MASTER EN SCIENCES MATHÉMATIQUES

Modélisation des écoles dans la plateforme VirtualBelgium

MARCHAL, Stéphanie

Award date:
2014

UNIVERSITE DE NAMUR

Faculté des Sciences

Modélisation des écoles dans la plateforme VirtualBelgium

Mémoire présenté pour l'obtention

du grade académique de master en Sciences Mathématiques à finalité spécialisée

Stéphanie Marchal

Juin 2014

UNIVERSITE DE NAMUR

Faculté des Sciences

Modélisation des écoles dans la plateforme VirtualBelgium

Mémoire présenté pour l'obtention

du grade académique de master en Sciences Mathématiques à finalité spécialisée

Stéphanie Marchal

Promoteur : Philippe Toint

Co-promoteur : Eric Cornelis

Juin 2014

Tout d'abord, je tiens à remercier l'ensemble des professeurs et assistants du département mathématique de l'Université de Namur, qui nous ont formés tout au long des cinq années d'études.

Je remercie tout particulièrement mon promoteur Philippe Toint, mon co-promoteur Eric Cornelis et les deux assistants Johan Barthelemy et Laurie Hollaert qui m'ont aidée à réaliser ce mémoire.

Et pour finir, je dédie un remerciement aux membres de mon entourage qui m'ont soutenue lors de ce travail.

Résumé :

Le projet VirtualBelgium est une plateforme modulaire de simulation de l'évolution d'une population, créée par le Groupe de Recherche sur les Transport (GRT-naXys). Actuellement, cette plateforme est appliquée à la Belgique, où une population synthétique d'environ 10 millions d'individus répartis en plus de 4 236 000 ménages évolue temporellement mais également spatialement dans un réseau virtuel proche de celui de notre pays. Cette simulation est un outil qui permet de mieux comprendre l'évolution de la population belge à travers différentes problématiques dont la démographie et la mobilité et ainsi mieux gérer leurs conséquences. Le développement de cette plateforme passe par la modélisation du choix des écoles par les étudiants virtuels. En effet, la plupart des étudiants possèdent une chaine d'activités à effectuer en une journée dont l'activité « École ». Cependant, ils peuvent dans l'état actuel suivre leurs cours dans n'importe quel noeud du réseau. Comme ce choix n'est pas pertinent, ce travail consiste donc à la création d'un réseau d'écoles belges virtuelles et d'une modélisation de la distribution des étudiants virtuels dans ces écoles en tenant compte des données réelles récoltées. Différents modèles d'assignation ont été implémentés et testés dans le programme de la plateforme, écrit en programmation parallèle dans le langage orienté objet C++. Le modèle retenu correspondant le mieux à la réalité est choisi sur base de la pertinence des résultats dans et entre les différentes solutions testées.

Mots clés : VirtualBelgium, Simulation, Mobilité, Chaine d'activités, École

Abstract :

The VirtualBelgium project is a modular simulation platform of the evolution of a population, created by « le Groupe de Recherche sur les Transport (GRT-naXys) » of the University of Namur. For this moment this platform is applied on Belgium. A synthetic population about 10 million individuals distributed in around 4 236 000 households evolves over time but also spatially in a virtual network close to the actual Belgian one. This simulation is a tool which allows to better understand the evolution of the Belgian population and considers various problems as the demography or the mobility and so better to manage their consequences. The development of this platform requires the modelling of the choice of schools for the virtual students. Indeed, most of students have a activity chain that they have to make for a day. In this chain there is the « School » activity. However at present, they can attend their courses in any nodes of the network. Since this choice is not relevant, this work consists in the creation of a network of virtual Belgian schools and a modeling of the distribution of the virtual students in these schools by taking into account actual collected data. Different models of allocation were implemented and tested in the program of the platform, written in parallel programming in Object-oriented language C++. After a comparison of the models, the one which has the students' distribution closest to the reality is chosen.

Keywords : VirtualBelgium, Micro-simulation, Activity chains, Mobility, School

Table des matières

Introduction

Un des sujets les plus importants de notre société concerne l'évolution de la population dans le monde. Cette dernière peut prendre différents aspects tels que la démographie, la mobilité,... Par exemple, en Belgique, la mobilité est une thématique sérieuse dans la capitale où la circulation est presque à l'arrêt aux heures de pointe.

Afin de mieux gérer cette évolution et ses conséquences, il faut d'abord se doter d'outils basés sur la situation actuelle. Une application a été créée par le Groupe de Recherche sur les Transport (GRT-naXys) mettant en collaboration différents membres du personnel académique et scientifique de l'Université de Namur tels que Philippe Toint, Eric Cornelis et Johan Barthélemey. Ce projet est nommé « VirtualBelgium : une plateforme de simulation de la population belge » et consiste en la création d'une plateforme modulaire qui peut être adaptée à l'ensemble des pays sous la contrainte que les informations nécessaires à sa réalisation soient disponibles, c'est-à-dire des données utiles à la création d'une population synthétique et d'un réseau virtuel composé de noeuds et de liens.

