). La gestion de la multi-échelle temporelle (Figure 29b) est effectuée par un ordonnanceur de niveau supérieur nommé *ordonnanceur général* (*Global Scheduler*) comme représenté au niveau de l'architecture du meta-modèle à la Figure 31.

I.B Diagramme UML et description du meta-modèle

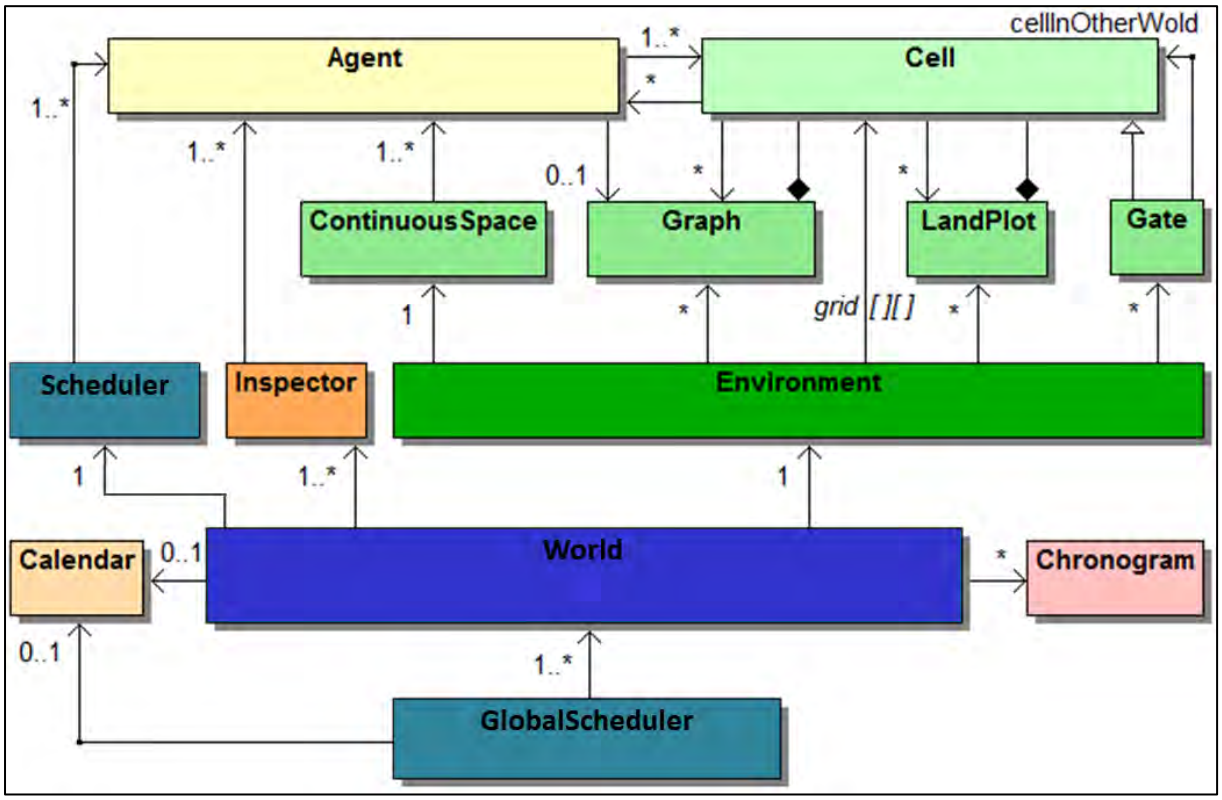

Figure 31 – Diagramme de classes UML du meta-modèle multi-monde multi-échelle spatiale et temporelle.

– Comme avec *A_Protocol*, *World* est la classe qui permet de créer des mondes (mono-échelle, vu dans la section II, p. 43). En plus des caractéristiques des mondes mono-échelle que nous avons présentés dans la partie B.1, p. 42, l'environnement de *World* peut avoir des *portes* d'entrées-sorties (objets de la classe *Gate*) permettant aux agents (*Agent*) de passer d'un monde à un autre. Ainsi un *gate* est une cellule (*Cell*, une classe qui se présente comme les classes *C_SoilCell* et *C_SoilCellGraphed* du Schéma UML du modèle *centennal*) qui contient une autre cellule *C*, du même type, appartenant à un autre monde ; en effet, comme dans le cas d'une *soilCell*, un *gate* peut avoir plusieurs types en particulier route, rail ou rivière. Ainsi, si par exemple un modèle doit prendre en compte des agents transporteurs utilisant un type de graphe et devant passer par un *gate* pour aller dans un autre monde, alors la cellule *C* du *gate* doit appartenir, dans l'autre monde, à un graphe du même type que celui dans lequel il se déplace. Les *gates* sont les liens (canaux) entre les mondes (Figure 37). Nous utilisons l'approche orientée connaissance-événement avec le chronogramme (*Chronogram*) pour déterminer les propriétés des *gates* en l'occurrence leur position dans un monde et celle de leur cellule *C* appartenant à un autre monde (voir Tableau 6).
  Supposons que cette portion de chronogramme (Tableau 6) soit associée au monde 1 par exemple, la ligne n représente alors l'instruction de création d'une porte à la cellule de coordonnées (x1, y1) du monde 1. Cette porte (qui peut optionnellement avoir un nom)

permet de passer, du monde 1, à un autre monde (*e.g.* monde 2) dont la cellule d'arrivée *C* a pour coordonnées (x2, y2). On peut ainsi remarquer que ce *gate* de coordonnées (x1, y1) du monde 1 est une porte de sortie. Le fait qu'il soit une porte d'entrée ou pas est défini dans le chronogramme d'un autre monde, par exemple celui du monde 2. Il suffit de respecter le même principe et que la cellule d'arrivée *C* au monde 1 ait pour coordonnées (x1, y1), qui représente la porte de sortie précédente du monde 1.

Ligne	**Date**	**X**	**Y**	**Event**	**VALUE1**	**VALUE2**
n	jj/mm/aaaa	x1	y1	gate	nomAutreMonde:x2,y2:NomDuGate	type
n+1	…	…	…	…	…	…

Tableau 6 – Evénement concernant un *gate* dans un chronogramme.

– L'*ordonnanceur général* (*GlobalScheduler*) du meta-modèle est comme l'ordonnanceur dans les mondes. La seule différence est que ce premier ordonnance la façon de donner la main aux mondes eux-mêmes pour qu'ils se gèrent (voir la partie B.1, p. 42) ; en effet il connait la liste des mondes et leur échelle temporelle. Ainsi, ce n'est plus l'ordonnanceur du monde qui est exécuté à chaque pas de simulation (ici, par l'ordonnanceur interne de Repast Simphony (avec une annotation @Scheduler)), mais plutôt l'ordonnanceur général. Ce pas de simulation est appelé tick général, global, absolu ou de référence [Vo *et al.*, 2012]. Il a une durée qui représente elle aussi l'échelle temporelle générale, globale, absolue ou de référence. Elle peut être égale, plus petite ou plus grande que celle d'un monde et peut même varier en cours de simulation (voir la section I.D). Nous reviendrons sur cette classe dans la section I.D.2 de cette partie.

I.C Gestion de plusieurs échelles spatiales

Comme nous l'avons déjà dit, la multi-échelle spatiale porte, en partie, sur le fait de mettre en place un espace continu universel et plusieurs environnements d'échelle spatiale et de niveau de détail différents (voir I.A.1). Ainsi après avoir mis en place un modèle mono-échelle spatial :

(1) l'environnement dispose d'un référentiel géométrique déterminé principalement par une origine, une extension et une granularité. Ces dernières étant déterminées par le choix du domaine et le pas de discrétisation correspondant à l'échelle spatiale et au niveau de détail choisi (voir modèle mono-échelle *B1*.II.A.2, p. 46) ;

(2) les agents sont préparés à un éventuel changement d'échelle spatiale qui pourrait survenir au cours de la simulation (voir la section B1.II.A.2.a, p. 47).

Il reste alors à bien repérer les mondes par rapport à l'espace continu.

L'espace continu est un plan cartésien muni d'un repère orthogonal $(O_0, \vec{\imath}, \vec{\jmath})$ que nous appelons le repère absolu. O_0 est son origine et $\vec{\imath}$ et $\vec{\jmath}$ ses vecteurs unitaires avec $\|\vec{\imath}\| = \|\vec{\jmath}\|$. Chaque monde a aussi un repère relatif $(O_i, \vec{\imath}_i, \vec{\jmath}_i)$, dont O_i est l'origine et $\vec{\imath}_i$ et $\vec{\jmath}_i$ les vecteurs unitaires avec $\|\vec{\imath}_i\| = \|\vec{\jmath}_i\|$. Cette norme est aussi égale à la TCG du monde et les vecteurs unitaires $\vec{\imath}_i$ et $\vec{\jmath}_i$ sont parallèles respectivement à $\vec{\imath}$ et $\vec{\jmath}$ du repère absolu (Figure 32). Grace à ces repères, nous pouvons positionner et translater à notre guise les mondes, sur le 1er

quadrant[12] du plan cartésien muni du repère absolu. Ce qui est utile pour l'emboîtement spatial des mondes et la perception des agents se trouvant dans un autre monde. En effet il est nécessaire de bien positionner un monde par rapport à la zone qu'il représente dans un autre monde afin que les agents puissent recenser, à travers les mondes, les objets/agents qui se trouvent dans leur disque de perception (voir I.E.2 de cette partie).

Ainsi, un agent ou objet dans un monde peut être localisé de deux manières différentes : *(1)* sur l'espace continu universel qui fournit des coordonnées en nombre réels et *(2)* sur la grille de l'environnement de ce monde qui fournit des coordonnées discrètes (en nombre entier). Si nous prenons l'exemple de la Figure 32, les agents A1, A2 et A3 respectivement des mondes 1, 2 et 3 ont pour coordonnées respectivement sur l'espace continu et sur la grille de leur monde :

- A1 : (2 ; 2,6) et (2 ; 4) ;
- A2 : (4 ; 2,6) et (3 ; 3) ;
- A3 : (6,25; 2,6) et (9 ; 3).

Si nous voulons avoir des coordonnées avec une unité spatiale (mètre par exemple), il suffit de multiplier les coordonnées dans l'espace continu par la norme des vecteurs unitaires du repère absolu.

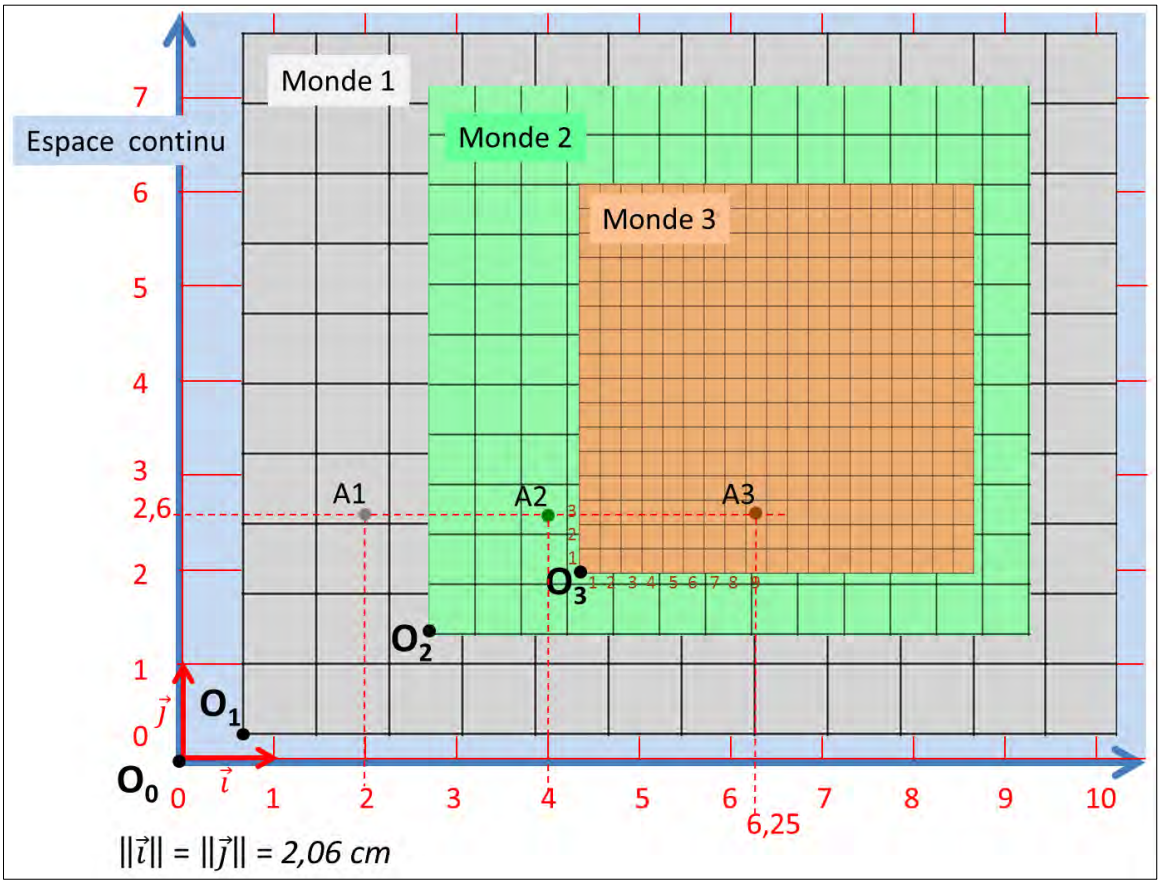

Figure 32 – Repère absolu de l'espace continu et trois mondes d'échelles spatiales différentes.

[12] http://www.cslaval.qc.ca/sitsatlll/maths2003/cartesien.html
http://www.math.uqam.ca/~progiciels/labelleetlesbetes/notesdecours/notesdecours.pdf

I.C.1 Retrouver les coordonnées discrètes à partir des coordonnées continues

Soient :

- $\|\vec{\imath}\|$ ou $\|\vec{\jmath}\|$ la norme d'un vecteur unitaire du repère absolu ;
- TCG_i la taille des cellules de la grille du monde i (échelle spatiale);
- *rapportSpatial* le rapport TCG_i sur $\|\vec{\imath}\|$;
- *coord_Uec(O_i)* les coordonnées dans l'espace continu de l'origine O_i du repère d'un monde i ;
- *coord_Uec(A)* les coordonnées dans l'espace continu de l'élément *A* ;
- *coord_Ugrille(A)* les coordonnées discrètes de l'élément *A* ;
- $partie\_entière\ (x, y)$ la partie entière du nombre x et du nombre y ;
- $arrondi\_3\ (x, y)$ l'arrondi à 3 chiffres après la virgule du nombre x et du nombre y[13].
- $(x1, y1) - (x2, y2) = (x1 - x2, y1 - y2)$.
- $\frac{(x,y)}{RAPPORT} = \left(\frac{x}{RAPPORT}, \frac{y}{RAPPORT}\right)$

Alors :

$$coord\_Ugrille(A) = partie\_entière\left\{arrondi\_3\left[\frac{coord\_Uec(A) - coord\_Uec(O_i)}{rapportSpatial}\right]\right\}$$

Exemple (sur l'espace continu le monde 3):

- $\|\vec{\imath}\| = \|\vec{\jmath}\|$ = *2,06 cm ;*
- TCG_3 = *0,4 cm ;*
- *rapportSpatial* = $TCG_3 / \|\vec{\imath}\|$ = *0,194*
- $coord\_Uec(A3) = (6{,}25\ ;\ 2{,}6)$;
- $coord\_Uec(O_3) = (4{,}35\ ;\ 2)$.

$$coord\_Ugrille(A3) = partie\_entière\left\{arrondi\_3\left[\frac{(6{,}25\ ;\ 2{,}6) - (4{,}35\ ;\ 2)}{0{,}194}\right]\right\}$$

$$coord\_Ugrille(A3) = partie\_entière\left\{arrondi\_3\left[\frac{(1{,}9\ ;\ 0{,}6)}{0{,}194}\right]\right\}$$

$$coord\_Ugrille(A3) = partie\_entière(9{,}793\ ;3{,}092)$$

$$coord\_Ugrille(A3) = (9\ ;3)$$

Ainsi nous calculons les coordonnées discrètes à partir des coordonnées continues. Ce calcul est souvent nécessaire pour savoir si un agent doit changer de cellule courante ou pas. En effet les agents se déplacent en premier lieu sur l'espace continu. De ce fait leurs coordonnées suivant ce dernier sont toujours à jour. A la fin de leur unité de déplacement (sur l'espace continu), les agents calculent à partir de leur position courante (sur l'espace continu) les coordonnées discrètes correspondantes (sur la grille). Si ces dernières sont différentes de

[13] Nous prenons l'*arrondi_3* car il peut arriver que l'on ait par exemple $x = 1$ et que l'on obtienne $x =$ 0,99999999999999 à cause des divisions. La partie entière de ce dernier serait donc 0 alors que si on prenait l'arrondi à 2 ou 3 chiffres après la virgule avant de prendre la partie entière, on aurait 1,000 et ensuite l'entier 1. Par contre l'arrondi par excès simple ne marcherait pas, voir l'exemple suivant.

celles de leur cellule courante (qui n'est pas encore mise à jour) alors ils mettent à jour cette cellule courante en prenant celle dont les coordonnées sur la grille correspondent aux coordonnées discrètes calculées.

Pour mettre en œuvre ceci, nous avons prévu *(i)* dans classe *GlobalScheduler* un champ *normVectUnitaire* qui contient la norme du vecteur unitaire du repère absolu (de l'espace continu) et *(ii)* dans la classe *Environment* un champ *origineCoord_Uec* qui contient les coordonnées par rapport à l'espace continu du point d'origine du monde, un champ *rapportSpatial* qui contient le rapport de la *normVectUnitaire* sur le *TCG* et une fonction *coord_Ugrille()* qui prend en paramètre un agent ou un objet et calcule ses coordonnées discrètes à partir de celles sur l'espace continu.

I.D Gestion de plusieurs échelles temporelles

Notre objectif dans cette section est, au niveau multi-monde, *1)* de coordonner, en fonction d'une échelle temporelle gloabale, des échelles temporelles (de mondes) hétérogènes quelque soit leur durée ; *2)* de permettre la possibilité de changer l'échelle temporelle globale et des mondes, même en cours de simulation, sans avoir à reprogrammer (en ce qui concerne le temps) les mondes ou l'ordonnanceur global.

La multi-échelle temporelle est principalement géré par l'ordonnanceur gobal. C'est lui qui se charge de donner, avec des fréquences N_i, la main aux "mondes" afin qu'ils donnent à leur tour la main aux agents qu'ils contiennent avec un autre ordonnancement que nous avons déjà étudié dans la partie modèle mono-échelle.

Les fréquences N_i sont déterminées en fonction de la durée des ticks mondes et de la durée du tick global (DTG). Ainsi, on ne peut avoir que quatre situations possibles ; selon que la durée d'un tick monde (DTM) est :

a) un multiple de la DTG ;
b) supérieure à la DTG sans être son multiple ;
c) un diviseur de la DTG ;
d) inférieure à la DTG sans être son diviseur.

L'approche que nous proposons marche aussi bien pour le temps d'observation que pour le temps de simulation. En effet Ratzé *et al.* [2007] distinguent le temps de simulation, du temps d'observation.

I.D.1 Ordonnancement global

Pour que l'ordonnanceur (global) puisse prendre en compte ces quatre cas de figure, nous proposons de les étudier un à un avec des exemples à l'appui. Nous allons nous baser ainsi sur l'exemple du Tableau 7 ci-dessous correspondant respectivement aux situations a), b), c) et d). Nous allons aussi utiliser les abréviations suivantes :

DTM_{Mi} : la durée du tick monde pour un monde M_i,
TG : tick global, celui de l'ordonnanceur général (global),
DTG : la durée du *TG*,

N1 : le nombre de *TG* au bout duquel le monde M_i reçoit la main pour s'exécuter (situations où DTM_{Mi} est supérieure à *DTG*).
N2 : le nombre de fois que le monde M_i reçoit la main à chaque *TG* pour s'exécuter (situations où DTM_{Mi} inférieure à *DTG*).

Mondes M_i	M_1	M_2	M_3	M_4
DTM_{Mi}	DTM_{M1} = 12 h	DTM_{M2} = 8 h	DTM_{M3} = 3 h	DTM_{M4} = 4 h

Tableau 7 – Durée du tick monde pour chaque monde Mi.

DTG = *6 h* (pris arbitrairement ici, mais qui peut prendre n'importe quelle autre valeur en particulier le PGCD ou un des DTM).

I.D.1.a Situation où DTM_{Mi} est un multiple de DTG

Nous prenons l'exemple de M_1 :

$DTM_{M1} = 12h$;
$N1 = DTM_{M1} / DTG = 2$.

Le monde M_1 doit donc être exécuté *1 fois* tous les *2 TG* (*N1 fois*), c'est-à-dire quand la valeur du *compteur de TG* est un multiple de *N1*. Nous pouvons illustrer ceci avec la Figure 33 suivante :

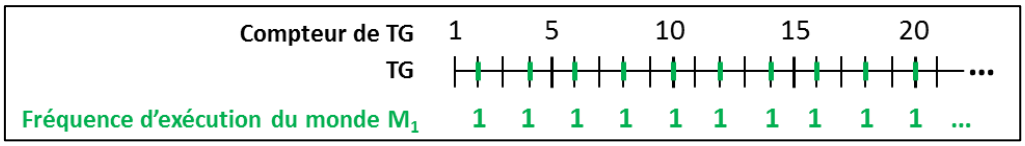

Figure 33 – Situation où DTM_{Mi} est un multiple de DTG.

Prenons par exemple le cas où $N1 = 2$ (DTM_{M1} / DTG). Le monde M_1 s'exécute 1 fois (en vert) tous les *2 TG*. C'est-à-dire quand la valeur du *compteur de TG* est un multiple de *2*.

Le calendrier de M_1 s'incrémente par pas de *12 heures* (échelle temporelle de M_1) à chaque fois que M_1 s'exécute. En d'autres termes les champs d'incrémentation du calendrier (*tickAmount_Ucalendar*, *tick_UcalendarField* et *tickAmountMultiplicator*) gardent leur valeur par défaut (voir II.A.3, p. 48). Ainsi M_1 finit son exécution avec son calendrier présentant la même date que celui de l'ordonnanceur global.

Nous proposons dans l'annexe B (Partie II de l'Algorithme 5) un algorithme permettant de mettre en œuvre cet ordonnancement.

I.D.1.b Situation où DTM_{Mi} est supérieure à DTG mais n'étant pas son multiple

Nous prenons l'exemple de M_2 :

$DTM_{M2} = 8\ h$;
$N1 = DTM_{M2} / DTG = 1,3$.

Le monde M_2 doit donc s'exécuter *1 fois* tous les *1,3 TG*. En d'autres termes, puisque le *tick* est discret, il doit s'exécuter *10 fois* tous les *13 TG* (pour être encore plus précis on peut aussi dire *100 fois* tous les *133 TG* etc.). Nous considérons alors, que le monde M_2 s'exécute *1 fois* tous les *2 TG* (arrondi par excès de DTM_{M2} / DTG) au lieu de *1,3 TG* (DTM_{M2} / DTG). Par contre, lorsque M_2 parvient à un certain nombre d'exécutions que nous notons *B*, il n'a plus la main tous les *2 TG*, mais tous les *TG* afin de parvenir à s'exécuter *10 fois* à la fin des *13 TG* (voir Figure 34).

Détermination de *B*

Soient :

N1 = arrondi par excès de DTM_{M2} / DTG, (*N1* = *2* avec l'exemple) ;

A = arrondi par défaut de $DTM_{M2} \times 10 / DTG$, (*A* = *13* avec l'exemple).

Alors :

$B = (A - 10) / (N1 - 1)$, (*B* = 3 avec l'exemple).

Preuve

Pour une meilleure compréhension on peut se baser sur la Figure 34 ci-dessous.

Si à partir d'un certain nombre de *TG*, 1 par exemple, le monde M_2 s'est déjà exécuté *B fois* et qu'il faire au total *10* exécutions quand on parvient à *A TG*, alors il lui reste *10 - B* exécutions à raison d'une exécution par *TG*. Au même moment, il reste à l'ordonnanceur global lui aussi *A - (B×N1) TG* pour arriver à *A TG*. Ainsi nous avons :

$$10 - B = A - (B \times N1) \Rightarrow B = (A - 10) / (N1 - 1)$$

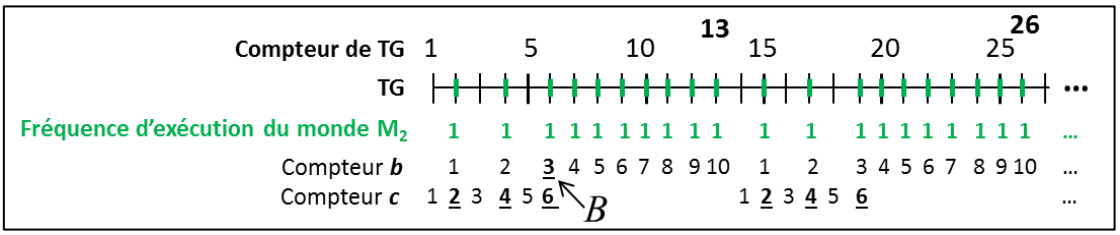

Figure 34 – Situation où DTM_{Mi} est supérieure à DTG mais n'étant pas son multiple. Prenons par exemple *N1 = 2* (arrondi par excès de DTM_{M2} / DTG). Le monde M_2 s'exécute 1 fois (en vert) tous les *2 TG* (*N1 TG*). Après *B* exécutions, la fréquence change, il s'exécute alors *1 fois* à chaque *TG* pendant *B – 10 TG*.