Ce projet permet deux évolutions différentes. La première consiste à faire évoluer les individus de la population synthétique à travers les années afin d'étudier les conséquences de la démographie mais également les changements liés aux déménagements, aux mariages, aux divorces,... La seconde donne la possibilité de faire évoluer une personne tout au long d'une journée de 24h en lui assignant un agenda à suivre, afin de comprendre ses déplacements et ainsi mieux gérer la mobilité. Grâce à ces deux aspects, cette plateforme tient compte des principales problématiques liées à l'évolution de la population.

Actuellement, cet outil, développé dans le premier chapitre de ce travail, est appliqué à notre pays. À ce stade, les noeuds de réseau n'ont pas de fonction définie. En effet, les différents lieux de la Belgique, tels que les entreprises, les restaurants, les cinémas,... ne sont pas précisés. En particulier, les écoles ne sont pas localisées et dès lors, les étudiants suivent leurs cours dans n'importe quel noeud du réseau. Cet aspect du projet doit être modifié afin d'améliorer l'évolution spatiale de la plateforme. Ce mémoire consiste donc en l'implémentation d'un réseau d'écoles dans la plateforme afin d'attribuer une école à un étudiant de la population virtuelle tout en respectant la réalité. En d'autres termes,

les nombres d'étudiants dans ces écoles virtuelles doivent correspondre aux nombres réels d'étudiants dans les écoles belges.

Pour atteindre cet objectif, il faut scinder le travail en plusieurs étapes. La première consiste à comprendre le programme lié à la plateforme que l'on appelle VirtualBelgium et plus précisément, les agendas attribués aux individus. Ces derniers sont en réalité une suite d'activités que les individus doivent accomplir tout au long de leur journée. Ces activités sont localisées aléatoirement en différents noeuds. Vu que le but de ce travail est de modifier cette méthode pour les activités liées à l'école, nous devons la comprendre dans les moindres détails. Toute cette démarche est expliquée dans le chapitre 2.

La seconde étape consiste à récolter différentes informations concernant les écoles belges. Ces données correspondent à la liste des institutions reprenant le nom de l'école, le type d'enseignement (primaire, secondaire, supérieur), le code postal, le nombre d'élèves inscrits. Au final, trois fichiers contenant ces éléments sont créés. Toutes les explications relatives à cette recherche sont détaillées dans le troisième chapitre.

Par la suite, nous entrons dans le vif du sujet en passant à la création du réseau d'écoles dans la plateforme et du modèle qui permet l'assignation d'une école à un étudiant virtuel tout en respectant les mêmes proportions que dans les écoles belges. Contrairement à la création des écoles qui se réalise avec une méthode unique, différents modèles d'assignation ont été testés afin de trouver le plus pertinent et le plus proche de la réalité. Le chapitre 4 est consacré à cette partie et présente la partie théorique des modèles utilisés et les résultats obtenus.

Toutes les modifications exécutées sur le programme de la plateforme sont reprises dans le chapitre 5 et concernent principalement les personnes ayant une connaissance du langage de programmation orientée objet C++.

Pour terminer, une conclusion reprendra les points importants du travail, les avancées et les différentes perceptives qui pourraient suivre ce développement.

Le sujet de ce mémoire a été choisi pour son caractère pratique qui permet l'usage d'outils mathématiques et de langages de programmation afin d'étudier l'évolution de la population. Le choix particulier de travailler à la construction du réseau des écoles a été réalisé suite à un intérêt prononcé pour la mobilité et en particulier pour les déplacements scolaires qui prennent une part incontestable dans cette dernière. Cette remarque est appuyée par l'analyse de Ph. Toint des déplacements scolaires récoltés avec l'enquête de mobilité MOBEL [21], effectué en 1999, présente dans la référence [35].

Chapitre 1

Présentation de VirtualBelgium

La plateforme de simulation VirtualBelgium a pour ambition de comprendre l'évolution d'une population via une simulation de différents aspects : démographie, choix résidentiel, mobilité, ... Ce projet est mené par le Groupe de Recherche sur les Transports (GRT-naXys) et constitue une partie de la thèse de J. Barthélemy[6]. Cet outil est actuellement appliqué à la Belgique.

Dans cette section, nous expliquons les différents éléments qui composent la plateforme mais également les différentes données nécessaires à son utilisation. De plus, à partir de maintenant, nous considérons seulement l'outil appliquée à la Belgique.

1.1 Génération d'une population synthétique

Avant de pouvoir simuler une quelconque évolution avec l'outil VirtualBelgium, ce dernier a besoin d'une population initiale. Malheureusement, pour des raisons de confidentialité et budgétaires, il n'est pas possible de récupérer les données personnelles attendues concernant chaque citoyen belge. Il faut donc générer une population synthétique qui doit être aussi représentative que possible de la population réelle belge.