Pour mettre en place un algorithme (voir Partie III de l'Algorithme 5) permettant de mettre en œuvre cette ordonnancement, nous aurons besoin, en plus de *N1* et de *B*, de deux compteurs *b* et *c* :

b : permet au simulateur de savoir s'il se trouve avant ou après *B* et s'il a atteint *10*. Si *b* est inférieur ou égal à *B* alors M_2 s'exécute *1 fois* tous les *N1 TG* ; et si *b* est supérieur à *B* alors M_2 s'exécute *1 fois* à chaque *TG*. Ce compteur *b* s'incrémente par pas de *1* à chaque fois que M_2 a la main et est réinitialisé à *1* quand il atteint *10* ;

c : permet de connaitre les multiples de *N1* quand *b* est inférieur ou égal à *B* (car, pour ce cas-ci, le compteur de TG ne marche pas pour déterminer les bons multiples de *N1*). Le compteur *c* s'incrémente par pas de *1* à chaque *TG* à condition que *b* soit inférieur à *B*. Il est réinitialisé à *1* quand *b* atteint *10*.

En ce qui concerne l'incrémentation du calendrier de M_2 :

i. si *b* <= *B*, il s'incrémente de *N1* x *DTG* (2 x 6 h = 12 h ici) à chaque fois que M_2 s'exécute ;
ii. si *B* < b <= 10, il s'incrémente de *DTG* (6 h ici) à chaque fois que M_2 s'exécute (*i.e.* à chaque TG).

Les champs *tickAmount_Ucalendar* et *tick_UcalendarField* du calendrier de M_2 seront donc les mêmes que ceux du calendrier de l'ordonnanceur global. Par contre *tickAmountMultiplicator* va prendre la valeur de *N1* pour le cas i) et la valeur 1 pour le cas ii). Ainsi M_2 finit sa main avec son calendrier présentant la même date que celui de l'ordonnanceur global.

I.D.1.c Situation où DTM_{Mi} est un diviseur de DTG

Nous prenons l'exemple de M_3 :

$DTM_{M3} = 3h$;

$N2 = DTG / DTM_{M3} = 2$.

Le monde M_3 doit donc s'exécuter *2 fois* (*N2 fois*) à chaque *TG*. Nous pouvons illustrer ceci avec la Figure 35 suivante :

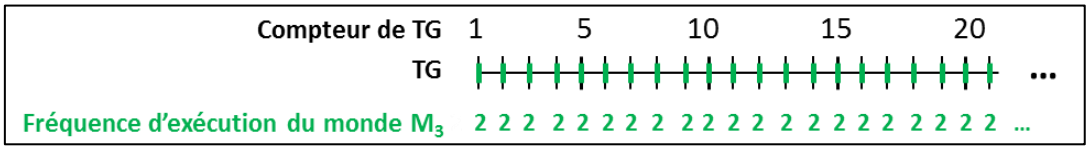

Figure 35 – Situation où DTM_{Mi} est un diviseur de DTG.
Prenons par exemple *N2* = 2 (DTG / DTM_{M3}). Le monde M_3 s'exécute *2 fois* (en vert) à chaque *TG*.

Le calendrier de M_3 s'incrémente par pas de *3 heures* (son échelle temporelle) à chaque fois que M_3 s'exécute. En d'autres termes les champs d'incrémentation du calendrier gardent leur valeur par défaut (voir II.A.3, p. 48). Le calendrier de M_3 et celui de l'ordonnanceur global sont aussi synchronisés.

Nous proposons dans l'annexe B (Partie IV de l'Algorithme 5) un algorithme permettant de mettre en œuvre cet ordonnancement.

I.D.1.d Situation où DTM_{Mi} est inférieure ou égal à DTG mais n'étant son diviseur.

Nous prenons l'exemple de M_4 :

$DTM_{M4} = 4\ h$;

$N2 = DTG / DTM_{M4} = 1,5$.

Le monde M_4 doit donc s'exécuter *1,5 fois* à chaque *TG*. En d'autres termes, puisque le *tick* est discret, il doit s'exécuter *15 fois* tous les *10 TG*. Nous allons donc dire que le monde M_4 s'exécute, non pas *1,5 fois* (DTG / DTM_{M4}), mais *2 fois* (arrondi par excès de DTG / DTM_{M4}) à chaque *TG*. Par contre, lorsque M_4 parvient à un certain nombre d'exécutions que nous notons encore *B*, il n'a plus la main tous les *2 TG*, mais tous les *TG* afin de parvenir à s'exécuter *15 fois* au bout des *10 TG*.

Détermination de *B*

Soient :

N2 = arrondi par excès de DTG / DTM_{M4}, (*N2 = 2* avec l'exemple) ;

A = arrondi par défaut de $DTM_{M2} \times 10 / DTG$, (*A = 15* avec l'exemple).

Alors :

B = (A - 10) / (N2 - 1), (*B* = 5 avec l'exemple).

Preuve

Pour une meilleure compréhension on peut se baser sur la Figure 20 ci-dessous) :

Si à partir d'un certain nombre de TG, 1 par exemple, le monde M_4 s'exécute $N2 \times B$ *fois* et qu'il doit s'exécuter *A fois* en *10 TG*, alors il lui reste $A - N2 \times B$ exécutions à raison d'une par *tick*. Au même moment, il reste à l'ordonnanceur global *10 - B TG* pour arriver à *10 TG*. Ainsi nous avons :

$$A - (B \times N2) = 10 - B \Rightarrow B = (A - 10) / (N2 - 1)$$

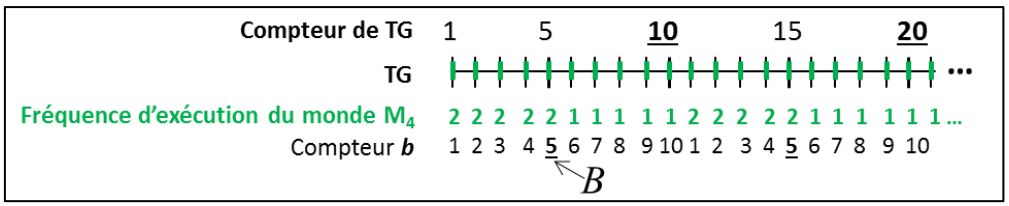

Figure 36 – Situation où DTM_{Mi} est inférieure ou égal à DTG mais n'étant son diviseur. On prend par exemple le cas où *N2 = 2* (arrondi par excès de DTG / DTM_{M4}). Le monde M_4 s'exécute à chaque TG : *2 fois* (*N2* fois) quand *b* est inférieur ou égal à *B* et *1 fois* quand *b* est supérieur à *B*. Ce qui lui permet d'avoir *A exécutions* (*15 exécutions*) tous les *10 TG*.

Pour mettre en place un algorithme (voir Partie V et Partie VI de l'Algorithme 5 dans l'annexe B) permettant de mettre en œuvre cet ordonnancement, nous aurons besoin ici aussi, en plus de *N2* et de *B*, d'un compteur *b* :

b : permet au simulateur de savoir s'il se trouve avant ou après *B* et s'il a atteint *10 TG*. Si *b* est inférieur à *B* alors le monde M_4 s'exécute *2 fois* à chaque *TG*; et si *b* est supérieur à *B* alors il s'exécute *1 fois* à chaque *TG*. Ce compteur *b* s'incrémente par pas de *1* à chaque exécution de M_4 et recommence à *1* quand il atteint *10*.

En ce qui concerne l'incrémentation du calendrier de ce monde, il s'incrémente de *DTG* (12 h ici) à chaque TG (non pas à chaque exécution de M_4, voir Partie VI de l'Algorithme 5

dans l'annexe B). Ainsi le calendrier de M_4 est aussi synchronisé à celui de l'ordonnanceur global.

En ce qui concerne les deux dernières situations c) et d), où un monde peut s'exécuter plusieurs fois lors d'un TG, l'ordonnanceur global (voir I.D.2) ne lui donne pas, de manière consécutive, toutes ses mains pour ce TG. Mais plutôt il recence tous les mondes de ce genre, leur donne tous la main une fois, ensuite tous la main une deuxième fois ainsi de suite pour qu'ils s'exécutent ; si un monde épuise le nombre de fois pour lesquelles il doit s'exécuter, alors il cesse d'être activé ; jusqu'à ce que tous les mondes pour ce TG, s'exécutent un nombre normal de fois. Le même type d'ordonnancement se passe au niveau de chaque monde si des situations similaires à celles de c) et d) s'y présentent (voir II.A.3, p. 48). Cette approche permet de bien mélanger la façon de donner la main aux mondes et l'exécution des unités d'activité des agents dans les mondes dans le cas où la *DTM* de certains mondes est plus petite que la *DTG* de l'ordonnanceur global. Ceci est pris en compte dans la dernière partie de l'Algorithme 5 qui concerne la fonction *ordonnancerMainsMondes* (voir Partie VI de l'Algorithme 5 dans l'annexe B).

I.D.2 Retour sur la classe GlobalScheduler

Pour ordonnancer la manière de donner la main aux mondes suivant les quatre situations présentées précédemment, nous avons mis en place la classe *GlobalScheduler* (voir Figure 31) disposant de :

- quatre tableaux associatifs permettant d'associer les mondes (M_i) aux informations (*N1*, *N2*, *A*, *B*, *b* et/ou *c*) étudiées précédemment. Ces tableaux associatifs contiennent donc des mondes (objet de la classe *World*) associés à des tableaux d'entiers (*tableauInfos*) dont le contenu dépend uniquement des quatre situations précédemment présentées :
  a) *multiple_DTG_Mondes* : quand *DTM* est un multiple de *DTG*. *tableauInfos* ne contient que *N1* ;
  b) *non_multiple_DTG_Mondes* : quand *DTM* est supérieure à *DTG* mais n'étant pas son multiple. *tableauInfos* contient *N1*, *B* et les compteurs *b* et *c* ;
  c) *diviseur_DTG_Mondes* : quand *DTM* est un diviseur *DTG*. *tableauInfos* ne contient que *N2* ;
  d) *non_diviseur_DTG_Mondes* : quand *DTM* est inférieure à *DTG* mais n'étant pas son diviseur. *tableauInfos* contient *N2*, *B* et un compteur *b* ;
- un champ *DTG* qui contient la durée du tick global ;
- une fonction *ajoutMonde(monde)* permettant de renseigner les tableaux associatifs cités ci-dessous. Cette fonction prend en paramètre un monde (une référence de *monde*).
- une fonction *ordonnancerMainsMondes()* qui permet, durant la simulation, de donner la main aux mondes avec les bonnes fréquences ;
- une fonction *changementDeLaDTG()* qui permet de reconstruire les tableaux associatifs en recalculant les éléments des tableaux *tableauInfos* après un changement de l'échelle temporelle globale en cours de simulation (voir section II.A.3.d) ;

- une fonction *priseEnCompteMondeChangeantDTM()* qui permet, après un changement d'échelle temporelle d'un monde en cours de simulation, de recalculer les informations de son *tableauInfos* et de le replacer dans les tableaux associatifs (voir section II.A.3.d, p. 56) ;
- une fonction *ajoutMondeChangeantDTM()* permettant d'ajouter dans une liste temporaire (*mondesChangeantDTM* ci-dessous) les mondes ayant changé d'échelle temporelle avant que l'ordonnanceur global ne prenne en compte ces changements avec la fonction *priseEnCompteMondeChangeantDTM ()*. En effet cette dernière est exécutée dans la fonction *ordonnancerMainsMondes()* qui boucle sur des listes qu'elle ne peut pas modifier en même temps (voir Partie I de l'Algorithme 5 dans l'annexe B) ;
- un champs *mondesChangeantDTM* ;
- une fonction *lectureDesParametres()* qui permet de lire les paramètres temporels. Ces derniers ne sont pris en compte qu'en début de simulation et quand ils sont modifiés (voir section II.A.3.d, p. 56) en cours de simulation.

I.D.3 Mis en œuvre de l'ordonnancement global

Pendant le développement du modèle, il suffit donc de construire les mondes, renseigner leur attribut *DTM* et renseigner les tableaux associatifs de l'ordonnanceur global en utilisant la fonction *ajoutMonde(monde)*. Cette fonction sera donc appelée autant de fois qu'il y aura de mondes et à chaque fois qu'une *DTM* ou la *DTG* change en cours de simulation (voir la section II.A.3.d, p. 56).

Après cela, c'est la fonction *ordonnancerMainsMondes()* qui est exécutée à chaque tick global par le scheduler interne de Repast Simphony (avec une annotation @Scheduler). Cette fonction (*ordonnancerMainsMondes*) ordonnance à son tour et pendant toute la simulation la manière de donner la main aux mondes afin qu'ils se gèrent.

Nous avons présenté dans l'annexe B les algorithmes (Algorithme 4 et Algorithme 5) des fonctions *ajoutMonde()* et *ordonnancerMainsMondes()*.

I.D.4 Changer une échelle temporelle en cours de simulation

I.D.4.a Changer l'échelle temporelle des mondes

Comme nous l'avons présenté avec le modèle mono-échelle, il est possible de changer l'échelle temporelle d'un monde en cours de simulation (voir II.A.3.d, p. 56). Ce changement est pris en compte dans le meta-modèle multi-échelle grace aux fonctions *ajoutMondeChangeantDTM()* et *priseEnCompteMondeChangeantDTM()* dont nous présentons les algorithmes (Algorithme 6 et Algorithme 7) à l'annexe B.

La fonction *ajoutMondeChangeantDTM()* est appelée par les mondes quand il est nécessaire de changer d'échelle temporelle en cours de simulation. Elle permet de mettre temporairement les mondes qui doivent changer de *DTM* (rappel DTM : durée du tick monde) dans la liste *mondesChangeantDTM*. En effet il n'est pas possible de changer la *DTM* des mondes au moment où l'ordonnanceur général est en train de leur donner la main pour qu'ils s'exécutent (il n'est pas commode de boucler sur une liste et de la modifier en même temps).

Au cours de son exécution, pendant qu'elle est en dehors des boucles utilisant les listes contenant les mondes, la fonction *ordonnancerMainsMondes()*,teste tout d'abord s'il y a des mondes dans la liste *mondesChangeantDTM*. Si c'est le cas, elle fait appel à la fonction *priseEnCompteMondeChangeantDTM()* afin qu'elle les retire de leur ancienne liste pour les mettre (avec la fonction *ajoutMonde()*) dans celle correspondant à leur nouvelle *DTM*.

I.D.4.b Changer l'échelle temporelle globale

Comme nous l'avons présenté pour les modèles mono-échelles, nous pouvons ici aussi changer l'échelle temporelle globale en cours de simulation. Nous rappelons que les paramètres temporels peuvent être changés, lus et pris en compte en cours de simulation.

Ainsi pour changer l'échelle temporelle globale on change la durée du pas de simulation dans les paramètres de simulation (voir Figure 21). Après cela l'ordonnanceur global, avec la fonction *lectureDesParametres()*, lit ces paramètres temporels, puis modifie les valeurs du champ *DTG* et celles des champs d'incrémentation de son calendrier. Enfin, il réadapte les fréquences auxquelles il donne la main aux mondes. Pour cette réadaptation nous proposons la fonction *changementDeLaDTG()* de l'Algorithme 8 dans l'annexe B qui se déroule en trois étapes : *(i)* réaliser des copies des quatre tableaux associatifs concernant l'ordonnancement global, *(ii)* les vider et *(iii)* à partir des copies, les reconstruire en réutilisant simplement la méthode *ajoutMonde()*.

La fonction *changementDeLaDTG()* est donc appelée par la fonction *lectureDesParametres()* qui est à son tour appelée par la méthode *ordonnancerMainsMondes()* qui est à son tour exécutée à chaque tick global.

I.E Interaction entre mondes

I.E.1 Agent et déplacement d'un monde à un autre

Dans ce modèle multi-échelle comme dans le modèle mono-échelle, les agents ne perçoivent que leur environnement proche. Cependant, dans certains cas, des agents peuvent avoir à se déplacer dans d'autres mondes, à des endroits inscrits dans leur liste de destinations possibles. Pour cela ils doivent eux-mêmes, après avoir choisi une destination dans un autre monde, se construire un chemin complet pour s'y rendre. Un chemin contenant notamment les points de passage où il doit/peut changer de monde : les *portes* (objet de la classe *Gate*, voir Figure 31 et Figure 37).

Pour montrer comment nous avons procédé, nous prenons l'exemple de la Figure 37 dont le monde 1 contient une zone dans laquelle on veut entrer plus en détail, représentée par le monde 2, lui-même contenant une zone dans laquelle on veut entrer encore plus en détail, représentée par le monde 3. Pour des raisons de simplicité, il faut impérativement passer par un monde intermédiaire (s'il existe) pour aller dans un monde donné (*i.e.*, on ne peut pas aller de M_1 à M_3 sans passer par M_2).

Ainsi, supposons qu'un agent (transporteur) veuille aller du point $M_{1,1}$ du monde 1 au point $M_{3,2}$ du monde 3. Puisque la destination $M_{3,2}$ est dans un monde différent du monde courant de l'agent, alors l'agent passera par les étapes suivantes pour s'y rendre :

- construire ce que nous appelons le *chemin de mondes* (avec la fonction *construireCheminDeMonde()* de l'Algorithme 9 que nous proposons dans l'annexe B), ce qui donne le chemin $\{M_2, M_3\}$. C'est-à-dire que l'agent étant dans M_1, il doit aller à M_2 ensuite à M_3 avant de pouvoir aller au point $M_{3,2}$;
- après cela parcourir successivement le chemin de monde $\{M_2, M_3\}$. Pour chaque monde de ce chemin (par exemple M_2) :
  - l'agent doit chercher la porte (*gate*) la plus adéquate pour y aller (par exemple le *gate* $M_{1,2}$). C'est-à-dire la porte la plus proche *(i)* appartenant aux graphes sur lesquels évolue l'agent, *(ii)* appartenant à sa liste de destinations possibles et *(iii)* dont la cellule *C* d'arrivée dans l'autre monde appartient aussi aux graphes sur lesquels évolue l'agent (voir I.B) ;
  - après avoir trouvé la bonne porte, l'agent construit et parcourt le bon chemin entre sa position courante et la porte (par exemple le chemin entre $M_{1,1}$ et $M_{1,2}$ qui donne $\{M_{1,1} \dots M_{1,2}\}$) ;
  - parvenu à la porte ($M_{1,2}$) l'agent quitte son monde courant et entre dans l'autre (monde 2) par la cellule d'arrivée ($M_{2,1}$) reliée à la porte (voir la classe *Gate*). En changeant de monde, l'agent se déréférence, ainsi que les agents et objets qu'il contient, de l'ancien monde et se référence, ainsi que les agents qu'il contient, au nouveau monde. Ceci grâce au gestionnaire d'environnement et aux inspecteurs qui se chargent de les référencer dans les bonnes listes d'agents afin que l'ordonnanceur du monde puisse leur donner la main avec les bonne fréquences. Les agents dont l'unité d'activité dépend du temps vont recalculer leur vitesse par rapport à la durée du tick de ce nouveau monde. Ce qui représente l'*accommodation* à la nouvelle échelle temporelle ;
  - l'agent doit ensuite s'accommoder au nouvel environnement et à la nouvelle échelle spatiale. Par exemple, il récupère à partir de la cellule d'arrivée le graphe à partir duquel il doit se déplacer ;
  - il réalise le même processus pour se rendre dans le monde suivant (M_3) et ainsi de suite s'il y a d'autres mondes ;
- si le chemin de mondes est épuisé, c'est-à-dire si l'agent arrive au monde (M_3) où se trouve sa destination finale ($M_{3,2}$), il construit et parcourt le bon chemin entre sa position courante ($M_{3,1}$) et cette destination finale ($M_{3,2}$).

Ainsi, dans notre cas d'étude, les rats perçoivent et montent dans les véhicules, qui à leur tour se déplacent dans d'autres mondes et donc y amènent les rats transportés. Ainsi les rats se diffusent d'un monde à l'autre.

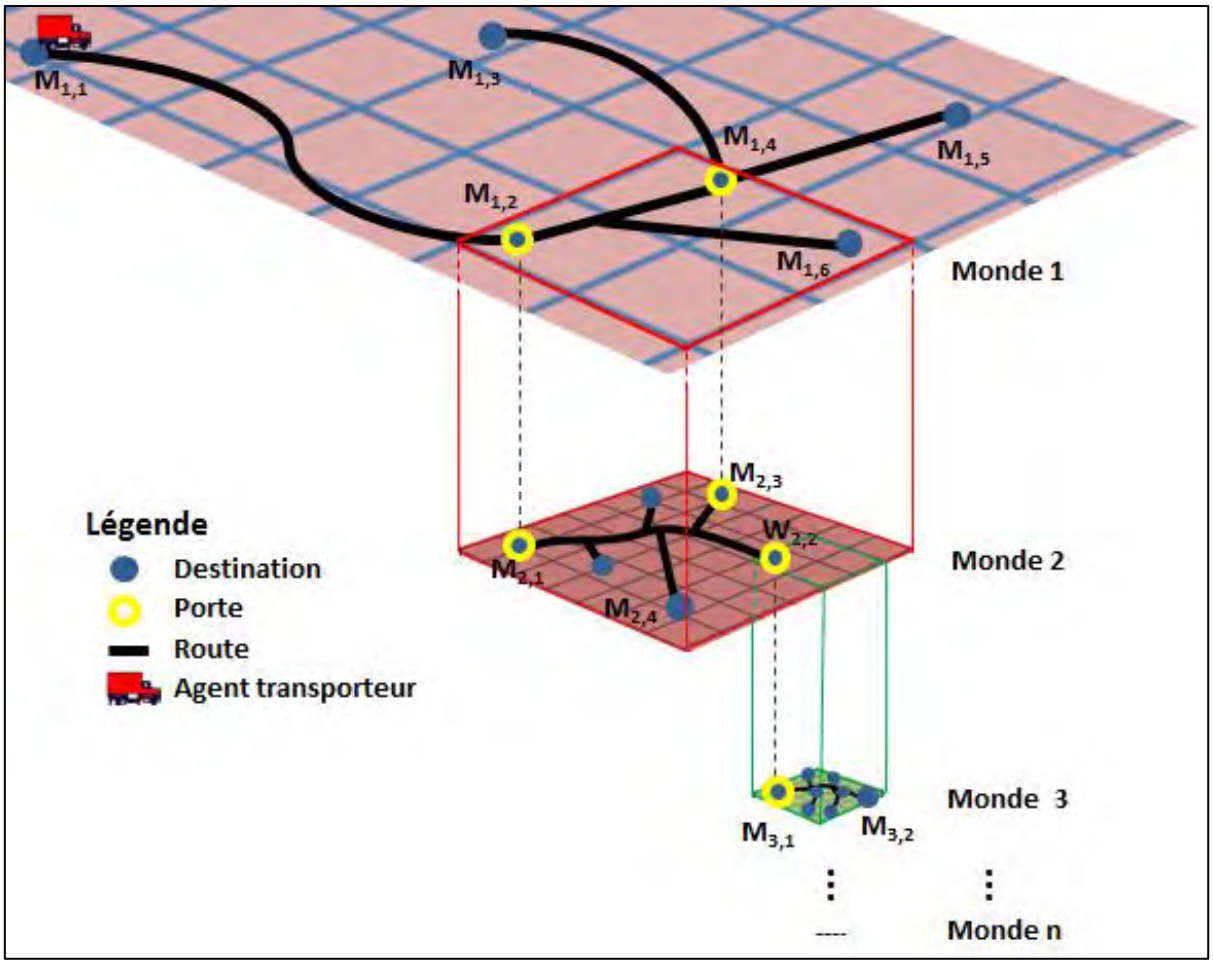

Figure 37 – Notion de points (portes) de passage entre mondes.
Une porte de sortie permettant de passer de "monde i" à "monde j" (porte se trouvant donc dans "monde i") est associée à une cellule (d'arrivée, voir I.B, p. 75) appartenant à "monde j" qui peut elle aussi être une porte de sortie pour passer de "monde j" à "monde i". Nous rappelons que les mondes se trouvent sur un même plan (voir Figure 28).

I.E.2 Perception inter-mondes

La partie encadrée en rouge du monde 1 (sur la Figure 38) est représentée plus en détail et avec une échelle spatiale plus fine au niveau du monde 2. Ces deux mondes ont le même espace continu mais leur grille et leurs graphes sont différents. Ainsi si un agent utilise le voisinage de Moore ou de von Neumann pour percevoir son environnement à partir de la grille, il ne pourra pas percevoir certains agents, même très proches de lui (par rapport à l'espace continu) mais se trouvant dans un autre monde. Pour régler ce problème, nous choisissons les zones les plus importantes (zones d'intérêt) et communes aux mondes comme étant exactement les mêmes zones (les même objets ou références en POO). Par exemple sur la Figure 38 les zones en jaune du monde 1 et du monde 2 représentent exactement la même ville. Ainsi les agents se trouvant dans cette ville peuvent se percevoir les uns les autres même s'ils sont dans des mondes différents. Cependant la condition est que chaque monde soit bien positionné, sur l'espace continu, par rapport à la partie qu'il représente dans un autre monde

(voir la section I.C, p. 76 et la Figure 38) ; ce qui est utile pour la troisième étape de la perception des agents percevant leur environnement proche (ci-dessous).

Ainsi, après avoir créé un monde (monde 1 par exemple) avec une instance d'une ville du nom de Tambacounda par exemple, si on crée plus tard un autre monde (monde 2 par exemple) devant avoir une ville du même nom que Tambacounda, alors on n'instancie pas une nouvelle ville avec le même nom, mais on utilise la même instance de ville pour les deux mondes. On procède de la même façon pour un troisième monde qui contient une ville du même nom (voir Figure 38).

L'espace continu est identique mais les grilles ne le sont pas d'un monde à un autre. Ainsi nous avons mis en place une classe *C_City* qui étend *LandPlot* (block de cellules contiguës) et qui a une liste de cellules par monde (voir Figure 38 ci-dessous). De ce fait, la perception des agents percevant leur environnement proche sera en 3 étapes :

1. à partir de son rayon de perception, calculer l'ordre n du voisinage de Moore. A partir de n, recenser les cellules voisines et les objets et agents qu'elles contiennent (recensement dans le monde courant) ;
2. récupérer toutes les autres cellules (et leur contenu) de la ville appartenant à d'autres mondes (recensement dans les autres mondes, les agents connaissent aussi leur ville courante) ;
3. enfin, parmi les éléments recensés, enlever tous ceux dont la distance avec l'agent réalisant la "perception" est supérieure au rayon de perception.

Avec cette méthode un agent (rat par exemple) pourra percevoir et monter dans un véhicule évoluant dans un monde différent de celui du rat. Alors si le véhicule reste dans le même monde, il ne va pas faire de référencement/déréférencement pour lui mais le fera pour les agents qui proviennent d'un autre monde quand il les "décharge".

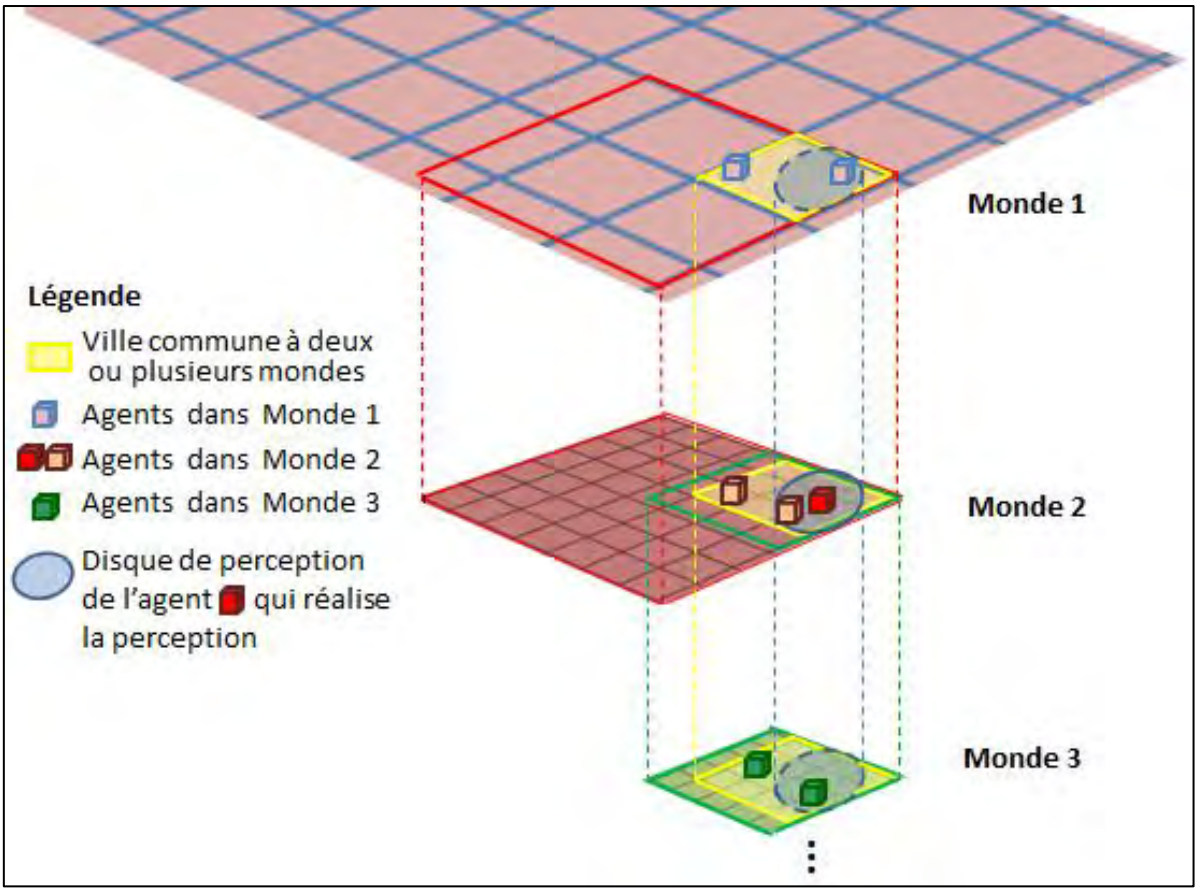

Figure 38 – Perception d'un agent se trouvant dans une ville commune à plusieurs mondes. La ville (en jaune) existe dans les trois mondes. Elle occupe le même espace continu et représente le même objet (même référence en POO). Cependant elle couvre une liste de cellules différente pour chaque monde (1 dans monde 1, 9 dans monde 2 et 16 dans monde 3). Ainsi un agent (agent en couleur rouge par exemple) peut percevoir en utilisant le voisinage de Moore ou de von Neumann, les agents/objets se trouvant dans son disque de perception même s'ils se trouvent dans d'autres mondes. Nous rappelons que les mondes se trouvent sur un même plan (voir Figure 28, p. 72).

II. Implémentation et vérification

Comme avec le modèle de monde nous avons aussi implémenté et testé, dans le projet de plate-forme SimMasto ce meta-modèle qui permet une représentation multi-monde multi-échelle permettant la simulation de modèles en cascade et/ou emboités. Les mondes interagissent et les agents qui s'y trouvent se déplacent d'un monde à l'autre et s'adaptent aux mondes courants.

Nous avons essayé ainsi d'appliquer ce meta-modèle à notre étude de cas en emboîtant en cascade les trois mondes que nous avons présentés à la section I, p. 42.

II.A Gestion d'échelles spatiale multiples

En ce qui concerne la multi-échelle spatiale, nous pouvons instancier plusieurs mondes d'échelle spatiale différentes et les positionner là où l'on souhaite sur l'espace continu, il suffit de changer les coordonnées de l'origine O_i du repère du monde *i* (comme nous l'avons étudié à la section I.C de cette partie). Ainsi nous pouvons créer des mondes, chacun sur un "display" (déplaçable sur l'écran de l'ordinateur), tout en respectant leur position exacte (sur l'espace continu) par rapport à la zone du monde qu'il représente (voir Figure 39 et Figure 40). Ce qui permet de mieux gérer l'emboîtement de modèles en particulier la perception d'agents/d'objets appartenant à un autre monde (voir la section I.E.1 de cette partie).

Nous avons ainsi avec notre cas d'étude créé les trois *mondes* :

1) nous avons pris comme origine (O_0) du repère absolu le même que celui du *monde 1* (O_1) et comme norme du vecteur unitaire du repère absolu l'échelle spatiale du *monde 1* (7,5 km, voir la phase 1, p. 42). Ainsi O_0 et O_1 ont pour coordonnées (0 ; 0) (voir Figure 39) ;
2) le *monde 2* qui représente les régions de Tambacounda et de Kédougou du *monde 1* (la partie encadrée en rouge du *monde 1* sur la Figure 39) a pour origine O_2 de coordonnées (315 km ; 0 km) sur le repère absolu. Sur l'axe vertical passant par O_2 nous avons la ville de Koumpentoum qui représente une *porte* (*gate*) de passage entre *monde 1* et *monde 2* (voir II.C ci-dessous) ;
3) le *monde 3* représente la région de Kédougou du *monde 2* (la partie encadrée en bleu du *monde 2* sur la Figure 39). Il a pour origine O_3 de coordonnées (473 km ; 0 km) sur le repère absolu. Sur l'axe vertical passant par O_3 nous avons la ville de Dienou Diala qui représente un *porte* de passage entre *monde 2* et *monde 3* (voir II.C ci-dessous).

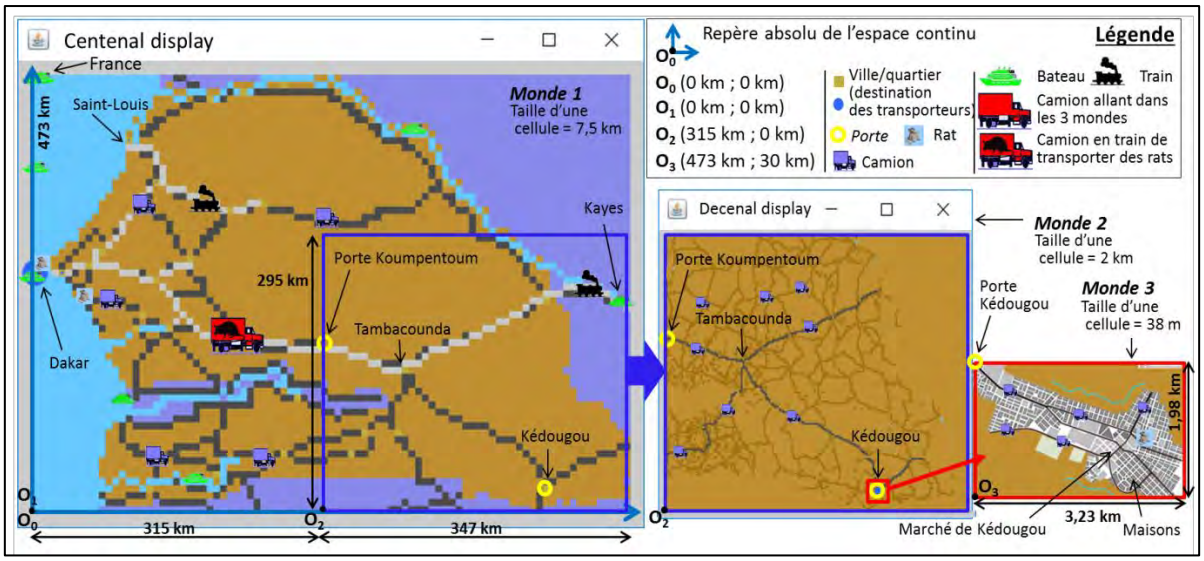

Figure 39 – Trois mondes emboités.
Voir le paragraphe suivant.

Sur la Figure 39, les cercles jaunes représentent les portes de passage entre mondes. Les camions rouges s'en servent pour se déplacer d'un monde à l'autre alors que les camions

bleus, les bateaux et les trains réalisent leur trafic en restant dans leur monde. Ces derniers peuvent ainsi traverser les zones représentées par d'autres mondes sans y entrer. Par exemple le train qui fait le trajet Dakar-Kayes n'entre pas dans le monde 2. Les trois mondes sont chacun dans un display et peuvent être zoomés/dé-zoomés avec la roulette de la souris (voir le monde 3 sur la Figure 41), cependant ils sont spatialement superposés et situés aux bons endroits sur l'espace continu comme représenté sur la Figure 40. Nous avons aussi représenté la France pour prendre en compte les bateaux coloniaux qui transportaient les rats aux comptoirs coloniaux du Sénégal[14].

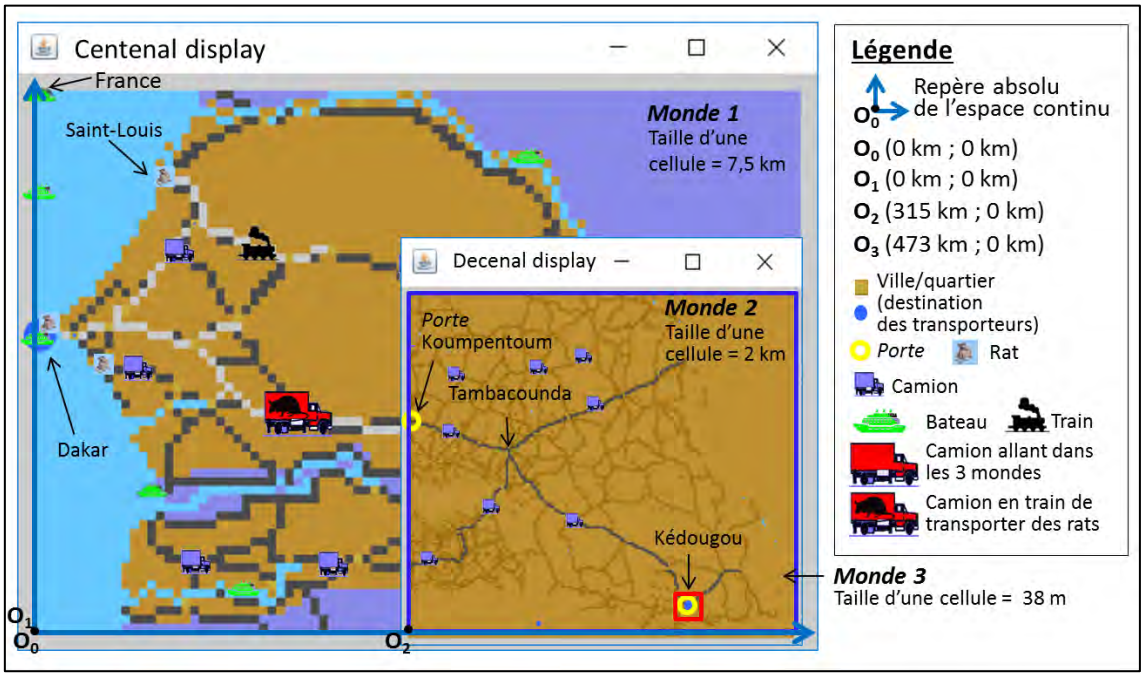

Figure 40 – Mondes emboités superposés.

Comme nous l'avons expliqué dans le paragraphe précédent, les displays peuvent dans certains cas (Centenal et Decenal) être superposés de telle sorte que chaque monde couvre exactement la zone qu'il représente dans un autre monde (Figure 40). Les portes sont ainsi superposées, ce qui peut permettre une meilleure observation du passage des agents transporteurs d'un monde à un autre.

[14] On peut noter que sur la Figure 39 la distance entre la France et ses comptoirs (Dakar, Saint-Louis) n'est pas exacte. Pour remédier à ce problème, il suffit de varier la qualité des cellules qui composent les graphes dans ces trajets, de telle sorte que les bateaux prennent le temps nécessaire pour atteindre ces comptoirs. En effet la vitesse des transporteurs change en fonction de la qualité des cellules qu'ils parcourent.

Figure 41 – Zoom sur le display du monde 3 (voir Figure 26).
Nous pouvons zoomer/dé-zoomer sur un display, il suffit d'utiliser la roulette de la souris (avec la version 2.2 de Repast Simphony).

II.B Gestion d'échelles temporelles multiples

En ce qui concerne la multi-échelle temporelle, nous avons implémenté et testé la classe *GlobalScheduler* et les fonctions lui permettant d'ordonnancer la manière de donner la main aux mondes avec les bonnes fréquences. Les agents qui changent de monde s'adaptent à l'échelle temporelle du monde accueillant. Par exemple les transporteurs adaptent leur vitesse par rapport à la durée du tick du monde courant et le nombre de cellules qu'ils parcourent à chaque unité d'activité compte tenu de la taille des cellules de leur environnement courant. Puisque ces agents sont référencés dans les bonnes listes lorsqu'ils arrivent au sein d'un autre monde (cf. I.E.1), l'ordonnanceur du monde courant leur donne alors la main avec les bonnes fréquences exactement comme nous l'avons décrit à la section I.E.1, p. 86 de cette partie.

Nous avons aussi implémenté et testé le fait qu'un observateur puisse changer l'échelle temporelle d'un monde et/ou l'échelle temporelle globale en cours de simulation comme nous l'avons présenté avec les algorithmes Algorithme 6, Algorithme 7 et Algorithme 8 (voir annexe B).

II.C Interaction entre mondes

II.C.1 Construction des portes et liste de destination inter-mondes

Comme nous l'avons expliqué, nous nous servons du chronogramme pour créer et positionner les portes (*gates*) dans les mondes comme indiqué sur la Figure 39 ci-dessus. Ainsi, les portions de chronogramme suivantes contiennent les instructions qui concernent les portes sur l'exemple de la Figure 39.

	Quand	Où		Quoi	Comment/combien/Spécificité…	
N°	**Date**	**X**	**Y**	**Evénement**	**VAL1**	**VAL2**
1	01/01/1910	…	…	…	…	…
…	…	…	…	…	…	…
n	05/01/1980	43	23	gate	monde2:0,94:Koumpentoum	road
…	…	…	…	…	…	…

Tableau 8 – Portion de chronogramme du monde 1.

Pour le Tableau 8, la ligne n représente l'événement de création d'une porte du nom de Koumpentoum sur la cellule (43, 23) du monde 1, permettant de passer au monde 2 par la cellule d'arrivée (*C*) de coordonnées (0, 94). La porte de coordonnées (43, 23) dans monde 1 est donc une porte permettant le déplacement des agents transporteurs utilisant des graphes de type route (road), du monde 1 au monde 2. Il faut aussi noter que cet événement "gate" se passe le 05/01/1980, c'est-à-dire après avoir entamé la deuxième phase (décennal) correspondant au monde 2.

N°	**Date**	**X**	**Y**	**Evénement**	**Valeur 1**	**Valeur 2**
1	01/01/1980	…	…	…	…	…
…	…	…	…	…	…	…
n	05/01/1980	0	94	gate	monde1:43,23: Koumpentoum	road
n+1	06/01/1980	47	83	truck	5	multiMonde:city::monde1:
n+2	05/01/2000	79	51	gate	monde3:0,51: Dienou Diala	road
n+3	06/01/2000	47	83	truck	2	multiMonde:city:Koumpentoum, Dienou Diala:monde1:Dakar: monde3:Fongolimbi
…	…	…	…	…	…	…

Tableau 9 – Portion de chronogramme du monde 2.

Pour le Tableau 9, la ligne n représente l'événement de création d'une porte du même nom de Koumpentoum sur la cellule (0, 94) du monde 2, permettant de passer au monde 1 par la cellule d'arrivée (*C*) de coordonnées (43, 23). La porte de coordonnées (0, 94) dans monde 1 est donc une porte permettant le déplacement des agents transporteurs utilisant des graphes de type route (road), du monde 2 au monde 1. On peut à présent créer des agents transporteurs dont la liste de destinations possibles peut contenir des destinations appartenant au monde 1. C'est le cas de l'événement de la ligne n+1 qui consiste à créer 5 agents transporteurs (truck qui parcourent des graphes de type route) dont la liste de destinations possibles est indiquée à la colonne *valeur 2* dont la syntaxe, en ce qui concerne la création d'un agent transporteur, est la suivante :

multiMonde:typeDest:dest1,…,destN:monde1:dest1,…,destN:monde3:dest1,…,destN

Si nous découpons cette chaine de caractères au niveau des deux points (:) alors on aura le tableau suivant :

1	2	3	4	5	6	7
multiMonde	*typeDest*	*dest1,…,destN*	*mondeI*	*dest1,…,destN*	*mondeJ*	*dest1,…,destN*

Tableau 10 – Découpage du modèle de chaine de caractères correspondant à l'instruction de création d'un agent transporteur.

Nous avons numéroté les éléments de ce Tableau 10 pour que les explications soient plus simples :

- 1 : *multiMonde*, indique que ces transporteurs auront des destinations qui appartiennent à plusieurs mondes ;
- 2 : *typeDest*, désigne le type de ces destinations (ce sont des villes dans notre cas d'étude) ;
- 3 : *dest1,…,destN*, désigne la liste de destinations dans le monde à qui appartient le chronogramme (monde 2 pour ce chronogramme du Tableau 9) ;
- 4 : *mondeI*, indique un monde (*e.g.* monde 1) dont la liste de destinations qui suit (partie 5 suivante) sera parmi celles des transporteurs ;
- 5 : *dest1,…,destN*, liste de destinations pour le monde précédent (*mondeI*) ;
- 6 : *mondeJ*, indique un autre monde (*e.g.* monde 3) dont la liste de destinations qui suit (partie 7 suivante) sera parmi celles des transporteurs ;
- 7 : *dest1,…,destN*, liste de destinations pour le monde précédent (*mondeJ*).

Il faut noter qu'à partir de la 3^{ième} partie du tableau (Tableau 10), les numéros pairs désignent les mondes et les numéros impairs désignent la liste de destinations dans ces mondes. Cette dernière peut être vide pour un (des) monde(s). Dans cas les transporteurs dans cet événement vont ajouter dans leur liste de destinations toutes celles reliées à leur graphe dans ce (ces) monde(s).

Ainsi les transporteurs (dans l'événement de la ligne n+1), vont avoir comme liste de destinations toutes les villes, du monde 2 et du monde 1, reliées aux graphes sur lesquels ils se déplacent.

L'événement de la ligne n+2 se passe à la date 05/01/2000, après la création du monde 3 le 01/01/2000. Il consiste à créer une porte de type route permettant aux agents transporteurs utilisant des graphes routes de pouvoir se déplacer du monde 2 au monde 3. Une porte de retour (du monde 3 au monde 2) est aussi créée au niveau du chronogramme du monde 3 (voir Tableau 11). Il est à présent possible de créer des agents transporteurs pouvant se déplacer dans les trois mondes. C'est le cas de l'événement de la ligne n+3 qui consiste à créer 2 agents transporteurs (truck) dont la liste de destinations possibles comprend : Koumpentoum et Dienou Diala du monde 2 (courant), Dakar du monde 1 et Fongolimbi du monde 3. Il faut remarquer que la ville Koumpentoum est commune au monde 1 et au monde 2. Ainsi les agents qui s'y trouvent se perçoivent mutuellement même s'ils se trouvent à des mondes

différents, à condition que la distance qui les sépare (sur l'espace continu) soit inférieure à leur rayon de perception comme nous l'avons étudié à la section I.E.2 de cette partie.

Date	X	Y	Evénement	Valeur 1	Valeur 2
05/01/2000	0	51	gate	monde2:79,51: Dienou Diala	road

Tableau 11 – Portion de chronogramme de monde 3.

Après leur création et la construction de leur liste de destinations, les agents transporteurs peuvent choisir une destination parmi cette liste. Ce choix peut être soit suivant la grille gravitaire (cf. II.A.5.a, p. 60 et II.B.4, p. 65) soit aléatoirement avec un poids égal à l'unité pour chaque destination possible. Le transporteur réalise ensuite le processus décrit dans la section I.E.1, p. 86 pour effectuer son "voyage".

II.C.2 Diffusion des rats d'un monde à un autre via les véhicules

Comme nous l'avons fait avec le modèle mono-échelle, les agents rats commensaux de classe *C_RodentCommensal* perçoivent les véhicules en particulier ceux qui font des déplacements inter-mondes, y montent et se font donc transporter d'un monde à un autre comme nous l'avons décrit à la section I.E.1, p. 86. La Figure 42 ci-dessous montre ainsi, le résultat de 6 simulations (les nombres suivants sont arbitraires), chacune lancée sur 10 000 pas de temps (5,5 mois environ, à raison de 4 h pour la DTG, 12 h pour la DTM du monde 1, 4 h pour la DTM du monde 2 et 2 h pour la DTM du monde 3). Avec ces simulations la vitesse des bateaux est de 10 km/h et celle des camions de 60 km/h. Une population de 120 rats est mise à chaque mois dans la zone représentant la France (voir II.B.5, p. 66). Les rats sont transportés de France au Sénégal (à Dakar particulièrement) par 4 bateaux et de Dakar aux autres mondes par des camions dont le nombre varie de 1 à 25. La probabilité (paramétrable) de monter des rats dans les véhicules quand ils les perçoivent se situe entre 90 à 100%.

On peut ainsi voir la dépendance du nombre de rats transportés d'un monde à un autre au nombre de transporteurs inter-mondes. Ce qui montre que le simulateur multi-monde ainsi que le meta-modèle permet bien la représentation de la problématique de la diffusion multi-niveau et multi-échelle du rat noir.

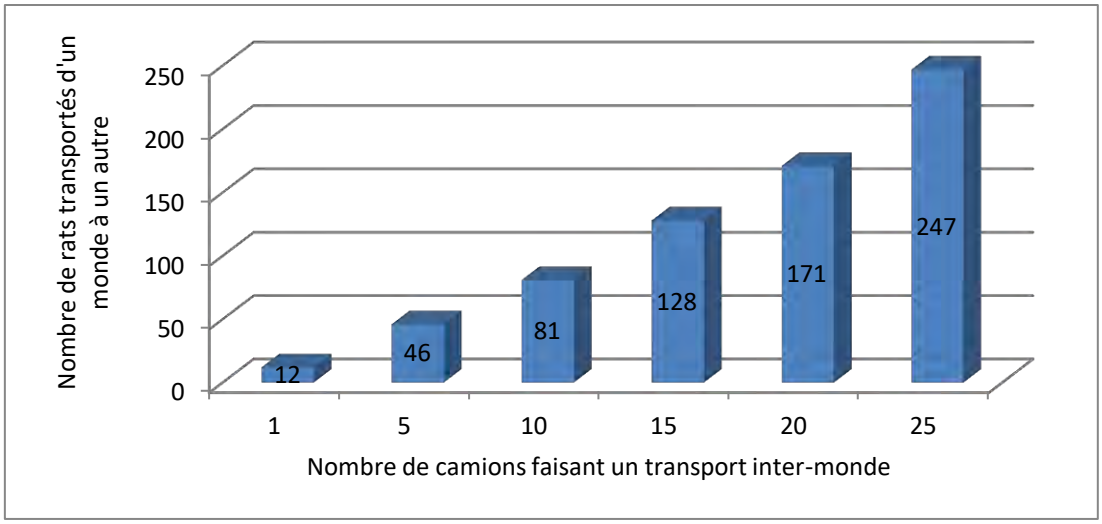

Figure 42 – Résultat de 6 simulations lancées sur 5,5 mois (10 000 TG) montrant l'effet du nombre de transporteurs inter-mondes sur la diffusion multi-monde des agents rats. La probabilité des rats de monter dans un véhicule s'ils le perçoivent se situe entre 90 et 100%.

III. Conclusion

Le meta-modèle proposé permet une représentation multi-monde multi-échelle de phénomènes géographiques dynamiques (inscrits dans une histoire). Il comprend un ordonnanceur global qui ordonnance l'exécution des mondes avec des fréquences calculées à partir de l'échelle temporelle globale et de l'échelle temporelle propre à chaque monde. Chaque monde dispose *(i)* de son propre environnement, avec une échelle spatiale, qui est construit et mis à jour au fil du temps à partir de données appréhendées sous forme d'événements historiques, *(ii)* des portes, greffées à l'environnement et reliées à d'autres portes appartenant à d'autres mondes, qui permettent aux agents d'entrer dans ou de sortir de ce monde et *(iii)* de son propre ordonnanceur qui permet de donner la main aux agents se trouvant dans ce monde, avec des fréquences calculées à partir de l'échelle temporelle du monde et de la durée de l'unité d'activité des agents qui s'y trouvent.

Les agents connaissent ainsi leur monde mais aussi l'existence de lieux appartenant à d'autres mondes. Ils peuvent se déplacer en ces lieux en parcourant des itinéraires complets, comprenant les portes de passage entre mondes, qu'ils construisent au fur et à mesure de leur déplacement. Arrivés à destination, ils déchargent leur contenu, en l'occurrence, les rats qui auraient perçu et seraient montés dans leur véhicule au moment où ils étaient parqués dans ce monde ou dans un autre.

Dans ce meta-modèle les mondes sont autonomes et peuvent fonctionner seuls. Ils peuvent donc être créés n'importe où sur l'espace continu et n'importe quand sur l'étendue temporelle de la simulation globale. Les mondes peuvent aussi être en cascade, emboités et les agents qui s'y trouvent peuvent percevoir leur voisinage dans leur monde courant mais aussi dans les autres et interagir avec les agents et objets perçus (appartenant à leur disque de perception).

Discussion générale

L'approche développée et les algorithmes utilisés ont permis d'atteindre les objectifs liés à notre projet de thèse. C'est-à-dire la mise en place d'un modèle multi-monde multi-échelle avec des emboîtements en cascade portant sur des environnements évolutifs. Un modèle qui permet en particulier la représentation de la problématique inscrite dans le contexte plus global du projet CHANCIRA, qui est la diffusion multi-échelle du rat noir au Sénégal sur le siècle écoulé.

D'autres alternatives auraient pu être considérées pour atteindre ces objectifs. On aurait pu notamment s'intéresser à la plateforme GAMA qui, à l'époque où nous commencions cette thèse, paraissait la plus avancée et la plus proche comme outil de modélisation multi-agents et multi-échelle pour traiter notre problématique (cf. VII.B.3.a, p. 15 et VIII.E, p. 35). Cependant, comme nous l'avons déjà montré (VIII.F, p. 36), elle présentait plusieurs limites par rapport à notre problématique telles que :

- la représentation de l'emboitement de niveaux avec la méthode agent récursif [Gil-Quijano *et al.*, 2012; Lepagnot et Hutzler, 2009; Servat *et al.*, 1998b, 1998a; Tranouez *et al.*, 2006; Vo, 2012; Vo *et al.*, 2012] qui, appliquée à notre problématique, nous amène à avoir des niveaux avec des grilles dissociées dont on a cité les inconvénients à la section VIII.F.1, p. 36. Au lieu de mettre en place des procédures qui peuvent être compliquées pour coupler les grilles dissociées à cause de la méthode agent récursif, nous avons ainsi préféré avoir un modèle (sous-modèle) pour chaque niveau. Ce modèle disposant d'une seule grille avec la granularité (échelle spatiale) que l'on souhaite (VIII.F.1, p. 36) ;
- le passage inter-niveau des agents avec les méthodes *capture/libération* [Vo, 2012; Vo *et al.*, 2012], *émergence*, *création de groupe* [Servat *et al.*, 1998b, 1998a], *agrégation/désagrégation* [Navarro *et al.*, 2011, 2013; Soyez *et al.*, 2012] permettant à des individus d'un niveau de s'agréger ou de se désagréger vers un autre niveau (passage dynamique entre niveaux). Cependant, dans notre modèle c'est l'agent lui-même qui choisit d'aller d'une échelle (monde) à une autre, lui et son éventuel contenu, en suivant des itinéraires et en passant par des portes bien déterminées de passage entre mondes (voir VIII.F.2, p. 38).

L'approche que nous avons utilisée ainsi que la plateforme SimMasto permettent une modélisation et une simulation de phénomènes pluridisciplinaires et multi-échelles plus adéquate vis-à-vis des besoins exprimés.

Pour la modélisation du temps, on aurait pu s'intéresser à la modélisation à événements discrets [Banks *et al.*, 1999, 2005, Müller, 2004, 2007; Steiniger *et al.*, 2012; Uhrmacher *et al.*, 2007], en usant l'approche proposé par Guo et Tay [2008]. Cette dernière permet l'ordonnancement non uniforme de l'exécution des agents avec des durées d'unité d'activité diverses. Cependant nous avons préféré la modélisation à pas de temps discrets qui est utilisé par la plupart des plateformes de simulation multi-agent : NetLogo [Wilensky, 1999b], GAMA [Taillandier *et al.*, 2012], Repast [Collier *et al.*, 2003], Cormas [Bousquet *et al.*,

1998] etc. Nous avons complété ce formalisme en proposant des algorithmes appropriés, permettant non seulement d'exécuter de façon intégrée des mondes ayant des échelles temporelles différentes mais aussi, au sein d'un monde, des agents ayant des unités d'activité de durées différentes.

En ce qui concerne les approches que nous avons utilisées pour la discrétisation de l'environnement, la perception des agents de leur voisinage et la construction des chemins à partir des graphes, il aurait été aussi possible de combiner des grilles avec des "quadtrees" (arbre quaternaire) [Finkel et Bentley, 1974] et les algorithmes de calcul de chemins A* [Hart *et al.*, 1968], ALT [Goldberg et Harrelson, 2005] ou encore Map-Matching [Lou *et al.*, 2009; Quddus et Washington, 2015] pour une optimisation spatiale et temporelle. En effet :

– les "quadtrees" permettent une discrétisation minimaliste de l'environnement. Autrement dit, le niveau de discrétisation n'est pas le même partout sur l'environnement. On ne discrétise d'avantage que là où c'est nécessaire, par exemple là où les agents doivent effectuer beaucoup de calculs pour réaliser la perception de leur environnement (voir Figure 43) ou là où passe une route pour l'élaboration d'un graphe à partir des cellules.

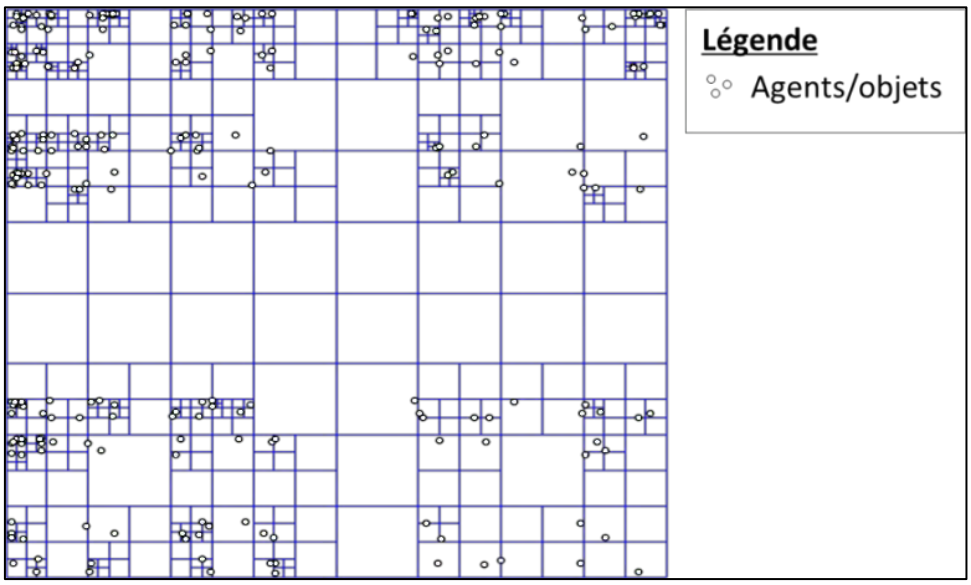

Figure 43 – Discrétisation minimaliste d'un environnement avec un QuadTree.
Source : https://fr.wikipedia.org/wiki/Quadtree, consulté le 27/08/2016.

Cependant nous avons préféré n'utiliser que des grilles, car cette topologie était déjà opérationnelle dans SimMasto. Cette approche était suffisante et représentait un bon choix, car simple à appréhender et fonctionnelle, pour une utilisation efficiente de l'environnement de simulation. Elle a permis une délimitation aisée des zones homogènes (blocs de cellules contigües de même nature) telle que le bassin arachidier (zone de commerce), les routes, les pistes, les grandes villes, les quartiers. Elle permet aux agents de percevoir leur voisinage (voisinage de Moore) avec une rapidité acceptable. Et enfin, elle nous a permis, à partir des zones homogènes de type voie de transport, d'élaborer une topologie de type graphe. Ce dernier a permis aux agents transporteurs de calculer les meilleurs chemins pour se rendre d'un endroit à un autre et d'un monde à un autre.

– En ce qui concerne la manière dont les agents transporteurs calculent leurs chemins, on aurait pu s'intéresser à l'algorithme A* (ou d'autres algorithmes qui l'améliorent tels que ALT [Goldberg et Harrelson, 2005], Map-Matching [Lou *et al.*, 2009; Quddus et Washington, 2015]). A* donne souvent de bons chemins (sinon le meilleur) et il est généralement plus rapide que l'algorithme de Dijkstra. En effet, avec l'algorithme de Dijkstra, il est nécessaire d'explorer tous les nœuds d'un graphe pour trouver à coup sûr le meilleur chemin entre deux nœuds de ce graphe. Ce qui n'est pas le cas avec l'algorithme A* (voir Figure 44).

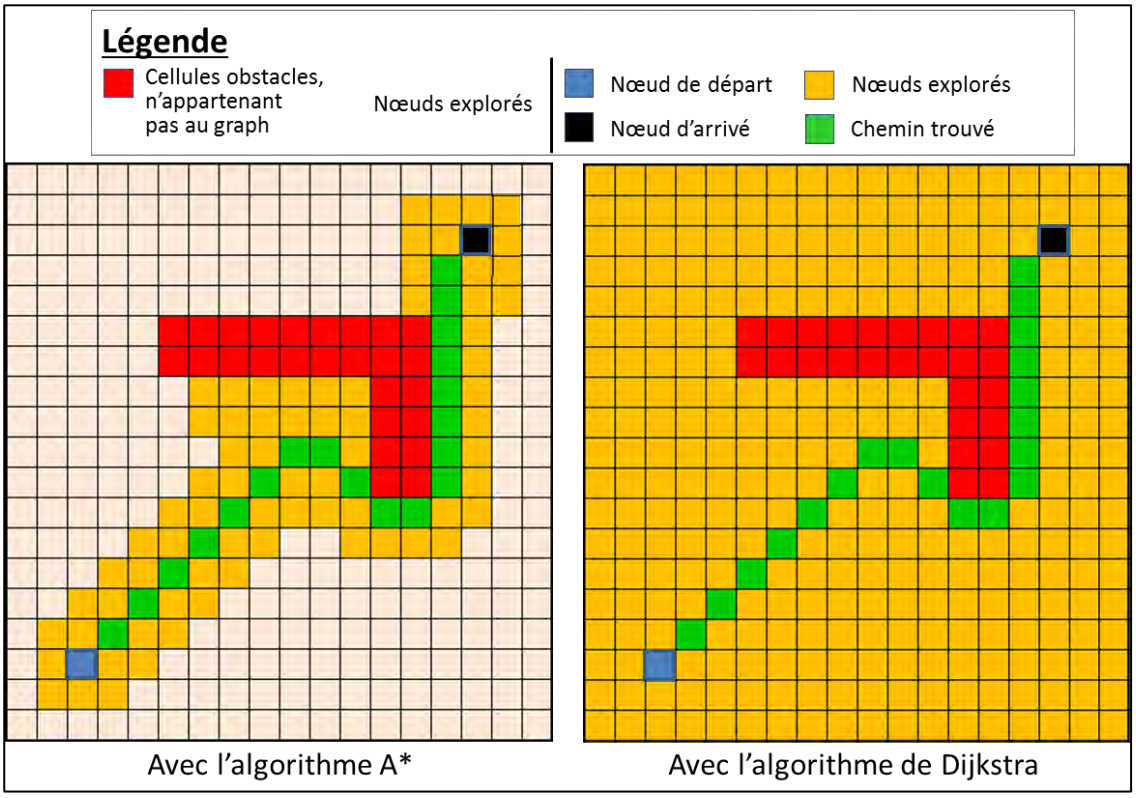

Figure 44 – Nombre de nœuds explorés entre l'algorithme A* et l'algorithme de Dijkstra. Les deux grilles sont identiques. Leurs cellules, sauf les rouges (qui représentent des obstacles), constituent des nœuds de graphe. L'algorithme A* n'a pas besoin d'explorer tous les nœuds du graphe (d'où sa rapidité) pour trouver un bon (voire le meilleur) chemin. Tandis qu'avec l'algorithme de Dijkstra on explore tous les nœuds du graphe et trouve à coup sûr le meilleur chemin.

Cependant nous avons préféré continuer avec l'algorithme de Dijkstra que nous avions déjà implémenté dans la plateforme SimMasto pendant notre Master. En effet cet algorithme est assez rapide et fournit, à coup sûr et dans toutes les situations, le meilleur chemin ; En plus de cela, nous avons apporté une optimisation de son utilisation en vue de remédier à d'éventuelles lourdeurs qui seraient issues de son utilisation sauf en début de simulation ou quand les graphes se mettent à jour (voir annexe C).

En ce qui concerne l'approche que nous avons utilisée pour le déplacement des agents d'un monde à un autre, il aurait été possible de procéder de telle sorte que : si un agent, se trouvant dans un monde (monde 1 par exemple), choisit une destination se trouvant dans un autre monde (monde 2 par exemple) alors il se construit un chemin global complet afin d'être sûr de transiter globalement par le meilleur chemin. Avec cette façon de faire, si par exemple un transporteur qui se trouve à la position $M_{1,1}$ du monde 1 veut se rendre à la position $M_{2,3}$ du monde 2, alors il passe par la porte $M_{1,3}$ (au lieu de la porte $M_{1,2}$). Ce qui lui permet d'emprunter le meilleur chemin (le plus court, voir Figure 45).

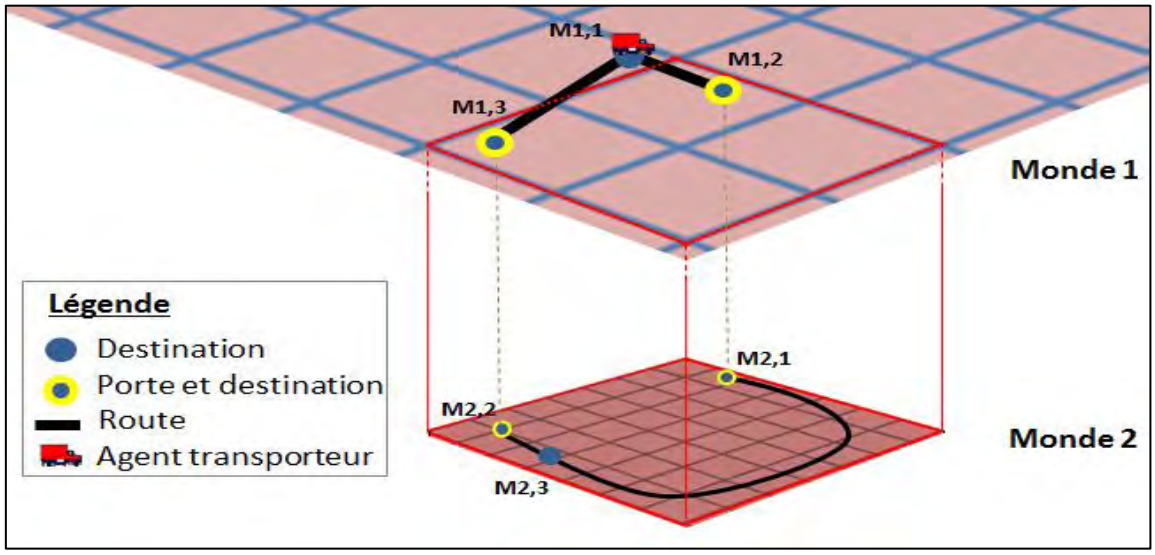

Figure 45 – Passer par la porte la plus proche ne garantit pas de passer par le meilleur chemin. $M_{1,1}$: point de départ ; $M_{2,3}$: point d'arrivée.

Cependant, nous avons préféré que les agents utilisent simplement l'algorithme de Dijkstra pour détecter et passer par la porte la plus proche de leur position pour se rendre dans un autre monde. C'est-à-dire la porte $M_{1,2}$ sur la Figure 45 (avec l'exemple précédent). Ce qui ne lui garantit pas de passer par le meilleur chemin.

Nous avons adopté cette façon de faire car elle est acceptable, plus simple que la première et déjà opérationnelle. En effet les graphes d'un monde à un autre ne sont pas reliés. Donc pour élaborer un meilleur chemin global à partir de ces graphes, il serait nécessaire de mettre en place et/ou d'implémenter des algorithmes de détection de meilleur chemin entre des graphes non connexes (mais présentant des points de jonctions : les portes de liaison entre les mondes).

Dans ce modèle les mondes sont autonomes. Ils peuvent avoir leur propre échelle spatiale et temporelle, des types d'agents qui leur sont propres mais aussi des types d'agents en commun avec les autres mondes. Ces agents sont référencés dans des listes associées avec les informations qui permettent à l'ordonnanceur du monde où ils se trouvent de pouvoir les exécuter avec les bonnes fréquences (voir II.A.3, p. 48). Ainsi si un agent change de monde, il est référencé dans la liste adéquate dans le monde d'arrivée. Cependant nous n'avons pas pris en compte le fait qu'un agent, dont le type n'existe que dans un monde, puisse arriver dans un

autre monde (où ce type d'agent n'existe pas). Par exemple dans le modèle contact (du troisième monde) on pouvait représenter des agents de type moustique, pouvant favoriser la transmission de zoonoses du rat à l'Homme. Ce type d'agent n'existerait donc pas dans les modèles centennal et décennal. Autrement dit, ces derniers ne possèdent pas de listes pour les référencer (avec les informations concernant leur fréquence d'exécutions). Nous n'avons pas jugé nécessaire de prendre cela en compte car il y a des moyens simples pour éviter cette situation. En effet il suffit de faire de telle sorte que si un type d'agent n'existe pas dans un monde x, qu'il ne puisse pas se déplacer par lui-même ou via un autre agent pour se rendre à ce monde x.

Le troisième monde (du modèle contact) que nous avons développé à la section IV, p. 69 ne permet pas d'étudier les modalités de transmission des zoonoses par le rat en fonction de la nature de l'environnement urbain. Il aurait été nécessaire d'identifier et d'intégrer dans le modèle, pour cette phase, les facteurs clés de la transmission de maladie de rat à l'homme : mettre de nouveaux agents tels que des moustiques qui pourraient s'impliquer à la transmission de maladie de rats à l'homme. Cependant les données pour cette phase n'ont pas été disponibles jusqu'à ce que cette thèse arrive à terme. Néanmoins nous avons intégré cette phase afin d'étudier l'approche multi-monde multi-échelle au moins avec trois mondes d'échelles spatiales et temporelles différentes, sans oublier que le meta-modèle et les algorithmes qui lui sont associés sont généralisables pour représenter *n* mondes.

Conclusion – perspectives

Les systèmes complexes en particulier les systèmes complexes sociaux présentent souvent des problématiques pluridisciplinaires, évolutives et multi-échelles spatiales et temporelles : qui résultent *(i)* de déterminants multiples qui opèrent et donc qui doivent être saisis par plusieurs types de *disciplines* et dont les processus sont *(ii)* évolutifs et *(iii)* se déroulent à plusieurs niveaux présentant *(iv)* des emboîtements en cascade et *(v)* des échelles spatiales et temporelles différentes. Nous avons ainsi, après avoir fourni des éclaircissements et des définitions sur les notions que nous avons employées (niveau, échelle, multi-niveau, multi-échelle, multi-monde, emboîtements en cascade), nous avons proposé un meta-modèle permettant de prendre en compte ces cinq différents aspects.

Ce meta-modèle est implémenté dans le projet SimMasto avec la problématique multi-échelle de la diffusion du rat noir au Sénégal sur un environnement évolutif d'un siècle. Cela contribue à améliorer sa généralité et offre en même temps une rétroaction directe pour l'affiner. Ce simulateur (SimMasto) peut à présent être réutilisé pour d'autres études présentant des phénomènes similaires. Il suffira de créer de nouveaux mondes avec de nouveaux chronogrammes et en adaptant l'échelle spatiale et temporelle aux niveaux de détails auxquels on souhaiterait étudier les nouveaux phénomènes.

Les avancées réalisées avec cette thèse permettent ainsi d'apporter une contribution à la modélisation et à la simulation informatique par les systèmes multi-agents, particulièrement à la modélisation multi-échelle présentant des phénomènes biologiques, écologies, historiques et géographiques.

Il apparaît dès à présent que ce meta-modèle peut être amélioré afin de prendre en compte d'autres aspects. Il serait ainsi intéressant de rendre les mondes mobiles ou que la classe Agent du meta-modèle (figure 30), par exemple, puisse étendre la classe World de telle sorte que certains des agents, comme les rats, puissent être représentés comme étant des mondes (des modèles complets). Ce qui permettra d'augmenter la généralité du meta-modèle comme suggéré sur la Figure 46.

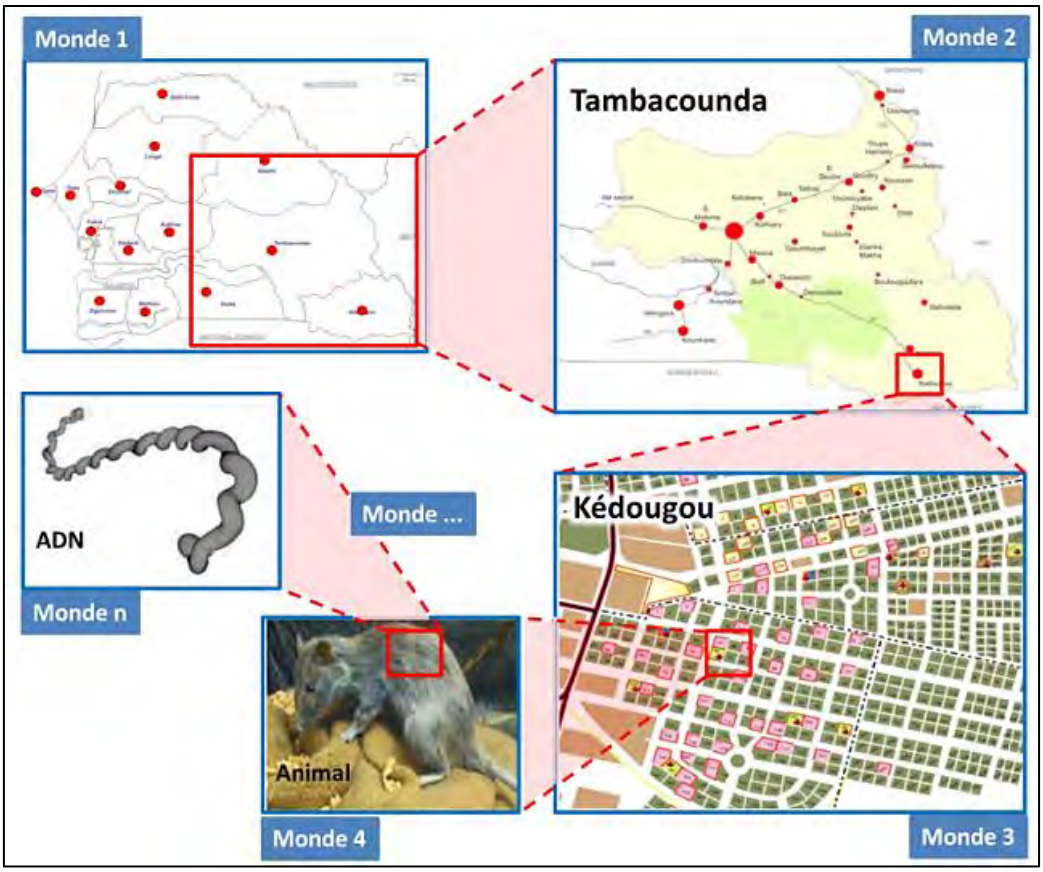

Figure 46 – Multi-monde multi-échelle avec des mondes mobiles.

Il serait aussi intéressant que les transporteurs qui se déplacent d'un monde à un autre, au lieu de construire progressivement des bouts de chemin au cours du déplacement, puissent construire un chemin global complet afin d'être sûrs de passer globalement par le meilleur chemin. Pour cela il serait nécessaire de mettre en place et/ou d'implémenter des algorithmes de détection de meilleur chemin entre des graphes non connexes avec des composants connexes et présentant des points de jonctions : les portes de liaison entre les mondes.

A partir de ce simulateur et avec les données spécifiques récoltées par le projet CHANCIRA le modèle pourra permettre la conduite d'analyse de sensibilité multi-échelle de paramètres clés tels que :

- pour la phase 1, la calibration du processus de transport, essentiellement la probabilité de montée des rats dans les différents types de véhicules, et celle de la dynamique de population de rats à l'échelle nationale-centennale [Le Fur *et al.*, 2016b] ;
- pour la phase 2, dans un deuxième temps et à partir des processus validés lors de la première phase, calibrer l'effet de modifications de l'environnement sur ce processus de diffusion du rat. Le cas d'étude sur cette phase pourrait être l'impact de la construction d'une route bitumée entre Tambacounda et Kédougou à la fin des années 1990. La présence avérée de rat noir à Kédougou depuis ce changement alors que cette ville n'en contenait pas peut constituer le point focal pour la calibration et la validation du processus de diffusion du rat à l'échelle régionale-décennale ;

- enfin, pour la phase 3, à partir des données produites et validées lors de la deuxième phase, à une échelle spatiale et temporelle locale (échelle urbaine-mensuelle/échelle contact), étudier le phénomène de la propagation du rat noir et ses possibilités d'implantation au sein même des systèmes urbains en fonction des paramètres sociaux-environnementaux et de la configuration des espaces (gares routières, marchés, types d'habitation, connectivité, etc.). La ville de Kédougou constitue encore le cas type pour cette phase où l'on pourra s'intéresser aux modalités de transmission des zoonoses par le rat à l'homme en fonction de la nature de l'environnement urbain dans une perspective épidémiologique.

On peut enfin envisager l'adaptation du modèle proposé et de sa gestion multi-échelle à d'autres projets en cours tel que le projet CERISE[15] qui vise notamment à mieux comprendre la colonisation de la souris *Mus musculus* au Sénégal, processus qui se déroule également à plusieurs échelles.

En tout état de cause, ce modèle et les algorithmes qui lui sont associés offrent la possibilité d'appréhender en cascade (ou non) des modèles, informés par des connaissances disciplinaires, représentant les mondes étudiés à plusieurs échelles d'espace et de temps.

[15] Projet CERISE: SCEnarios of Rodent Invasion in the SahEl: Global change impact on the expansion of the Nigerian Gerbil and the House Mouse in Senegal. FRB (Fondation pour la Recherche sur la Biodiversité), n° AAP-SCEN-20B III, 2014-2017.

Références

[Abouaissa *et al.*, 2014] Abouaissa, H., Yoann, K., et Morvan, G. (2014). Modélisation hybride dynamique de flux de trafic. Rapport scientifique. LGI2A, Université d'Artois.

[Allan, 2010] Allan, R. (2010). A survey of agent based modelling and simulation tools. Sci. Technol. Facil. Counc.

[Amouroux *et al.*, 2013] Amouroux, E., Huraux, T., Sempé, F., Sabouret, N., et Haradji, Y. (2013). Simulating Human Activities to Investigate Household Energy Consumption. In ICAART (2), pp. 71–80.

[Aniorté *et al.*, 2006] Aniorté, P., Cariou, E., et Gouardères, E. (2006). Modélisation de systèmes complexes distribués: l'ingénierie des modèles pour l'intégration des paradigmes «agent» et «composant». Proc. 2nd Journ. Multi-Agent Compos. JMAC Nımes *21*.

[Aplin *et al.*, 2003] Aplin, K.P., Chesser, T., et Have, J. t (2003). Evolutionary biology of the genus Rattus: profile of an archetypal rodent pest. ACIAR Monogr. Ser. *96*, pp. 487–498.

[ArrayList Java] http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#add(int,%20E), consulté 01/01/2016.

[Aubert et Müller, 2013] Aubert, S., et Müller, J.-P. (2013). Incorporating institutions, norms and territories in a generic model to simulate the management of renewable resources. Artif. Intell. Law *21*, pp. 47–78.

[Audouin-Rouzeau et Vigne, 1994] Audouin-Rouzeau, F., et Vigne, J.D. (1994). La colonisation de l'Europe par le rat noir (Rattus rattus). Rev. Paléobiol. *13*, pp. 125–145.

[Auffray *et al.*, 2011] Auffray, Y., Barbillon, P., et Marin, J.-M. (2011). Modèles réduits à partir d'expériences numériques. J. Société Fr. Stat. *152*, pp. 89–102.

[Auger *et al.*, 1992] Auger, P., Baudry, J., Fournier, F., et Menaut, J.C. (1992). Hiérarchies et échelles en écologie. Auger.

[Bâ, 2002] Bâ, K. (2002). Systematics, Ecology and Dynamics of Small Rodents Populations Potentially Reservoirs or Hosts of Viruses in Senegal. Mémoire Ecole Pratique des Hautes Etudes, Montpellier, France.

[Baduel *et al.*, 2009] Baduel, Q., Le Fur, J., et Piry, S. (2009). Chaînes de traitement et procédures pour l'exploitation de données spatialisées par un simulateur multi-agents.

[Banks *et al.*, 1999] Banks, J., Carson II, J.S., et Nelson, B.L. (1999). 1.997, Discrete-event System Simulation. N. J. E. U. Am. Prentice-Hall.

[Banks *et al.*, 2005] Banks, J., CARSON II, J.S., Barry, L., et others (2005). Discrete-event system simulationfourth edition (Pearson).

[Belem *et al.*, 2011] Belem, M., Bousquet, F., Müller, J.P., Bazile, D., et Coulibaly, H. (2011). A participatory modeling method for multi-points of view description of a system from scientist's perceptions: application in seed systems modeling in Mali and Chile.

[Bousquet *et al.*, 1998] Bousquet, F., Bakam, I., Proton, H., et Le Page, C. (1998). Cormas: common-pool resources and multi-agent systems. In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, (Springer), pp. 826–837.

[Brooks et Rowe, 1987] Brooks, J.E., et Rowe, F.P. (1987). Commensal rodent control.

[Buxton, 1936] Buxton, P.A. (1936). Breeding rates of domestic rats trapped in Lagos, Nigeria, and certain other countries. J. Anim. Ecol. pp. 53–66.

[Cedeao et Csao, 2005] Cedeao, et Csao (2005). Atlas régional des transports et des télécommunications dans la CEDEAO (Abuja et Issy-les-Moulineaux).

[Chancira a] http://www.agence-nationale-recherche.fr/suivi-bilan/editions-2013-et-anterieures/environnement-et-ressources-biologiques/societes-et-changements-environnementaux/fiche-projet-cep-s/?tx_lwmsuivibilan_pi2[CODE]=ANR-11-CEPL-0010, consulté le 01/01/2016.

[Chancira b] http://www.senegal.ird.fr/toute-l-actualite/evenements/colloques-seminaires-conferences-d-experts/colloque-de-fin-programme-chancira, consulté le 01/01/2016.

[Collier *et al.*, 2003] Collier, N., Howe, T., et North, M. (2003). Onward and upward: The transition to Repast 2.0. In Proc. First Annual North American Association for Computational Social and Organizational Science Conference.

[Comte, 2012a] Comte, A. (2012). Caractérisation des barrières à l'hybridation de deux espèces jumelles de rongeurs afri-cains du genre Mastomys.

[Comte, 2012b] Comte, A. (2012). Évaluation des capacités d'introgression par hybridation entre deux espèces de rongeurs africains du genre Mastomys – étude par simulation inf ormatique multi-agents.

[Coquillard et Hill, 1997] Coquillard, P., et Hill, D.R.C. (1997). Modélisation et simulation d'écosystèmes: des modèles déterministes aux simulations à événements discrets (Paris).

[Dalu et Feresu, 1997] Dalu, J.M., et Feresu, S.B. (1997). Domestic rodents as reservoirs of pathogenic Leptospira on two city of Harare farms: Preliminary results of bacteriological and serological studies. Belg. J. Zool. Belg.

[Daudé, 2006] Daudé, E. (2006). Du complexe à la complexité: le retour de l'individu dans les modèles géographiques. In Géopoint.

[Daudé et Langlois, 2006] Daudé, E., et Langlois, P. (2006). Introduction à la modélisation multi-agents des systèmes complexes en géographie. Modélisation Simul. Multi-Agents Pour Sci. L'Homme Société Appl. Pour Sci. L'Homme Société pp.362–366.

[De Wolf et Holvoet, 2005] De Wolf, T., et Holvoet, T. (2005). Emergence versus self-organisation: Different concepts but promising when combined. In Engineering self-organising systems, (Springer), pp. 1–15.

[DeAngelis et Mooij, 2005] DeAngelis, D.L., et Mooij, W.M. (2005). Individual-based modeling of ecological and evolutionary processes. Annu. Rev. Ecol. Evol. Syst. pp.147–168.

[Delany, 1975] Delany, M.J. (1975). The Rodents of Uganda London: British Museum of Natural History.

[Dijkstra, 1959] Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. Numer. Math. *1*, pp. 269–271.

[Drogoul *et al.*, 2013] Drogoul, A., Amouroux, E., Caillou, P., Gaudou, B., Grignard, A., Marilleau, N., Taillandier, P., Vavasseur, M., Vo, D.-A., et Zucker, J.-D. (2013). GAMA: a spatially explicit, multi-level, agent-based modeling and simulation platform. In Advances on Practical Applications of Agents and Multi-Agent Systems, (Springer), pp. 271–274.

[Duboz, 2004] Duboz, R. (2004). Intégration de modeles hétérogenes pour la modélisation et la simulation de systemes complexes. Appl. À Modélisation Multiéchelles En Écologie Mar.

[Duboz *et al.*, 2003] Duboz, R., Ramat, É., et Preux, P. (2003). Scale transfer modeling: using emergent computation for coupling an ordinary differential equation system with a reactive agent model. Syst. Anal. Model. Simul. *43*, pp. 793–814.

[Duhail, 2013] Duhail, N. (2013). DEVS et ses extensions pour des simulations multi-modèles en interaction multi-échelles. Master's thesis, Université de Rennes 1 — Telecom Bretagne.

[Duplantier *et al.*, 1991] Duplantier, J.-M., Granjon, L., Adam, F., et Bâ, K. (1991). Répartition actuelle du rat noir (Rattus rattus) au Sénégal: facteurs historiques et écologiques.

[Dyck *et al.*, 1979] Dyck, V.A., Misra, B.C., Alam, S., Chen, C.N., Hsieh, C.Y., Rejesus, R.S., et others (1979). Ecology of the brown planthopper in the tropics. Int. Rice Res. Inst. Brown Planthopper Threat Rice Prod. Asia pp. 61–98.

[Epstein, 2006] Epstein, J.M. (2006). Generative social science: Studies in agent-based computational modeling (Princeton University Press).

[Ferber, 1999] Ferber, J. (1999). Multi-agent systems: an introduction to distributed artificial intelligence (Addison-Wesley Reading).

[Ferber et Perrot, 1995] Ferber, J., et Perrot, J.-F. (1995). Les systèmes multi-agents: vers une intelligence collective (InterEditions).

[Fiedler, 1988] Fiedler, L.A. (1988). Rodent problems in Africa.

[Finkel et Bentley, 1974] Finkel, R.A., et Bentley, J.L. (1974). Quad trees a data structure for retrieval on composite keys. Acta Inform. *4*, pp. 1–9.

[FIPA, 2002] FIPA (2002). Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification, 2002. URL: http://www.fipa.org/repository/aclspecs.html, consulté le 29/08/2016.

[Fishwick, 1997] Fishwick, P.A. (1997). Computer simulation: growth through extension. Trans. Soc. Comput. Simul. *14*, pp. 13–24.

[Gasmi *et al.*, 2015] Gasmi, N., Grignard, A., Drogoul, A., Gaudou, B., Taillandier, P., Tessier, O., et An, V.D. (2015). Reproducing and exploring past events using agent-based geo-historical models. In Multi-Agent-Based Simulation XV, (Springer), pp. 151–163.

[Gaud, 2007] Gaud, N. (2007). Systèmes Multi-Agents Holoniques: de l'analyse à l'implantation. Univ. Belfort-Montbél. Lab. Systèmes Transp.

[Gaud *et al.*, 2008a] Gaud, N., Galland, S., Gechter, F., Hilaire, V., et Koukam, A. (2008). Holonic multilevel simulation of complex systems: Application to real-time pedestrians simulation in virtual urban environment. Simul. Model. Pract. Theory *16*, pp. 1659–1676.

[Gaud *et al.*, 2008b] Gaud, N., Galland, S., et Koukam, A. (2008). Towards a multilevel simulation approach based on holonic multiagent systems. In Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on, (IEEE), pp. 180–185.

[Gaudou *et al.*, 2014] Gaudou, B., Sibertin-Blanc, C., Therond, O., Amblard, F., Auda, Y., Arcangeli, J.-P., Balestrat, M., Charron-Moirez, M.-H., Gondet, E., Hong, Y., *et al.* (2014). The MAELIA Multi-Agent Platform for Integrated Analysis of Interactions Between Agricultural Land-Use and Low-Water Management Strategies. In Multi-Agent-Based Simulation XIV, (Springer), pp. 85–100.

[Gibson *et al.*, 2000] Gibson, C.C., Ostrom, E., et Ahn, T.-K. (2000). The concept of scale and the human dimensions of global change: a survey. Ecol. Econ. *32*, pp. 217–239.

[Gilbert, 2007] Gilbert, N. (2007). Agent-based models (quantitative applications in the social sciences). Thousand Oaks Annot. Ed. Sage Publ. Inc.

[Gil-Quijano *et al.*, 2009] Gil-Quijano, J., Hutzler, G., et Louail, T. (2009). De la cellule biologique à la cellule urbaine: retour sur trois expériences de modélisation multi-échelles à base d'agents. In 17ème Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2009), pp. 187–198.

[Gil-Quijano *et al.*, 2012] Gil-Quijano, J., Louail, T., et Hutzler, G. (2012). From biological to urban cells: lessons from three multilevel agent-based models. In Principles and Practice of Multi-Agent Systems, (Springer), pp. 620–635.

[Goldberg et Harrelson, 2005] Goldberg, A.V., et Harrelson, C. (2005). Computing the shortest path: A search meets graph theory. In Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, (Society for Industrial and Applied Mathematics), pp. 156–165.

[Granjon et Duplantier, 2009] Granjon, L., et Duplantier, J.-M. (2009). Les rongeurs de l'Afrique sahélo-soudanienne.

[Gratz, 1997] Gratz, N.G. (1997). The burden of rodent-borne diseases in Africa south of the Sahara. Belg. J. Zool. Belg.

[Grignard *et al.*, 2013] Grignard, A., Taillandier, P., Gaudou, B., Vo, D.A., Huynh, N.Q., et Drogoul, A. (2013). GAMA 1.6: Advancing the art of complex agent-based modeling and simulation. In PRIMA 2013: Principles and Practice of Multi-Agent Systems, (Springer), pp. 117–131.

[Grimm et Railsback, 2013] Grimm, V., et Railsback, S.F. (2013). Individual-based modeling and ecology (Princeton university press).

[Grimm *et al.*, 2005] Grimm, V., Revilla, E., Berger, U., Jeltsch, F., Mooij, W.M., Railsback, S.F., Thulke, H.-H., Weiner, J., Wiegand, T., et DeAngelis, D.L. (2005). Pattern-Oriented Modeling of Agent-Based Complex Systems: Lessons from Ecology.

[Guo et Tay, 2008] Guo, Z., et Tay, J.C. (2008). Multi-timescale event-scheduling in multi-agent immune simulation models. Biosystems *91*, pp. 126–145.

[Handschumacher *et al.*, 1992] Handschumacher, P., Hervé, J.-P., et Hébrard, G. (1992). Des aménagements hydro-agricoles dans la vallée du fleuve Sénégal ou le risque de maladies hydriques en milieu sahélien. Sci. Chang. PlanétairesSécheresse *3*, pp. 219–226.

[Haradji *et al.*, 2012] Haradji, Y., Poizat, G., et Sempé, F. (2012). Human activity and social simulation. Adv. Appl. Hum. Model. Simul. pp. 416–425.

[Hart *et al.*, 1968] Hart, P.E., Nilsson, N.J., et Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. *4*, pp. 100–107.

[Huraux *et al.*, 2014a] Huraux, T., Sabouret, N., Haradji, Y., et EDF R&D, C. (2014). Un modèle multi-niveau pour simuler l'activité humaine dans le contexte de la consommation énergétique résidentielle.

[Huraux *et al.*, 2014b] Huraux, T., Sabouret, N., et Haradji, Y. (2014). A Multi-Level Model for Multi-Agent Based Simulation. In Proc. of the 6th International Conference on Agents and Artificial Intelligence (ICAART), Angers, France.

[Kelly, 1994] Kelly, K. (1994). Out of Control: The New Biology of Machines, Social Systems and the Economic. World Addison-Wesley Read. MA.

[Koestler, 1967] Koestler, A. (1967). The ghost in the machine. Lond. Hutchinson.

[Koestler, 1978] Koestler, A. (1978). Janus: A summing up.

[Konečnỳ, 2009] Konečnỳ, A. (2009). Consequences of anthropogenic changes on rodent communities and populations (study cases on native and introduced species in Eastern Senegal).

[Konečnỳ *et al.*, 2013] Konečnỳ, A., Estoup, A., Duplantier, J.-M., Bryja, J., Bâ, K., Galan, M., Tatard, C., et Cosson, J.-F. (2013). Invasion genetics of the introduced black rat (Rattus rattus) in Senegal, West Africa. Mol. Ecol. *22*, pp. 286–300.

[Kosoy *et al.*, 2015] Kosoy, M., Khlyap, L., Cosson, J.-F., et Morand, S. (2015). Aboriginal and invasive rats of genus Rattus as hosts of infectious agents. Vector-Borne Zoonotic Dis. *15*, pp. 3–12.

[Kubera *et al.*, 2008] Kubera, Y., Mathieu, P., Picault, S., et others (2008). Interaction-Oriented Agent Simulations: From Theory to Implementation. In ECAI, pp. 383–387.

[Lammoglia, 2010] Lammoglia, A. (2010). Évolution « spatio-temporelle » d'une desserte de transport flexible simulée en sma. In Modélisation des dynamiques spatiales, p. 21.

[Langlois, 2009] Langlois, P. (2009). Une ontologie formelle pour la modélisation de systemes complexes en géographie: le modele aoc. Denis Phan Ed.

[Le Fur, 2013a] Le Fur, J. (2013). A formal framework for linking multidisciplinary multiscale knowledge.

[Le Fur, 2013b] Le Fur, J. (2013). Extending life concepts to complex systems. Interdiscip. Descr. Complex Syst. *11*, pp. 37–50.

[Le Fur *et al.*, 2016] Le Fur, J., Duplantier, J.-M., Granjon, L., Handschumacher, P., Lombard, J., Luccacioni, H., Mboup, P.A., et Ninot, O. (2016). Using simulation to infer key parameters of the black rat colonization in Senegal during the elapsed century.

[Le Fur *et al.*, 2017] Le Fur, J., Mboup, P.A., et Sall, M. (July, 2017). A simulation model for integrating multidisciplinary knowledge in natural sciences. Heuristic and application to wild rodent studies. In International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2017). (Soumis).

[Leirs, 1994] Leirs, H. (1994). Population ecology of Mastomys natalensis (Smith, 1834): implications for rodent control in Africa.

[Lepagnot et Hutzler, 2009] Lepagnot, J., et Hutzler, G. (2009). A multi-scale agent-based model for the simulation of avascular tumor growth. Jour Biol Phys Chem *9*, pp. 17–25.

[Logo, 2012] Logo (2012). The Logo Programming Language - http://en.wikipedia.org/wiki/Logo_ (programming_language), consulté le 25/08/2016.

[Lou *et al.*, 2009] Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., et Huang, Y. (2009). Map-matching for low-sampling-rate GPS trajectories. In Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems, (ACM), pp. 352–361.

[Luke *et al.*, 2005] Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., et Balan, G. (2005). Mason: A multiagent simulation environment. Simulation *81*, pp. 517–527.

[Marceau, 1999] Marceau, D.J. (1999). The scale issue in the social and natural sciences. Can. J. Remote Sens. *25*, pp. 347–356.

[Maudet *et al.*, 2014] Maudet, A., Touya, G., Duchêne, C., et Picault, S. (2014). Representation of Interactions in a Multi-Level Multi-Agent Model for Cartography Constraint Solving. In Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection, (Springer), pp. 183–194.

[Maus *et al.*, 2011] Maus, C., Rybacki, S., et Uhrmacher, A.M. (2011). Rule-based multi-level modeling of cell biological systems. BMC Syst. Biol. *5*, p. 166.

[Mboup, 2012] Mboup, P.A. (2012). Construction d'un environnement de simulation multiagents pour l'étude de la diffusion du rat noir au Sénégal au cours du siècle écoulé (Mémoire de Master, FST, UCAD).

[Mboup *et al.*, 2015a] Mboup, P.A., Handschumacher, P., Konaté, K., et Le Fur, J. (2015). Des Connaissances à la Simulation Multi-agents: Modélisation orientée événements de la Colonisation du Rat noir au Sénégal par les Transports Humains sur un Siècle. In Colloque National sur la Recherche en Informatique et ses Applications (CNRIA), (Thiès, Sénégal).

[Mboup *et al.*, 2015b] Mboup, P.A., Mboup, M.L., Konaté, K., Handschumacher, P., et Le Fur, J. (2015). Optimisation de l'utilisation de l'algorithme de Dijkstra pour un simulateur multi-agents spatialisé. In Information Technology and Computer Applications Congress (WCITCA), 2015 World Congress on, (Hammamet, Tunisia: IEEE), pp. 1–6.

[Mboup *et al.*, 2017] Mboup, P.A., Konate, K., et Le Fur, J. (June, 2017). A multi-world agent-based model working at several spatial and temporal scales for simulating complex geographic systems. In International Conference on Computational Science (ICCS). (Accepté).

[Meerburg *et al.*, 2009] Meerburg, B.G., Singleton, G.R., et Kijlstra, A. (2009). Rodent-borne diseases and their risks for public health. Crit. Rev. Microbiol.

[Morand *et al.*, 2015] Morand, S., Bordes, F., CHEN, H.-W., Claude, J., COSSON, J.-F., Galan, M., Czirják, G.Á., Greenwood, A.D., Latinne, A., Michaux, J., *et al.* (2015). Global parasite and Rattus rodent invasions: The consequences for rodent-borne diseases. Integr. Zool. *10*, pp. 409–423.

[Morse, 1995] Morse, S.S. (1995). Factors in the emergence of infectious diseases. Emerg. Infect. Dis. *1*, p. 7.

[Morvan, 2012] Morvan, G. (2012). Multi-level agent-based modeling-bibliography. CoRR Abs12050561 May 2012.

[Morvan, 2013] Morvan, G. (2013). Multi-level agent-based modeling-A literature survey. ArXiv Prepr. ArXiv12050561.

[Morvan et Jolly, 2012] Morvan, G., et Jolly, D. (2012). Multi-level agent-based modeling with the Influence Reaction principle. ArXiv Prepr. ArXiv12040634.

[Morvan et Kubera, 2014] Morvan, G., et Kubera, Y. (2014). SIMILAR : Simulations with multi-level agents and reactions. Working paper.

[Morvan *et al.*, 2011] Morvan, G., Veremme, A., et Dupont, D. (2011). IRM4MLS: the influence reaction model for multi-level simulation. In Multi-Agent-Based Simulation XI, (Springer), pp. 16–27.

[Müller, 2004] Müller, J.P. (2004). The mimosa generic modelling and simulation platform: The case of multi-agent systems.

[Müller, 2007] Müller, J.P. (2007). Mimosa: using ontologies for modeling and simulation.

[Müller, 2009] Müller, J.-P. (2009). Towards a formal semantics of event-based multi-agent simulations. In Multi-Agent-Based Simulation IX, (Springer), pp. 110–126.

[Musser et Carleton, 2005] Musser, G.G., et Carleton, M.D. (2005). Superfamily Muroidea. Mammal Species World Taxon. Geogr. Ref. 3rd Ed. Wilson DM Reeder Eds Johns Hopkins Univ. Press Baltim. USA pp. 894–1531.

[Navarro *et al.*, 2011] Navarro, L., Flacher, F., et Corruble, V. (2011). Dynamic level of detail for large scale agent-based urban simulations. In The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pp. 701–708.

[Navarro *et al.*, 2013] Navarro, L., Corruble, V., Flacher, F., et Zucker, J.-D. (2013). A flexible approach to multi-level agent-based simulation with the mesoscopic representation. In Proceedings of the 2013 International Foundation for Autonomous Agents and Multiagent Systems, pp. 159–166.

[North *et al.*, 2013] North, M.J., Collier, N.T., Ozik, J., Tatara, E.R., Macal, C.M., Bragen, M., et Sydelko, P. (2013). Complex adaptive systems modeling with repast simphony. Complex Adapt. Syst. Model. *1*, pp. 1–26.

[OZIK, 2012] OZIK, J. (2012). RELOGO GETTING STARTED GUIDE.

[Pachet *et al.*, 1994] Pachet, F., Giroux, S., et Paquette, G. (1994). Pluggable Advisors as Epiphyte Systems. Calisce L94 Comput. Aided Learn. Sci. Eng. Paris pp. 167–174.

[Picault et Mathieu, 2011] Picault, S., et Mathieu, P. (2011). An interaction-oriented model for multi-scale simulation. In IJCAI'2011–Barcelona (Spain)–July, 16-22 2011, (AAAI Press), pp. 332–337.

[Piélou, 1975] Piélou, E.C. (1975). Ecological Diversity, John Wiley and Sons, New York, p. 165.

[Pollitzer, 1954] Pollitzer, R. (1954). La peste.

[Pumain, 2006] Pumain, D. (2006). Hierarchy in Natural and Social Sciences, vol. 3 of Methodos Series (Springer Netherlands, Dodrecht, The Netherlands).

[Pumain, 2012] Pumain, D. (2012). Multi-agent system modelling for urban systems: The series of SIMPOP models. In Agent-based models of geographical systems, (Springer), pp. 721–738.

[Quddus et Washington, 2015] Quddus, M., et Washington, S. (2015). Shortest path and vehicle trajectory aided map-matching for low frequency GPS data. Transp. Res. Part C Emerg. Technol. *55*, pp. 328–339.

[Quijano-Gil *et al.*, 2010] Quijano-Gil, J., Hutzler, G., et Louail, T. (2010). Accroche-toi au niveau, j'enlève l'échelle. Éléments d'analyse des aspects multiniveaux dans la simulation à base d'agents. Rev. Intell. Artif. *24*, pp. 625–648.

[Railsback et Grimm, 2011] Railsback, S.F., et Grimm, V. (2011). Agent-based and individual-based modeling: a practical introduction (Princeton university press).

[Ratzé *et al.*, 2007] Ratzé, C., Gillet, F., Müller, J.-P., et Stoffel, K. (2007). Simulation modelling of ecological hierarchies in constructive dynamical systems. Ecol. Complex. *4*, pp. 13–25.

[Resnick, 1994] Resnick, M. (1994). Turtles, termites, and traffic jams (MIT Press, Cambridge, MA).

[Rosevear, 1969] Rosevear, D.R. (1969). The rodents of west Africa (British Museum (Natural History) London).

[Saluzzo *et al.*, 1985] Saluzzo, J.F., Digoutte, J.P., Camicas, J.-L., et Chauvancy, G. (1985). Crimean-Congo haemorrhagic fever and Rift Valley fever in south-eastern Mauritania. The Lancet *325*, pp. 116.

[Schrag et Wiener, 1995] Schrag, S.J., et Wiener, P. (1995). Emerging infectious disease: what are the relative roles of ecology and evolution? Trends Ecol. Evol. *10*, pp. 319–324.

[Servat *et al.*, 1998a] Servat, D., Perrier, E., Treuil, J.-P., et Drogoul, A. (1998). When agents emerge from agents: Introducing multi-scale viewpoints in multi-agent simulations. In Multi-Agent Systems and Agent-Based Simulation, (Springer), pp. 183–198.

[Servat *et al.*, 1998b] Servat, D., Perrier, E., Treuil, J.-P., et Drogoul, A. (1998). Towards virtual experiment laboratories: How multi-agent simulations can cope with multiple scales of analysis and viewpoints. In International Conference on Virtual Worlds, (Springer), pp. 205–217.

[Shaw et Wagner, 2008] Shaw, K.L., et Wagner, K. (2008). Cricketsim: a Genetic and Evolutionary Computer Simulation. J. Artif. Soc. Soc. Simul. *11*, p. 3.

[Simon, 1962] Simon, H.A. (1962). The Architecture of Complexity', reprinted in Simon (1969). Simon84The Archit. Complexity1962 pp. 84–118.

[Sinou, 1981] Sinou, A. (1981). Key periods in the foundation of some colonial cities. Cahiers d'Etudes Africaines, 81–83, 375–388.

[Sinou *et al.*, 1989] Sinou, A., Poinsot, J., et Strenadel, J. (1989). Les villes d'Afrique noire: politiques et opérations d'urbanisme et d'habitat entre 1650 et 1960.

[Skonhoft *et al.*, 2006] Skonhoft, A., Leirs, H., Andreassen, H.P., Mulungu, L.S., et Stenseth, N.C. (2006). The bioeconomics of controlling an African rodent pest species. Environ. Dev. Econ. *11*, pp. 453–475.

[Smithers et Black, 1975] Smithers, R.H., et Black, R.A.R. (1975). Guide to the rats and mice of Rhodesia (Trustees of the National Museums and Monuments of Rhodesia).

[Solovyev *et al.*, 2010] Solovyev, A., Mikheev, M., Zhou, L., Dutta-Moscato, J., Ziraldo, C., An, G., Vodovotz, Y., et Mi, Q. (2010). SPARK: a framework for multi-scale agent-based biomedical modeling. In Proceedings of the 2010 Spring Simulation Multiconference, (Society for Computer Simulation International), p. 3.

[Soyez, 2013] Soyez, J.B. (2013). Conception et modélisation de systemes de systemes: une approche multi-agents multi-niveaux. Université de Lille 1.

[Soyez *et al.*, 2012] Soyez, J.-B., Morvan, G., Dupont, D., et Merzouki, R. (2012). A Methodology to Engineer and Validate Dynamic Multi-level Multi-agent Based Simulations. In MABS, (Springer), pp. 130–142.

[Steiniger *et al.*, 2012] Steiniger, A., Kruger, F., et Uhrmacher, A.M. (2012). Modeling agents and their environment in multi-level-DEVS. In Simulation Conference (WSC), Proceedings of the 2012 Winter, (IEEE), pp. 1–12.

[Taillandier, 2015] Taillandier, P. (2015). La modélisation du temps dans la simulation à base d'agents. Inf. Géographique *79*, pp. 65–78.

[Taillandier *et al.*, 2012] Taillandier, P., Vo, D.-A., Amouroux, E., et Drogoul, A. (2012). GAMA: a simulation platform that integrates geographical information data, agent-based modeling and multi-scale control. In Principles and Practice of Multi-Agent Systems, (Springer), pp. 242–258.

[Taillandier *et al.*, 2014] Taillandier, P., Grignard, A., Gaudou, B., et Drogoul, A. (2014). Des données géographiques à la simulation à base d'agents: application de la plate-forme GAMA. Cybergeo Eur. J. Geogr.

[Tatara et North, 2012] Tatara, E.R., et North, M.J. (2012). REPAST FLOWCHART GETTING STARTED.

[Taylor *et al.*, 2008] Taylor, P.J., Arntzen, L., Hayter, M., Iles, M., Frean, J., et Belmain, S. (2008). Understanding and managing sanitary risks due to rodent zoonoses in an African city: beyond the Boston Model. Integr. Zool. *3*, pp. 38–50.

[Tranouez, 2005] Tranouez, P. (2005). Contribution à la modélisation et à la prise en compte informatique de niveaux de descriptions multiples. Application aux écosystèmes aquatiques Penicillo haere, nam scalas aufero. Université du Havre.

[Tranouez *et al.*, 2006] Tranouez, P., Bertelle, C., et Olivier, D. (2006). Changing levels of description in a fluid flow simulation. In Emergent Properties in Natural and Artificial Dynamical Systems, (Springer), pp. 87–99.

[Trape *et al.*, 1996] Trape, J.-F., Godeluck, B., Diatta, G., Rogier, C., Legros, F., Albergel, J., Pepin, Y., et Duplantier, J.-M. (1996). The spread of tick-borne borreliosis in West Africa and its relationship to sub-Saharan drought. Am. J. Trop. Med. Hyg. *54*, pp. 289–293.

[Treuil *et al.*, 2008] Treuil, J.P., Drogoul, A., et Zucker, J.D. (2008). Modélisation et simulation à base d'agents: Approches particulaires, modèles à base d'agents, de la mise en pratique aux questions théoriques (Dunod).

[Tsunami, 2004] Tsunami (2004). Asian tsunami 2004 (Indian ocean tsunami) - http://www.tsunami2004.net/, consulté le 01/01/2016.

[Uhrmacher *et al.*, 2007] Uhrmacher, A.M., Ewald, R., John, M., Maus, C., Jeschke, M., et Biermann, S. (2007). Combining micro and macro-modeling in devs for computational biology. In Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come, (IEEE Press), pp. 871–880.

[Unhelkar, 2005] Unhelkar, B. (2005). Practical Object Oriented Analysis (Social Science Press).

[Varenne a, 2010] Varenne a, F. (2010). Framework for Models & Simulations with Agents in regard to Agent Simulations in Social Sciences: Emulation and Simulation, in A. Muzy, D. Hill & B. Zeigler (eds), Modeling & Simulation of Evolutionary Agents in Virtual Worlds, 2009, Cargèse, Corsica, pp. 49-80.

[Vo, 2012] Vo, D.A. (2012). An operational architecture to handle multiple levels of representation in agent-based models. Université Pierre et Marie Curie-Paris 6.

[Vo *et al.*, 2012] Vo, D.-A., Drogoul, A., et Zucker, J.-D. (2012). An Operational Meta-Model for Handling Multiple Scales in Agent-Based Simulations. In 2012 IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), pp. 1-6.

[Wilensky, 1999a] Wilensky, U. (1999). NetLogo - http://ccl.northwestern.edu/netlogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

[Wilensky, 1999b] Wilensky, U. (1999). NetLogo.

[Worboys, 2005] Worboys, M. (2005). Event-oriented approaches to geographic phenomena. Int. J. Geogr. Inf. Sci. *19*, pp. 1–28.

[Zeigler *et al.*, 2000] Zeigler, B.P., Praehofer, H., et Kim, T.G. (2000). Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems (Academic press).

Annexe A
Le projet CHANCIRA

Le projet CHANCIRA (CHANgements environnementaux, CIrculation de biens et de personnes : de l'invasion de réservoirs à l'apparition d'anthropozoonoses, le cas du RAt noir dans l'espace sénégalo-malien) est un projet s'inscrivant dans le débat sur les dynamiques spatio-temporelles et multi-échelles des zoonoses à travers la compréhension des processus de diffusion/propagation des réservoirs et de passage à l'homme des agents infectieux. Il est financé par l'ANR (Agence Nationale de la Recherche) entre le 1er novembre 2011 et le 31 octobre 2015. Il a englobé 4 équipes (CBGP, SESSTIM, PRODIG, IPD), 7 universités (Lille, Marseille, Montpellier, Paris I, Strasbourg, Dakar, Ziguinchor) et 6 Organismes de recherches (IRD, CNRS, IPD, INRA, SWISS TPH).

Face aux interrogations actuelles sur la diffusion de maladies animales passant à l'homme (anthropozoonoses) et de la contamination d'espaces jusque-là indemnes, ce projet vise à illustrer et à comprendre les processus qui régissent la diffusion des hôtes et le passage à l'homme des pathogènes à travers la dynamique du rat noir et des maladies dont il est porteur dans l'espace sénégalo-malien.

En vue d'identifier les maillons faibles de la diffusion et de la transmission de maladies par le rat noir (réservoirs de maladies transmissibles à l'homme), ce projet visait à fournir de la connaissance pour la lutte et la prévention. Ainsi il visait à répondre aux questions que sont :

- comment identifier les processus qui génèrent l'expansion des rats noirs ?
- comment les rats noirs diffusent-ils le risque de zoonoses au Sénégal?
- comment identifier et comprendre les conditions biologiques et sociales dans lesquelles les viroses dont les rats noirs sont porteurs passent à l'homme ?

L'objectif de ce projet est donc de comprendre comment les modifications climatiques et surtout anthropiques, qui affectent les pays de la bande soudano-sahélienne, interagissent pour créer des conditions propices à la diffusion d'un rongeur envahissant, *Rattus rattus*, réservoir redouté d'anthropozoonoses car proche de l'homme, et au passage des pathogènes à l'homme dans des conditions spécifiques de transmission. C'est par une approche multi-échelle de l'espace zonal à l'échelle de l'habitation que ce projet vise à comprendre comment de multiples déterminants interagissent (climat, population, mode de vie, mobilités, mode d'habitat et mode d'habiter…).

En s'appuyant sur les théories des systèmes de diffusion, en produisant de la connaissance de terrain mais aussi en valorisant la connaissance issue d'archives, ce projet visait la modélisation du risque en fonction de scenarii crédibles d'aménagement et de transformation de l'espace. Il cherchait ainsi à apporter et valider des stratégies de prévention non médicales en amont même du risque sanitaire et de proposer aux acteurs et aménageurs de terrain la connaissance nécessaire à la mise en œuvre de mesures préventives face à la diffusion du risque sanitaire.

I. Méthodologie : De la mobilisation d'archives à la production de connaissances interdisciplinaires de terrain pour la modélisation.

La méthode s'appuie sur la confrontation de corpus de données climatiques, historiques, démographiques, biologiques, relatives à l'urbanisation et aux transports. Des données préalablement collectées dans les archives sont comparées avec des mesures actuelles. Des enquêtes de terrain à échelle fine (comptage de flux, capture de rongeurs, de vecteurs, prélèvements sanguins, enquêtes domiciliaires socio-économiques) sont réalisées sur des sites ateliers. Une base de données spatialisée est construite afin de permettre l'articulation des données. Les dynamiques spatiales sont analysées à partir de marqueurs de la diffusion des réservoirs de pathogènes et de l'étude des dynamiques des flux de biens et de personnes, et des territoires.

La dimension récente de la diffusion et le passage à l'homme des anthropozoonoses sont étudiés sur deux axes connaissant une dynamique en cours, et dans des espaces ateliers inscrits dans des villages correspondant à des degrés de vulnérabilité différenciés :

- l'axe Tambacounda – Kédougou au sud, climatiquement favorable au rat noir, ouvert depuis les années 1990 dans un contexte de cul de sac produisant un enclavement formel mais propice aux circulations informelles entre Sénégal et Mali.
- l'axe Tambacounda – Kayes au nord, climatiquement défavorable, mais en augmentation de trafic à travers une frontière ouverte.

L'analyse de ces données à de multiples échelles temporelles et spatiales fait appel aux techniques d'analyses statistiques multi-niveaux et aux fonctionnalités des Systèmes d'Informations Géographiques et débouchent sur des perspectives de modélisation spatiale.

II. L'organisation des tâches

Le projet est divisé en 6 tâches. Celle dans laquelle cette thèse s'inscrit est la tâche qui concerne l'articulation interdisciplinaire et la production d'indicateurs de diffusion du rat noir et des risques sanitaires chez l'homme. D'autres informations sur le projet CHANCIRA peuvent être retrouvées dans [Chancira a; Chancira b]

III. Problématique liée au projet CHANCIRA

Le développement de la société sénégalaise depuis un siècle a conduit à des changements spectaculaires et multiples concernant tant l'usage des terres que la répartition des populations humaines, la dynamique des villes ou l'accélération des flux de population et de biens [Cedeao et Csao, 2005]. Ces changements s'accompagnent le plus souvent de conséquences pour la santé publique avec un développement concomitant des vecteurs de risques épidémiologiques.

Si les causes immédiates de l'apparition de phénomènes épidémiques peuvent être clairement identifiées (*e.g.*, barrage et épidémie de schistosomose intestinale dans la vallée du fleuve Sénégal, [Handschumacher *et al.*, 1992]), les processus soutenant l'introduction de nouveaux pathogènes et l'apparition de nouvelles maladies sont plus difficiles à cerner [Taylor *et al.*, 2008]. Il est en effet souvent difficile d'identifier les réels déterminants de risque [Morse, 1995; Schrag et Wiener, 1995] car les processus ont des origines souvent multifactorielles, s'expriment dans des contextes en mutation et ont des poids différents selon l'échelle considérée [Piélou, 1975].

Parmi les facteurs de risque au Sénégal, le rat noir (*Rattus rattus*) représente un enjeu fort en termes de santé publique car il est hôte de nombreuses maladies transmissibles à l'homme. Outre la peste qui ne sévit plus pour le moment en Afrique de l'Ouest, il est porteur d'agents de leptospirose, typhus murin, borrélioses, divers hantavirus [Granjon et Duplantier, 2009]. Plus généralement, sa proximité à l'homme en fait un élément majeur de la diffusion et de la propagation de risques sanitaires.

Dans cette dynamique, l'un des obstacles majeurs à surmonter tient au fait que les problèmes à résoudre résultent de processus multiples qui opèrent et doivent être saisis à des échelles de temps, d'espace et d'analyse différentes [Auger *et al.*, 1992; Pumain, 2006].

Dans le cas de la colonisation du rat noir au Sénégal il faut tenir compte des processus qui se sont déroulés sur le siècle écoulé à l'échelle du Sénégal, dans les vingt dernières années de développement à une échelle plus régionale et dans les centres urbains de ces mêmes régions à l'échelle quotidienne du contact entre rats et hommes.

Dans son effort pour décrypter des mécanismes complexes, la recherche doit construire et utiliser des modèles. Dans ce processus de compréhension, l'apport de la simulation est souvent un atout important pour deux raisons. En termes de compréhension, elle permet d'articuler des mécanismes de natures diverses et d'indiquer les relations entre processus se déroulant à plusieurs échelles. En termes de décision, les expériences numériques apparaissent dans un grand nombre de problématiques actuelles car elles permettent de remplacer des expériences difficilement ou non réalisables *in situ* [Auffray *et al.*, 2011].

Ce travail s'inscrit ainsi dans le cadre d'une approche de modélisation multi-échelle, il vise à représenter comment s'articulent les déterminants du risque au sein de systèmes environnementaux complexes tels que celui représenté au Sénégal par l'évolution des flux de personnes, rongeurs, et pathogènes aux échelles nationales, régionales et urbaines.

La thèse vise à explorer et exploiter le potentiel des outils de simulation pour construire une image intégrée de la connaissance acquise pouvant être mise en œuvre dans le cadre d'un processus de décision. Ce travail pourra répondre à plusieurs questions issues d'ingénieurs, d'experts ou de scientifiques et qui pourront être de la forme : « quels sont les facteurs déterminant la diffusion du rat et des risques sanitaires qu'il transporte ? » ; « quels facteurs influent sur le risque sanitaire global ? ».

Annexe B
Les algorithmes

Nous commençons par présenter les algorithmes des fonctions *ajoutListeAgent()*, *ordonancerActivitesAgents()* et *changementDeLaDTM()*. Elles sont assez longues mais assez faciles à comprendre et sont divisées en des parties. On peut aussi à tout moment retourner aux explications dans la section II.A.3.c, p. 51 pour une meilleure compréhension.

Algorithme 1 : Fonction ajoutListeAgent (List listeAgent**, entier** DUA**)**

```
Variables
    tableau1[2], tableau2[5], tableau3 [5] : tableaux d'entiers ;
    // On suppose que DTM est connu
Début
  ////// Partie I)
    Si (DUA >= DTM)
      Si (DUA % DTM == 0);                          // Si DUA est un multiple de DTM
        tableau1[0] = DUA / DTM ;                   // DUA / DTM = N1
        tableau1[1] = DUA;
        multiple_DTM_ListeAgent.mettre (listeAgent, tableau1);
      Sinon                                         // Si DUA n'est pas un multiple de DTM
        N1 = arrondi par excès de (DUA / DTM);
        M = arrondi par défaut de (DUA x 10 / DTM);
        tableau2 [0] = N1;
        tableau2 [1] = (M - 10) / (N1 - 1);                 // B
        tableau2 [2] = 1;                                   // Compteur b
        tableau2 [3] = 1;                                   // Compteur c
        tableau2 [4] = DUA;
        non_multiple_DTM_ListeAgent.mettre (listeAgent, tableau2);
      Fin si
  ////// Partie II)
    Sinon Si (DUA < DTM)
      Si (DTM % DUA == 0)                           // Si DUA est un diviseur de DTM
        tableau1[0] = DTM / DUA;                    // DTM / DUA = N2
        tableau1[1] = DUA;
        diviseur_DTM_ListeAgent.mettre (listeAgent, tableau1);
      Sinon                                         // Si DUA n'est pas un diviseur de DTM
        N2 = arrondi par excès de (DTM / DUA);
        M = arrondi par défaut de (DTM x 10 / DUA);
        tableau3[0] = N2;
        tableau3[1] = (M - 10) / (N1 - 1);                  //B
        tableau3[2] = 1;                                    // Compteur b
        tableau3 [3] = DUA;
        non_diviseur_DTM_ListeAgent.mettre (listeAgent, tableau3);
      Fin si
    Fin si
Fin fonction
```

Algorithme 2 : Fonction ordonancerActivitesAgents()

```
Variables
    B, b, c, N : entiers
    //Tableau temporaire pour contenir les listes d'agents devant avoir la main plusieurs fois dans un tick.
    diviseur_DTM_ListeAgentTmp : Tableau associatif
    //Liste pour contenir les listes d'agents qui ont épuisé leur nombre de mains dans un tick.
    listesPretes : tableau de liste d'agents
    //chaque agent a une fonction doStep.
Début
    //Nous rappelons que, multiple_DTM_ListeAgent, non_multiple_DTM_ListeAgent,
    diviseur_DTM_ListeAgent, //non_diviseur_DTM_ListeAgent et compteurTickMonde sont des champs de
    l'ordonnanceur (ou Protocol).
    // N, B, b et c sont aussi déjà définis.
////// Partie I)
    // Lecture et prise en compte des paramètres en cours de simulation
    lectureDesParametres();
    compteurTickMonde ++; // compteurTickMonde est initialisé à 0 et s'incrémente par pas de 1 ici.
    calendrier.incrementDate() ;
////// Partie II) Si DUA est un multiple de DTM
    Pour (chaque listeAgent de multiple_DTM_ListeAgent)
        N = multiple_DTM_ListeAgent.get(listeAgent)[0];
        Si (compteurTickMonde % N == 0)
            Pour (chaque agent de listeAgent)
                agent.doStep() ;
            Fin pour
        Fin si
    Fin pour
////// Partie III) Si DUA est supérieur à DTM mais n'étant pas son multiple
    Pour (chaque listeAgent de non_multiple_DTM_ListeAgent)
        N = non_multiple_DTM_ListeAgent.get(listeAgent)[0];
        B = non_multiple_DTM_ListeAgent.get(listeAgent)[1];
        b = non_multiple_DTM_ListeAgent.get(listeAgent)[2];
        c = non_multiple_DTM_ListeAgent.get(listeAgent)[3];
        Si (1 <= b et b <= B)
            Si (c % N == 0)              // c remplace le compteurTickMonde
                Pour (chaque agent de listeAgent)
                    agent.doStep() ;
                Fin pour
                non_multiple_DTM_ListeAgent.get(listeAgent)[2]++;        // b++
            Fin si
            non_multiple_DTM_ListeAgent.get(listeAgent)[3]++;            // c++
        Sinon si (B < b et b <= 10)
            Pour (chaque agent de listeAgent)
                agent.doStep() ;
```

```
        Fin pour
        non_multiple_DTM_ListeAgent.get(listeAgent)[2]++;          // b++
      Fin si
      Si (b == 10)
        non_multiple_DTM_ListeAgent.get(listeAgent)[2] = 1;        // b = 1
        non_multiple_DTM_ListeAgent.get(listeAgent)[3] = 1;        // c = 1
      Fin si
    Fin pour
////// Partie IV) Si DUA est un diviseur de DTM
    Pour (chaque listeAgent de diviseur_DTM_ListeAgent)
      N = diviseur_DTM_ListeAgent.get(listeAgent)[0];
      diviseur_DTM_ListeAgentTmp.put(listeAgent, N);
    Fin pour
////// Partie V) Ici on recense les agents devant avoir la main plusieurs fois dans un pas de simulation.
    // Si DUA est inférieur à DTM mais n'étant pas son diviseur
    Pour (chaque listeAgent de non_multiple_DTM_ListeAgent)
      B = non_diviseur_DTM_ListeAgent.get(listeAgent)[1];
      b = non_diviseur_DTM_ListeAgent.get(listeAgent)[2];
      Si (b <= B)
        N = non_diviseur_DTM_ListeAgent.get(listeAgent)[0];
        diviseur_DTM_ListeAgentTmp.put(listeAgent, N);
      Sinon          // Si (B < b et b <= 10), 1 fois
        diviseur_DTM_ListeAgentTmp.put(listeAgent, 1);
      Fin si
      non_diviseur_DTM_ListeAgent.get(listeAgent)[2]++;            //b++
      Si (b == 10)
        non_diviseur_DTM_ListeAgent.get(listeAgent)[2] = 1;
      Fin si
    Fin pour
////// Partie VI) Ici on donne de façon mélangée la main aux agents devant en avoir plusieurs en un tick
    Faire
      Pour (chaque listeAgent de diviseur_DTM_ListeAgentTmp)
        a = 0 ;
        N = diviseur_DTM_ListeAgentTmp.get(listeAgent);
        Si (N > 0)
          Pour (chaque agent de listeAgent)
            agent.doStep();
          Fin pour
          diviseur_DTM_ListeAgentTmp.put(listeAgent, N - 1) ;
          a = a + N - 1 ;
        Sinon
          listesPretes.mettre(listeAgent) ;
        Fin si
      Fin pour
```

```
        diviseur_DTM_ListeAgentTmp.enleve(listesPretes) ;
        listesPretes.vider() ;
    Tant que (a > 0)
Fin fonction
```

Algorithme 3 : Fonction changementDeLaDTM()

```
Début
    ////// Faire des copies les tableaux associatifs et ensuite les vider
    multiple_DTM_ListeAgent_Copie.mettreTout(multiple_DTM_ListeAgent);
    non_multiple_DTM_ListeAgent_Copie.mettreTout(non_multiple_DTM_ListeAgent);
    diviseur_DTM_ListeAgent _Copie.mettreTout(diviseur_DTM_ListeAgent);
    non_diviseur_DTM_ListeAgent _Copie.mettreTout(non_diviseur _DTM_ListeAgent);
    ////// Vidange
    multiple_DTM_ListeAgent.vider();
    non_multiple_DTM_ListeAgent.vider();
    diviseur_DTM_ListeAgent.vider();
    non_diviseur _DTM_ListeAgent.vider();
    ////// Reconstruire les tableaux associatifs en utilisant de nouveau la méthode ajoutListeAgent
    Pour (chaque listeAgent de multiple_DTM_ListeAgent_Copie)
        DUA = multiple_DTM_ListeAgent_Copie.get(listeAgent)[1];
        ajoutListeAgent (listeAgent, DUA);
    Fin pour
    Pour (chaque listeAgent de non_multiple _DTM_ListeAgent_Copie)
        DUA = non_multiple _DTM_ListeAgent_Copie.get(listeAgent)[4];
        ajoutListeAgent (listeAgent, DUA);
    Fin pour
    Pour (chaque listeAgent de diviseur _DTM_ListeAgent_Copie)
        DUA = diviseur _DTM_ListeAgent_Copie.get(listeAgent)[1];
        ajoutListeAgent (listeAgent, DUA);
    Fin pour
    Pour (chaque listeAgent de non_diviseur _DTM_ListeAgent_Copie)
        DUA = non_diviseur _DTM_ListeAgent_Copie.get(listeAgent)[3];
        ajoutListeAgent (listeAgent, DUA);
    Fin pour
Fin fonction
```

Nous présentons les algorithmes des fonctions *ajoutMonde()* et *ordonnancerMainsMondes()* (voir les sections I.D.2 et I.D.3). Ils sont assez longs mais aussi assez faciles à comprendre et sont divisés en parties. On peut aussi à tout moment retourner aux explications dans les sections I.D.1, p. 79 et I.D.2, p. 84 pour une meilleure compréhension.

Algorithme 4 : Fonction ajoutMonde(Monde monde**)**

```
Variables
    tableau1[1], tableau2[4], tableau3 [3] : tableaux d'entier ;
    A : entier ;
    // On suppose que DTG est connu
Début
    DTM = monde.DTM;                                  //récupération du DTM du monde

////// Partie I)
    Si (DTM >= DTG)
        Si (DTM % DTG == 0);                          // Si DTM est un multiple de DTG
            tableau1[0] = DTM / DTG;                  // DTM / DTG = N1
            multiple_DTG_Mondes.mettre (monde, tableau1);
        Sinon                                         // Si DTM n'est pas un multiple de DTG
            N1 = arrondi par excès de (DTM / DTG);
            A = arrondi par défaut de (DTM x 10 / DTG);
            tableau2 [0] = N1;
            tableau2 [1] = (A - 10) / (N1 - 1);                    // B
            tableau2 [2] = 1;                                      // Compteur b
            tableau2 [3] = 1;                                      // Compteur c
            non_multiple_DTG_Mondes.mettre (monde, tableau2);
            monde.calendrier.champsIncrementation =
            calendrierGlobal.champsIncrementation;
        Fin si
////// Partie II)
    Sinon Si (DTM < DTG)
        Si (DTG % DTM == 0)                           // Si DTM est un diviseur de DTG
            tableau1[0] = DTG / DTM;                  // DTG / DTM = N2
            diviseur_DTG_Mondes.mettre (monde, tableau1);
        Sinon                                         // Si DTM n'est pas un diviseur de DTG
            N2 = arrondi par excès de (DTG / DTM);
            A = arrondi par défaut de (DTG x 10 / DTM);
            tableau3 [0] = N2;
            tableau3 [1] = (A - 10) / (N1 - 1);                    //B
            tableau3 [2] = 1;                                      // Compteur b
            non_diviseur_DTG_Mondes.mettre (monde, tableau3);
            monde.calendrier.champsIncrementation =
            calendrierGlobal.champsIncrementation;
        Fin si
    Fin si
Fin fonction
```

Algorithme 5 : Fonction ordonnancerMainsMondes()

```
Variables
    B, b, c, N : entiers
    //Tableau temporaire pour contenir les mondes devant avoir la main plusieurs fois dans un tick.
    diviseur_DTG_MondesTmp : Tableau associatif
    //Liste pour contenir les mondes qui ont épuisé leur nombre de mains dans un tick.
    listeMondesFinis : tableau de mondes

    //compteurTickGlobal est connu, il est géré par Repast Simphony.
    //chaque monde a une fonction ordonancerActivitesAgents.
Début
    //Nous rappelons que, multiple_DTG_Mondes, non_multiple_DTG_Mondes, diviseur_DTG_Mondes et
    //non_diviseur_DTG_Mondes sont des tableaux associatifs de la classe ordonnanceur globale.
    // N, B, b et c sont aussi déjà définis

////// Partie I)
    calendrierGlobal.incrementDate();
    // Lecture et prise en compte des paramètres en cours de simulation (voir I.D.4.b ci-dessus)
    lectureDesParametres();
    Si (mondeChangeantDTM n'est pas vide)
        priseEnCompteMondeChangeantDTM(); //algorithme de cette fonction dans la section suivante.
    Fin si
////// Partie II)
    // Si DTM est un multiple de DTG :
    Pour (chaque monde de multiple_DTG_Mondes)
        N = multiple_DTG_Mondes.get(monde)[0];
        Si (compteurTickGlobal % N == 0)
            monde.ordonancerActivitesAgents();
        Fin si
    Fin pour
////// Partie III)
    // Si DTM est supérieur à DTG mais n'étant pas son multiple
    Pour (chaque monde de non_multiple_DTG_Mondes)
        N = non_multiple_DTG_Mondes.get(monde)[0];
        B = non_multiple_DTG_Mondes.get(monde)[1];
        b = non_multiple_DTG_Mondes.get(monde)[2];
        c = non_multiple_DTG_Mondes.get(monde)[3];
        Si (1 <= b et b <= B)
            Si (c % N == 0)              // c remplace le compteurTickGlobal
                monde.calendrier.tickAmountMultiplicator = N;
                monde.ordonancerActivitesAgents();
                non_multiple_DTG_Mondes.get(monde)[2]++;        // b++
            Fin si
            non_multiple_DTG_Mondes.get(monde)[3]++;           // c++
```

```
        Sinon si (B < b et b <= 10)
            monde.calendrier.tickAmountMultiplicator = 1;
            monde.ordonancerActivitesAgents();
            non_multiple_DTG_Mondes.get(monde)[2]++;          // b++
        Fin si
        Si (b == 10)
            non_multiple_DTG_Mondes.get(monde)[2] = 1;        // b = 1
            non_multiple_DTG_Mondes.get(monde)[3] = 1;        // c = 1
        Fin si
    Fin pour

// Dans les parties IV et V les mondes obtiennent leur 1ère main et auront celles qui restent dans la partie VI.
////// Partie IV) Ici on recense les mondes devant avoir la main plusieurs fois dans un pas de simulation :
  // Si DTM est un diviseur de DTG :
    Pour (chaque monde de diviseur_DTG_Mondes)
        N = diviseur_DTG_Mondes.get(monde);
        diviseur_DTG_MondesTmp.put(monde, N);
    Fin pour
////// Partie V) Si DTM est inférieur à DTG mais n'étant pas son diviseur :
    Pour (chaque monde de non_diviseur_DTG_Mondes)
        B = non_diviseur_DTG_Mondes.get(monde)[1];
        b = non_diviseur_DTG_Mondes.get(monde)[2];
        monde.calendrier.tickAmountMultiplicator = 1;
        Si (b <= B)
            N = non_diviseur_DTG_Mondes.get(monde)[0];
            monde.ordonancerActivitesAgents();
            monde.calendrier.tickAmountMultiplicator = 0;
            diviseur_DTG_MondesTmp.put(monde, N-1);
        Sinon          // Si (B < b et b <= 10), 1 fois
            monde.ordonancerActivitesAgents();
        Fin si
        non_diviseur_DTG_Mondes.get(monde)[2]++;              //b++
        Si (b == 10)
            non_diviseur_DTG_Mondes.get(monde)[2] = 1;        //b = 1
        Fin si
    Fin pour
////// Partie VI) Ici on donne de façon mélangée la main aux mondes devant en avoir plusieurs en un tick
    Faire
        Pour (chaque monde de diviseur_DTG_MondesTmp)
            a = 0 ;
            N = diviseur_DTG_MondesTmp.get(monde);
            Si (N > 0)
                monde.ordonancerActivitesAgents();
                diviseur_DTG_MondesTmp.put(monde, N - 1) ;
                a = a + N - 1 ;
```

```
            Sinon
                listeMondesFinis.mettre(monde) ;
            Fin si
        Fin pour
        diviseur_DTG_MondesTmp.enleve(listeMondesFinis) ;
        listeMondesFinis.vider() ;
    Tant que (a > 0)
Fin fonction
```

Algorithme 6 : Fonction ajoutMondeChangeantDTM (Monde monde**, Décimal** ancienDTM**)**

```
Debut
    mondesChangeantDTM.mettre(monde, ancienDTM) ;
Fin fonction
```

Algorithme 7 : Fonction priseEnCompteMondeChangeantDTM()

```
Variables
    ancienDTM : Décimal;
Debut
    //on enlève les mondes de leur ancienne liste et on les ajoute de nouveau dans les bonnes listes
    Pour (chaque monde de mondesChangeantDTM)
        ancienDTM = mondesChangeantDTM.obtenir (monde) ;
        Si(ancienDTM >= DTG)
            Si(ancienDTM % DTG == 0)
                multiple_DTG_Mondes.enlève(monde) ;
            Sinon
                non_multiple_DTG_Mondes.enlève(monde) ;
            Fin si
        Sinon
            Si(DTG % ancienDTM == 0)
                diviseur_DTG_Mondes.enlève(monde) ;
            Sinon
                non_diviseur_DTG_Mondes.enlève(monde) ;
            Fin si
        Fin si
        ajoutMonde(monde) ;                    //fonction ajoutMonde déjà définie.
    Fin Pour
Fin fonction
```

Algorithme 8 : Fonction changementDeLaDTG()

```
Variables
    multiple_DTG_Mondes_Copie, non_multiple_DTG_Mondes_Copie,
    diviseur_DTG_Mondes_Copie et non_diviseur_DTG_Mondes _Copie : tableaux
    associatifs
Debut
  // Faire des copies les tableaux associatifs et ensuite les vider
    multiple_DTG_Mondes_Copie.mettreTout(multiple_DTG_Mondes);
    non_multiple_DTG_Mondes_Copie.mettreTout(non_multiple_DTG_Mondes);
    diviseur_DTG_Mondes _Copie.mettreTout(diviseur_DTG_Mondes);
    non_diviseur_DTG_Mondes _Copie.mettreTout(non_diviseur _DTG_Mondes);
  //Vidange
    multiple_DTG_Mondes.vider();
    non_multiple_DTG_Mondes.vider();
    diviseur_DTG_Mondes.vider();
    non_diviseur _DTG_Mondes.vider();
  // Reconstruire les tableaux associatifs en utilisant de nouveau la méthode ajoutMonde
    Pour (chaque monde de multiple_DTG_Mondes_Copie)
        ajoutMonde (monde);
    Fin pour
    Pour (chaque monde de non_multiple _DTG_Mondes_Copie)
        ajoutMonde (monde);
    Fin pour
    Pour (chaque monde de diviseur _DTG_Mondes_Copie)
        ajoutMonde (monde);
    Fin pour
    Pour (chaque monde de non_diviseur _DTG_Mondes_Copie)
        ajoutMonde (monde);
    Fin pour
Fin fonction
```

Algorithme 9 Construction d'un chemin de monde:
Avant d'entamer la fonction *construireCheminDeMonde*, nous jugeons nécessaire de faire quelques remarques :

- Si monde j représente une zone dans monde i alors on dit que monde i est le monde parent de monde j. Ainsi les numéros des mondes associés aux numéros de leur monde parent sont mis dans un tableau associatif nommé *listDesParentsDesMonde*. Avec l'exemple précédent, *listDesParentsDesMonde* :

Numéro monde	1	2	3
Numéro monde parent		1	2

- La fonction *indexDe(mot)* appliquée à *listDesParentsDesMonde* donne la position de *mot* sur la liste.
- Chaque cellule de l'environnement connait le nom de son monde. La fonction *obtenirNomMonde* appliquée à une cellule permet d'obtenir ce nom.

```
Fonction construireCheminDeMonde(C_SoilCell cellCourante , C_SoilCell destFinal)
retourne cheminDeMonde                    // cette fonction retourne une liste de nom de monde

Variables
    cheminDeMonde : liste de chaine de caractères ;
    a, b : entiers ;
Début
    a = listDesParentsDesMonde.indexDe(cellCourante.obtenirNomMonde())
    b = listDesParentsDesMonde.indexDe(destFinal.obtenirNomMonde ())
    Si (a < b)
        Tant que (a < b)
            cheminDeMonde.ajoute(0, b) ;          // insertion de b à la position 0.
            b = listDesParentsDesMonde.obtenir(b) ;   // récupération du parent
        Fin tant que
    Sinon Si (a > b)
        Tant que (a > b)
            a = listDesParentsDesMonde.obtenir(a) ;   // récupération du parent
            cheminDeMonde.ajoute(a) ;             // insertion de a en fin de liste
        Fin tant que
    Fin si
    Renvoie cheminDeMonde;
Fin Fonction
```

Annexe C
Optimisation de l'utilisation de l'algorithme de Dijkstra

Les techniques de modélisation basées sur le déplacement d'agents dans une topologie de type graphe s'avèrent une approche fructueuse. Beaucoup de modèles liés aux déplacements d'agents utilisent l'algorithme de Dijkstra pour construire, à coup sûr, les plus courts chemins. Le modèle de monde que nous avons présenté dans la partie B, p. 41 en est un. Cependant un problème majeur de ces modèles est la fréquence à laquelle les nombreux agents, durant toute la simulation, utilisent Dijkstra pour construire leurs plus courts chemins entre les positions où ils se trouvent et les positions où ils veulent se rendre. Cette utilisation massive de l'algorithme nécessite un grand temps de calcul. Dans cette annexe, nous proposons un algorithme permettant une optimisation spatiale de la représentation informatique d'un graphe (matrice d'adjacence, liste d'adjacence), suivi d'un stockage optimisé de tout plus court chemin une fois construit. Cette optimisation évite aux agents d'avoir à reconstruire des chemins déjà construits et supprimés. Ce qui réduit considérablement le temps de calcul dû à la construction de plus court chemins.

Après un bref rappel sur les graphes, nous commençons par présenter comment nous sommes passés de l'optimisation spatiale de la représentation informatique d'un graphe (matrice d'adjacence, liste d'adjacence) pour avoir plus d'espace pour un stockage encore optimisé de tout plus court chemin une fois construit. Ensuite nous montrons comment les transporteurs utilisent les chemins stockés au lieu d'en construire de nouveaux. Après cela nous présentons quelques résultats, une discussion et terminons par une conclusion.

I. Graphe

Avant de commencer nous signalons que nous utilisons des graphes non-orientés, non-pondérés et dont, sur la représentation informatique d'un graphe, seule une information nous intéresse : la liste des nœuds adjacents (le voisinage) à un nœud donné. Ce qui est très utile pour la phase de mise à jour des labels des voisins non marqués dans l'algorithme de Dijkstra [1959].

Un graphe non-orienté G est un couple (V, E) où $V = \{v_1, v_2, ..., v_N\}$ est un ensemble fini d'objets et $E = \{e_1, e_2, ..., e_M\}$ est sous-ensemble de V x V. Les éléments de V (au nombre de N) sont appelés les sommets ou nœuds du graphe et les éléments de E (au nombre de M) sont appelés les arêtes du graphe. Une arête $\{v_i, v_j\}$ relie deux nœuds adjacents v_i et v_j ($v_i \in V$ et $v_j \in V$). On dit que l'arête $\{v_i, v_j\}$ est incidente aux nœuds v_i et v_j. Le degré D d'un nœud v_i du graphe G est le nombre d'arêtes incidentes à ce nœud. Pour faciliter la compréhension du lecteur, prenons le graphe G suivant (Figure 47) comme exemple pour le reste de l'article.

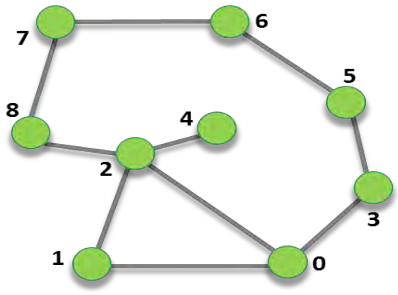

Figure 47 – Exemple de graphe pour illustration (représentation graphique).
V = {0, 1, 2, 3, 4, 5, 6, 7, 8}, E = {{0,1}, {0,2}, {0,3}, {1,2}, {2,4}, {2,8}, {3,5}, {5,6}, {6,7}, {7,8}}.

La représentation informatique d'un graphe est généralement faite par une matrice ou une liste d'adjacence.

I.A Matrice d'adjacence classique et ses inconvénients:

Une matrice d'adjacence (Tableau 12) du graphe G (V, E), non-pondéré et non-orienté, est la matrice M(G) ∈ M_N (ℝ) dont les coefficients $m_{i,j}$ sont définis par :

$$m_{i,j} \begin{cases} 1 \; si \; \{v_i, v_j\} \in E \\ 0 \; si \; \{v_i, v_j\} \notin E \end{cases}$$

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	0	1	1	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0
2	1	1	0	0	1	0	0	0	1
3	1	0	0	0	0	1	0	0	0
4	0	0	1	0	0	0	0	0	0
5	0	0	0	1	0	0	1	0	0
6	0	0	0	0	0	1	0	1	0
7	0	0	0	0	0	0	1	0	1
8	0	0	1	0	0	0	0	1	0

Tableau 12 – Matrice d'Adjacence Classique de G.

Il est vrai que l'accès à un élément est très rapide avec les matrices classiques (test d'adjacence entre deux nœuds en une complexité temporelle constante O(1)). Cependant un inconvénient avec cette représentation est la complexité temporelle de récupération de la liste des nœuds adjacents à un nœud donné (complexité linéaire O(N)). Il faut parcourir toute une ligne ou toute une colonne pour déterminer cette liste pour un nœud donné. Un autre inconvénient de cette représentation est la complexité spatiale. L'espace mémoire utilisé est de l'ordre de N^2 pour un graphe d'ordre N, même lorsque le graphe représente beaucoup moins que N^2 arêtes. Réduire ces complexités spatiale et temporelle peut conduire à l'utilisation d'une liste d'adjacence.

I.B Liste d'adjacence classique et ses inconvénients:

Une liste d'adjacence classique est un tableau de pointeurs où la liste d'adjacence relative au sommet i est la liste dans un ordre arbitraire des sommets adjacents à i. Chaque élément de la liste est constitué du numéro du sommet et d'un pointeur sur le suivant (listes chaînées). Par rapport à la matrice d'adjacence, la liste d'adjacence classique procure déjà une optimisation spatiale en ne représentant pas les non-adjacences. Le Tableau 13 est une liste d'adjacence du graphe G. L'un des inconvénients avec les listes d'adjacence est la complexité de tester l'adjacence entre deux nœuds (complexité linéaire O(D)). Par rapport à notre étude un inconvénient mineur des listes chainées est l'espace qu'occupent ses pointeurs et les procédures à mettre en place pour les manipuler.

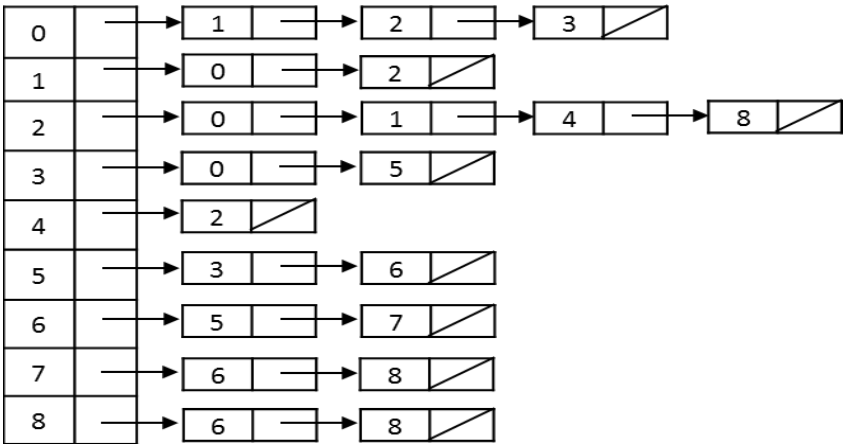

Tableau 13 – Liste d'adjacence classique de G.

II. Optimisation spatiale et temporelle de la représentation informatique d'un graphe

Notre objectif était de trouver une nouvelle structure de données plus optimisée de représentation informatique d'un graphe. Il s'agit d'une structure qui garantirait l'optimisation spatiale comme avec une liste d'adjacence classique (c'est-à-dire ne représentant pas les non-adjacences), sans l'utilisation des listes chaînées. Ce qui nous permettra de récupérer en temps constant O(1) le voisinage d'un nœud donné.

Pour cela nous avons utilisé un tableau associatif (clé-valeur). Les clés sont les numéros des nœuds du graphe et les valeurs associées aux clés sont des tableaux de taille variable (collection en Java). Le tableau associatif est caractérisé par la complexité constante O(1) de récupération de la valeur associée à une clé (get(clé)). Les tableaux simples (valeurs) sont caractérisés par la simplicité de parcourir les éléments qu'ils contiennent et d'y accéder par indice. Ainsi nous avons associé deux structures de données que nous appelons respectivement HashMap et ArrayList (comme en Java) : (*i*) HashMap est un tableau associatif qui référence ses objets (valeurs) par clé. Elle permet l'ajout (fonction put()) et la récupération (fonction get()) d'une valeur par une clé avec une complexité temporelle constante O(1). (*ii*) ArrayList

est un tableau qui permet d'ajouter (fonction add(élément)) un élément en son sein avec une complexité temporelle constante O(1).

En associant ces deux structures de données, nous avons écrit l'Algorithme 1 ci-dessous permettant une construction optimisée de la représentation informatique d'un graphe que nous appelons matrice d'adjacence optimisée (*matAdjOpt*).

Principe de Algorithme 1 : Soient nœud1 et nœud2 deux nœuds adjacents, de numéro respectif n1 et n2. Alors dans la *matAdjOpt*, n1 est une clé associée à un tableau contenant n2 et réciproquement.

Algorithme 1 ConstructionDeLaMatriceDadjacenceOptimisée()

```
Variable :
    nodeList : un tableau simple contenant par ordre tous les nœuds du graphe. La position
    d'un nœud dans ce tableau correspond à son numéro.
    matAdjOpt: HashMap <Entier, ArrayList <Nœud>>
Début
    Pour numéro allant de 0 à la taille de nodeList faire
        oneNode = nodeList[numéro]
        matAdjOpt.put(numéro, new ArrayList <Noeud>())
        Pour chaque numéro de noeud oneNeighboringNode dans le voisinage de oneNode
        faire
            matAdjOpt.get(numéro).add(oneNeighboringNode);
        Fin Pour
    Fin Pour
Fin algorithme
```

Le Tableau 14 représente la matrice d'adjacence optimisée du graphe G construit à partir de l'Algorithme 10.

0	1	2	3	4	5	6	7	8
{1, 2, 3}	{0,2}	{0,1,4,8}	{0,5}	{2}	{3,6}	{5,7}	{6,8}	{2,7}

Tableau 14 – Matrice d'adjacence optimisée de G.

Dans cette représentation optimisée, toutes les propriétés de la matrice d'adjacence classique ne sont pas conservées. Cela ne nous pose pas de problème car ce qui nous intéresse ici c'est de pouvoir récupérer la liste des nœuds voisins à un nœud donné. Ce que l'Algorithme 2 fait avec une complexité temporelle constante O(1).

Comparé à une matrice d'adjacence classique, cette manière de procéder nous permet de gagner plus de 50% en espace (64,20 % avec le petit graphe *G*). Ce pourcentage dépend de la densité des connexions entre les nœuds.

Algorithme 2 : ListDesVoisinDunNœud(noeud1 : Nœud)

Début
  renvoyer matAdjOpt.get(noeud1.numéro)
Fin algorithme

III. Stockage de la matrice des précédents

L'algorithme de Dijkstra utilise une représentation informatique d'un graphe (matrice d'adjacence optimisée) pour construire pour un sommet de départ donné, un tableau contenant le sommet précédant chaque sommet du graphe en partant du sommet de départ et en prenant le plus court chemin. Nous appelons ce tableau *tabDesPrécédents.* La matrice *matDesPrécédents* est composée des *tabDesPrécédents*. Le Tableau 15 représente le *tabDesPrécédents* pour le nœud de départ 0 avec le graphe *G*. Ce tableau nous indique par exemple qu'en quittant le nœud 0 et en prenant le plus court chemin, le nœud 7 sera précédé par le nœud 8 (voir Figure 47).

nœuds	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Nœuds précédents	0	0	0	0	2	3	5	8	2

Tableau 15 – *tabDesPrécédents* pour le nœud de départ 0 en prenant les plus courts chemins.

A partir de ce *tabDesPrécédents* (Tableau 15) il est possible de construire le plus court chemin entre 0 et tous les autres nœuds du graphe et réciproquement puisque le graphe est non-orienté. Par exemple le plus court chemin entre 0 et 7 est le plus court chemin entre 7 et 0 parcourut dans l'autre sens.

L'Algorithme 3 construit à partir de *matDesPrécédents* le plus court chemin entre deux nœuds avec une complexité temporelle O(L), L étant la longueur du chemin.

Algorithme 3 CheminFromMatriceDesPrécédents (entree: entier, sortie: entier)

Variables :
  chemin : liste d'entiers, exactement comme ArrayList en Java [ArrayList Java].
Début
  chemin.add(entree)
  chemin.add(sortie)
  tabDesPrécédents = matDesPrécédents.get(entree)
  Tant que (entrée != sortie)
    sortie = tabDesPrécédents.get(sortie)
    chemin.add(1, sortie)
  Fin tant que
Fin algorithme

Quand un agent transporteur veut se construire un plus court chemin entre un nœud de départ *noeud1* et un nœud d'arrivée *noeud2*, il utilise l'algorithme de Dijkstra pour construire d'abord le *tabDesPrécédents* pour *noeud1* pour ensuite en construire le chemin cherché. Au lieu de jeter ce *tabDesPrécédents* temporairement construit, l'idée est de le sauvegarder dans *matDesPrécédents* (Tableau 16). De ce fait quand un autre utilisateur voudra se construire un chemin dont *noeud1* est le nœud de départ ou d'arrivée, qu'il ne réinvente pas la roue en appelant à nouveau Dijkstra pour reconstruire le même *tabDesPrécédents* ou un autre. Il ne fera que construire son chemin à partir d'un *tabDesPrécédents* déjà dans *matDesPrécédents*. Pour mettre en place ce procédé il y a deux manières de faire : soit construire toute la *matDesPrécédents* juste après la construction du graphe, soit la construire au fur et à mesure que les utilisateurs appellent Dijkstra.

nœuds		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
nœud précédent	**0**	0	0	0	0	2	3	5	8	2
	1	1	1	1	0	2	3	5	8	2
	2	2	2	2	0	2	3	7	8	2
	3	3	0	0	3	2	3	5	6	2
	4	2	2	4	0	4	3	7	8	2
	5	3	0	0	5	2	5	5	6	7
	6	3	0	8	5	2	6	6	6	7
	7	2	2	8	5	2	6	7	7	7
	8	2	2	8	0	2	6	7	8	8

Tableau 16 – *matDesPrécédents* du graphe G.

III.A Construction et stockage de la matDesPrécédents au fur et à mesure que les utilisateurs appellent Dijkstra

Une des manières de faire est de ne pas construire et stocker à priori les *tabDesPrécédents* mais d'attendre qu'un utilisateur en construise un pour le sauvegarder en vue d'autres usages ultérieurs. Ainsi un *tabDesPrécédents* qui est construit l'est pour tous.

Il n'est pas possible de faire une optimisation spatiale de la *matDesPrécédents* du fait qu'elle ne contienne pas a priori toute l'information de sa partie triangulaire supérieure ou inférieure (voir le point suivant).

III.B Stockage de matDesPrécédents dès le début:

Il est possible, juste après la construction du graphe, de construire complètement la *matDesPrécédents*. La complexité temporelle est, alors, égale à N fois la complexité de l'algorithme de Dijkstra, N étant le nombre de nœuds du graphe. Ainsi Dijkstra ne sera plus utilisé pour le reste de la simulation (sauf si on met à jour le graphe). Dans ce cas il est possible de faire une optimisation spatiale de *matDesPrécédents* en ne stockant que sa partie triangulaire inférieure ou supérieure. En effet toute l'information pour construire un plus court chemin entre deux nœuds est contenue dans cette partie supérieure ou inférieure totalement construite.

III.B.1 Construction et stockage de la matDesPrécédents optimisée

Supposons que nous choisissions de stocker la partie triangulaire supérieure de la matrice. Alors cette partie ne contient donc que les nœuds précédents pour les chemins quittant un nœud de numéro plus petit vers un nœud de numéro plus grand. Nous pouvons utiliser la même structure de donnée que pour la matrice d'adjacence optimisée pour stocker les numéros de nœuds associés à des tableaux de taille variable.

Après que Dijkstra ait construit le *tabDesPrécédents* pour un nœud de départ *nœudDépart*, il faut le tronquer et ne garder que la partie dont les indices sont supérieurs au numéro de *nœudDépart*. Ceci peut être fait avec une fonction que nous appelons subList (comme en Java). Cette fonction prend deux paramètres : l'indice de début et l'indice de fin de la portion du tableau à garder comme présenté dans l'Algorithme 4.

Algorithme 4 : ConstructionDeLaMatriceDesPrécédentsOptimisée ()

```
Variables :
  nœudDépart : entier
Début
  Pour nœudDépart allant de 0 à taille de nodeList faire
    tabDesPrécédents = Dijkstra (nœudDépart)
    matDesPrécédents.put (nœudDépart,
            tabDesPrécédents.subList (nœudDépart + 1, taille de tabDesPrécédents))
  Fin Pour
Fin algorithme
```

Pour le graphe *G*, l'Algorithme 4 donne la matrice des précédents optimisée suivante (Tableau 17).

	nœuds	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
nœud précédent	**0**	0	0	0	2	3	5	8	2
	1	1	0	2	3	5	8	2	
	2	0	2	3	7	8	2		
	3	2	3	5	6	2			
	4	3	7	8	2				
	5	5	6	7					
	6	6	7						
	7	7							

Tableau 17 – *MatDesPrécédents* optimisée du oraphe G.

Avec cette matrice, pour un nœud de départ de numéro *nDépart*, le nœud précédant un nœud d'arrivé de numéro *nArrivé* (*nDépart* inférieur à *nArrivé*), est directement retrouvable

par l'instruction *matDesPrécédents.get(nDépart).get(nArrivé - nDépart)*. Exemple : si *nDépart = 3* et *nArrivé = 8* alors

matDesPrécédents.get(3).get(8-3) =
matDesPrécédents.get(3).get(5) = 2.

L'espace occupé par la matrice *matDesPrécédents* optimisée est de l'ordre de la moitié de l'espace occupé par la matrice d'adjacence classique, soit N(N-1)/2 avec N le nombre de nœuds du graphe.

III.B.2 Construction d'un plus court chemin à partir de la matDesPrécédents optimisée

Pour mieux expliquer le principe, essayons de retrouver par exemple le plus court chemin (*chemin*) entre les nœuds 3 et 8. Au début, *chemin* ne contient que 3 et 8 (*chemin* = {3, 8}). Il reste d'insérer, dans *chemin* et aux bons endroits, les nœuds intermédiaires. Puisque 3 est inférieur à 8, il est possible de récupérer le nœud précédant le nœud 8 par l'instruction *matDesPrécédents*.get(3).get(8-3), ce qui donne 2. *Chemin* devient {3, 2, 8}. Il faut à présent chercher le chemin entre 3 et 2 ou bien entre 2 et 3 (puisque 2 < 3) ; Le précédent de 3 en quittant 2 vaut *matDesPrécédents*.get(2).get(3-2) c'est-à-dire 0. *Chemin* devient alors {3, 0, 2, 8}. Le précédent de 2 en quittant 0 est 0 lui-même donc le chemin est complet. C'est avec ce principe que nous avons établi l'Algorithme 5 avec une complexité temporelle du même ordre que l'Algorithme 3 c'est-à-dire O(L) avec L la longueur du chemin.

Algorithme 5 CheminFromMatriceDesPrécédents (entree: entier, sortie: entier)

```
Variables :
    chemin : liste d'entiers, exactement comme ArrayList en Java [ArrayList Java].
    i : entier, indice d'insertion dans la liste chemin
    ie : entier, indice du nœud d'entrée.
    is : entier, indice du nœud de sortie.
    direction : entier, permet l'insertion à droite ou à gauche de sortie,
        (-1 : insertion à droite; +1 insertion à gauche).
Début
    chemin.add(entree) ; chemin.add(sortie) ;
    ie = 0 ; is = 1 ; i = is ; direction = +1 ; //insertion à gauche
    Tant que (True)
        sortie = matDesPrécédents.get(entree).get(sortie - entree)
        si entree == sortie alors
            arréter ;
        Fin si
        chemin.add(i, sortie) ; //insertion
        is = i ;
        si entree > sortie alors //permuter et changer de sens d'insertion
            entreeTmp = entree ; entree = sortie ; sortie = entreeTmp ;
            iTmp = ie ; ie = is ; is = iTmp ;
```

```
        direction = direction * (-1) ;
      Fin si
      Si (direction == -1) alors //insertion à droite
        i = is + 1 ; ie = i + 1;
      Fin si
    Fin Tant que
Fin algorithme
```

IV. Stockage des plus courts chemins

Puisque le graphe est non orienté, le plus court chemin entre A et B est le même que celui entre B et A, seul le sens de parcours les différencie. Il est donc possible de stocker les plus courts chemins une fois construits en faisant une optimisation spatiale. C'est-à-dire ne stocker que les plus courts chemins allant des nœuds de numéros inférieurs vers les nœuds de numéros supérieurs. Par exemple, avec le graphe *G* nous pouvons stocker les chemins entre 0 et tous les autres nœuds ; entre 1 et tous les autres nœuds sauf 0 ainsi de suite (comme dans Tableau 18). Ce qui donne au maximum, un nombre de chemins de l'ordre de N(N-1)/2 avec N le nombre de nœud du graphe. Ce stockage est fait dans une matrice appelé *matDesPlusCourtsChemins*.

Comme avec la *matDesPrécédents*, il est possible de construire toute la *matDesPlusCourtsChemins* justes après la construction du graphe ou au fur et à mesure que les agents transporteurs construisent les chemins.

0								**1**							**2**					
1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***3***	***4***	***5***	***6***	***7***	***8***
01	02	03	024	035	0356	0287	028	12	103	124	1035	10356	1287	128	203	24	2035	2876	287	28

Tableau 18 – *MatDesPlusCourtsChemins* entre les nœuds de départ 0, 1, 2 et tous les nœuds d'arrivée du graphe G.

V. Récupération d'un plus court chemin à partir de matDesPlusCourtsChemins

Lorsqu'un agent transporteur veut se construire un plus court chemin entre un nœud de départ *entree* et un nœud d'arrivée *sortie*, il utilise la fonction *FaireChemin* de l'Algorithme 6. Cette fonction prend en paramètre le nœud le plus petit suivi du plus grand (en comparant *entree* et *sortie*, l'agent saura dans quel sens il parcourra son chemin). La fonction *FaireChemin* cherche en premier lieu si le chemin est déjà construit pour le renvoyer directement ; Sinon elle cherche si le *tabDesPrécédents* pour le nœud *entree* (premier

paramètre) est déjà construit pour ensuite en élaborer le chemin, le sauvegarder dans *matDesPlusCourtsChemins* et le renvoyer ; Sinon elle appelle la fonction Dijkstra pour le nœud *entree*, renseigne *matDesPrécédents* et fait un appelle récursif de la fonction *FaireChemin*.

Algorithme 6 FaireChemin (entree: entier, sortie: entier)

```
Variables :
    chemin : liste d'entiers, exactement comme ArrayList en Java [ArrayList Java].
Début
    Si matDesPlusCourtsChemins.get(entree).get(sortie) existe alors
        renvoie matDesPlusCourtsChemins.get(entree).get(sortie)
    Sinon si matDesPrécédents.get(entree) existe alors
        chemin = CheminFromMatDesPrécédents(entree, sortie)
        Si matDesPlusCourtsChemins.get(entree) n'existe pas alors
            matDesPlusCourtsChemins.put(entree,
                HashMap<entier, ArrayList< entier >)
        Fin si
        matDesPlusCourtsChemins.get(entree).put(sortie, chemin)
        renvoie chemin
    Sinon
        tabDesPrécédents = Dijkstra(entree)
        matDesPrécédents.put(entree, tabDesPrécédents)
        FaireChemin (entree, sortie)
    Fin si
Fin algorithme
```

En plus du stockage de la matrice d'adjacence, nous avons ajouté deux autres niveaux de stockage permettant de gagner en temps processeur que sont la matrice des précédents et la matrice des plus courts chemins.

VI. Résultats

Nous avons utilisé cette optimisation dans deux simulateurs contenant des graphes de type voies de transport (Tableau 19). Les 5 premiers graphes (numérotés de 1 à 5) sont pour le premier simulateur. Le graphe 6 beaucoup plus grand est pour le deuxième simulateur. Les résultats du Tableau 19 montrent que, pour ces 6 cas de graphe, de 87,63 à 99,96 % de l'espace qu'occupait la matrice d'adjacence classique était inutile. Seul moins de 13 % de l'information est utile et est stockée dans la matrice d'adjacence optimisée ; Avec la complexité de récupération du voisinage d'un nœud (Algorithme 3) qui passe de O(N) avec la matrice d'adjacence classique à O(1) avec la matrice d'adjacence optimisée.

L'espace que nous avons libéré avec l'optimisation spatiale est pour le stockage de la matrice des précédents et de la matrice des plus courts chemins. Pour mieux étudier cela nous pouvons nous limiter au deuxième simulateur lancé avec 400 agents transporteurs utilisant le graphe 6. Parmi les deux manières de stocker la *matDesPrécédents* que nous avons proposé, nous avons adapté la 1ère. C'est-à-dire le stockage des tableaux des précédents et des chemins au fur et à mesure que les agents en construisent. Ainsi, autant la simulation progresse, autant la matrice des précédents et la matrice des plus courts chemins se remplissent (voir Tableau 20). Les appels de la fonction Dijkstra s'en trouvent considérablement réduits.

	graphe 1	**graphe 2**	**graphe 3**	**graphe 4**	**graphe 5**	**Graphe 6**
Nombre nœud	112	67	40	151	685	4677
EMAC	12544	4489	1600	22801	469225	21874329
EMAO	548	362	198	686	3920	9352
E libéré	11996	4127	1402	22115	465305	21864977
E. libéré (%)	95,63	91,94	87,63	96,99	99,16	99,96

Tableau 19 – Résultat de l'optimisation spatiale de la matrice d'adjacence.
EMAC : Espace occupé par la matrice d'adjacence classique ; EMAO : Espace occupé par la matrice d'adjacence optimisée ; E libéré : espace libéré quand on utilise MAO au lieu de MAC ; E libéré (%) : pourcentage de l'espace libéré par rapport à la MAC ; l'unité utilisée est le nombre d'élément de type entier stocké.

	10 000 ticks	**20 000 ticks**	**30 000 ticks**
matDesPrécédents	252612	266 646	266 646
matDesPlusCourtsChemins	25156	28229	30254
total	277768	294875	296900

Tableau 20 – Espace occupé par *matDesPrécédents* et *matDesPlusCourtsChemins* pour le graphe 6.
Un tick est un pas de temps de simulation. L'unité spatiale est l'espace mémoire que peut occuper un entier "Integer".

A 30 000 ticks (pas de temps de simulation, 1 tick = 1 jour), l'espace total occupé par la *matAdjOpt*, la *matDesPrécédents* et la *matDesPlusCourtsChemins* vaut 301 576. Ce qui est toujours inférieur à l'espace qu'aurait occupé la matrice d'adjacence classique pour le graphe 6 c'est-à-dire 21 874 329 (Tableau 19). Le nombre de chemin n'a pas explosé du fait qu'un chemin qui n'est jamais utilisé n'est jamais construit.

Pour estimer la rapidité gagnée, nous avons lancé le simulateur 2 pendant 30 000 ticks. Nous avons mesuré pour chaque tick et pour chaque construction d'un plus court chemin par un agent transporteur, la durée d'exécution dans chaque cas :

- cas où c'est Dijkstra qui est utilisé ;
- cas où c'est *matDesPrécédents* qui est utilisé ;
- cas de la récupération directe d'un chemin déjà construit.

Ces mesures sont réalisées avec la même machine et dans les mêmes conditions. Les résultats sont représentés dans le Tableau 21 ci-dessous.

	Dijkstra	matDesPrécédents	chemins prêts
Nombre d'utilisations	54	20399	1277
Durée moyenne (en ms)	2,40	0,01	0,00018

Tableau 21 – Nombre d'utilisation et durées moyennes dans chaque cas après 30 000 ticks

Le Tableau 21 montre qu'avec 400 agents transporteurs et pendant 30 000 ticks, Dijkstra n'est utilisé que 54 fois au lieu de 21 676 fois (20 399 + 1 277) pour construire *matDesPrécédents* à la demande des transporteurs. En d'autres termes 21 676 plus courts chemins sont construits ou utilisés en n'appelant que 54 fois Dijkstra. Ce tableau montre aussi que construire un plus court chemin à partir de *matDesPrécédents* (O(L)) est en moyenne 240 (2,40 / 0,01) fois plus rapide qu'en utilisant la fonction Dijkstra. Récupérer un chemin à partir de *matDesPlusCourtsChemins* est en moyenne 55,56 (0,01 / 0,00018) fois plus rapide qu'à partir de *matDesPrécédents*.

Un nœud de départ associé à un *tabDesPrécédents* (de taille N) peut être utilisé N fois pour construire N plus courts chemins différents (avec N le nombre de nœuds du graphe). Tandis qu'un chemin stocké est simplement pour un nœud de départ et un nœud d'arrivée. Ce qui explique que, jusqu'à 30 000 ticks, *matDesPrécédents* est plus sollicitée que *matDesPlusCourtsChemins*.

VII. Discussion

L'optimisation temporelle que nous venons de présenter ne prend pas effet tout au début de la simulation ou de l'utilisation d'un graphe, ou tout juste après la mise à jour d'un graphe. Il faut donc s'attendre à une certaine lourdeur (dépendant de la taille du graphe, du nombre d'agents et de la qualité de l'implémentation de l'algorithme de Dijkstra) pour les premiers appels de la fonction *FaireChemin* de l'Algorithme 6. Cette lourdeur serait maintenue durant toute la simulation si cette optimisation n'était pas réalisée.

L'algorithme de recherche de chemin dans un graphe A* [Hart *et al.*, 1968] qui est généralement plus rapide que Dijkstra [1959] et qui donne souvent de meilleurs chemins pouvait être utilisé, mais nous avons préféré utiliser Dijkstra qui est suffisamment rapide et qui donne exactement le plus court chemin dans toutes les situations. En plus de cela, Construire un plus court chemin à partir de *matDesPrécédents* est plus rapide et plus sûr qu'à partir de l'algorithme A*. Il n'est pas souhaitable d'utiliser cette optimisation avec A*. En effet l'algorithme A* n'explore pas tous les nœuds et donc il peut arriver qu'il fournisse un chemin qui n'est pas le meilleur, qui serait stocké et réutilisé par d'autres agents pendant toute une simulation.

Il faut aussi noter que si l'optimisation temporelle est suffisante avec le stockage de *matDesPrécédents*, alors on peut se passer du stockage des plus courts chemins.

VIII. Conclusion

Dans cette étude, nous avons montré comment nous pouvons passer de l'optimisation spatiale de la représentation informatique d'un graphe pour stockage optimisé des tableaux des précédents et les plus courts chemins construits par l'algorithme de Dijkstra. Ce qui permet d'éviter aux agents transporteurs de se reconstruire des chemins déjà construits auparavant et donc diminue considérablement le temps de calcul dû à la construction de plus court chemin. Ainsi nous sommes arrivés à une optimisation spatiale et temporelle de l'utilisation de l'algorithme de Dijkstra dans un simulateur multi-agents spatialisé.

Annexe D
Listes des publications scientifiques

Mboup, P.A., Konaté, K., et Le Fur, J. (June, 2017) A multi-world agent-based model working at several spatial and temporal scales for simulating complex geographic systems. In International Conference on Computational Science (ICCS), 2017 on (ETH Zürich, Switzerland :). (Accepted).

All accepted papers (in ICCS 2017) will be included in the open series and indexed by Scopus, ScienceDirect, and Thomson Reuters Conference Proceedings Citation (former ISI Proceedings) – an integrated index within Web of Science.

Mboup, P.A., Mboup, M.L., Konaté, K., Handschumacher, P., et Le Fur, J. (2015). Optimisation de l'utilisation de l'algorithme de Dijkstra pour un simulateur multi-agents spatialisé. In Information Technology and Computer Applications Congress (WCITCA), 2015 World Congress on, (Hammamet, Tunisia: IEEE), pp. 1–6.

Mboup, P.A., Handschumacher, P., Konaté, K., et Le Fur, J. (2015). Des Connaissances à la Simulation Multi-agents: Modélisation orientée événements de la Colonisation du Rat noir au Sénégal par les Transports Humains sur un Siècle. In Colloque National sur la Recherche en Informatique et ses Applications (CNRIA), (Thiès, Sénégal).

Diakhate, E.H.M., Diouf, N., Granjon, L., Konate, K., **Mboup, P.A.**, et Le Fur, J. (2014). Modélisation et simulation multi-agents d'un protocole de capture-marquage-recapture pour l'étude de la dynamique d'une population de rongeurs dans la réserve de Bandia (Sénégal). In Actes du CARI 2014, (Saint-Louis, Sénégal), p. 43-54.

Le Fur, J., Duplantier, J.M., Granjon, L., Handschumacher, P., Lombard, J., Luccacioni, H., **Mboup, P.A.** and O.Ninot (2016) Using simulation to infer key parameters of the black rat colonization in Senegal during the elapsed century. Internat. Conf. on Ecol. Sci., Marseille, 24-28 oct., https://sfecologie2016.sciencesconf.org/111995.

Le Fur, J., **Mboup, P.A.**, Sall, M., (July, 2017) A simulation model for integrating multidisciplinary knowledge in natural sciences. Heuristic and application to wild rodent studies. In International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2017). (Soumis).

Ndong, B., Diop, O., El Hadji, A.L., Mbodj, M., Gassama, S.S., Mboup M.L., Tall, K., Diop, A.K., Diop, I., Farssi, S.M. & **Mboup, P.A.** (March, 2015). JPEG2000 compression for scintigraphic images of metastasis of the prostatic cancer. In Web Applications and Networking (WSWAN), 2015 2nd World Symposium on (pp. 1-4). IEEE.