Cette population synthétique a été générée par J. Barthélemy et Ph. Toint selon une nouvelle méthode qu'ils proposent dans l'article [7]. Celle-ci est composée de 10 262 160 individus répartis en 4 236 202 ménages dans les 589 communes belges. Les données utilisées dans le processus concernent la population belge en 2001 et proviennent des sources suivantes :

- *Directorate-general Statistics and Economic information* du gouvernement fédéral belge (2001)
- *Service public fédéral Mobilité et Transports* du gouvernement fédéral belge (2000)

- *GéDAP*[1] centre de l'université de Louvain-la-Neuve (2001)
- L'enquête de mobilité *MOBEL* [21]

Ensuite, l'évolution de la population synthétique est prise en charge par un programme implémenté en C++ dont la structure est présentée à la figure 1.1. Ce programme simule une évolution temporelle (démographie) et spatiale (mobilité) de la population synthétique, variant selon les données qui lui sont fournies et utilisant le framework Repast HPC, expliqué dans la section suivante.

Après l'exécution du programme, des outputs tels que des statistiques concernant l'évolution ou encore des fichiers contenant les déplacements des individus sont créés. Ces fichiers sont ensuite exploités par le logiciel MATSim qui permet une visualisation du trafic routier. Ce logiciel sera également présenté dans la section suivante.

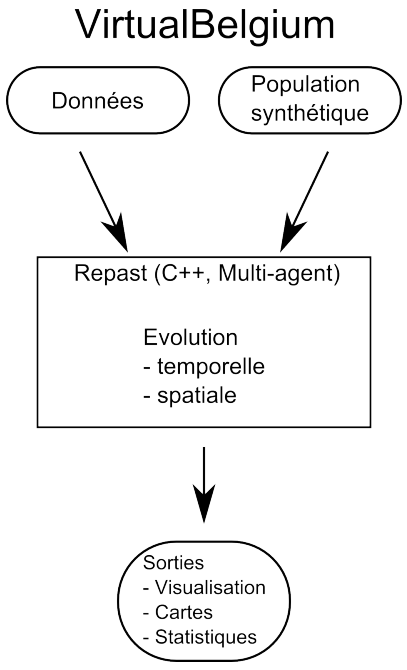


FIGURE 1.1 – Structure de VirtualBelgium

La suite de ce chapitre permet de comprendre le fonctionnement du programme VirtualBelgium lié à la plateforme. Il est donc conseillé de connaitre certaines notions des langages de programmation orientée objet. Dans le cas contraire, l'annexe E est mise à votre disposition, contenant quelques principes.

1. Groupe d'étude de démographie appliquée

1.2 Les logiciels utilisés

Comme nous venons de le dire dans la section précédente, les deux logiciels principalement utilisés par le programme VirtualBelgium sont Repast HPC et MATSim. Leur utilisation et leurs principales caractéristiques sont décrites ci-dessous.

1.2.1 Repast HPC

Repast HPC (REcursive Porous Agent Simulation Toolkit for High Performance Computing) est un framework C++ utilisé pour la simulation et la modélisation de systèmes multi-agents. D'après la référence [2], on appelle multi-agents, un système où plusieurs agents évoluent et qui possède des caractéristiques telles que le parallélisme [2], la robustesse [3] et l'extensibilité [4]. Cependant, le terme « agent » peut être interprété de plusieurs manières en informatique. Dans notre cas, un agent est un système informatique situé dans un environnement qu'il peut modifier, qui agit de manière autonome et flexible pour atteindre les objectifs qui lui sont fixés. Les agents sont capables d'interagir entre eux.

Repast est un outil gratuit, open source et une plate-forme de modélisation et simulation multi-agents. Cet outil est surtout utilisé pour des applications de sciences sociales. À l'origine, il a été développé par David Sallach ainsi que d'autres chercheurs de l'Université de Chicago et du Laboratoire National D'Argonne. Désormais, il est dirigé par l'organisation de bénévoles ROAD (Repast Organization for Architecture and Development).

Son architecture permet de représenter chaque agent par un identifiant unique et prend en charge la gestion des opérations parallèles grâce aux librairies MPI et Boost. Dans VirtualBelgium, les individus et les ménages de la population synthétique sont représentés par des agents.

Vous trouverez en annexe à la page 183, un manuel d'utilisation de VirtualBelgium. Il comprend également tout le processus d'installation de Repast HPC sur un ordinateur personnel ainsi que toutes les informations relatives à la compilation et au débuggage.

2. En informatique, le parallélisme consiste à implémenter des architectures d'électronique numérique permettant de traiter des informations de manière simultanée, ainsi que les algorithmes spécialisés pour celles-ci. Ces techniques ont pour but de réaliser le plus grand nombre d'opérations pour réduire le temps d'exécution. Cette définition provient du site de référence [41].

3. La robustesse d'un programme est entre autres, sa capacité à bien fonctionner. Cette définition est extraite du site de référence [14].

4. Un système est extensible si on peut étendre ses capacités par ajout de mises à jour ou de modules. Cette définition est extraite du site de référence [14].

1.2.2 MATSim

MATSim (Multi-Agent Transport Simulation) est un outil de simulation de transport. Il s'agit d'un projet open source principalement créé par les groupes suivants :

- *Transport Systems Planning and Transport Telematics*, Institute for Land and Sea Transport Systems, Technische Universität Berlin
- *Transport Planning*, Institute for Transport Planning and Systems, Swiss Federal Institute of Technology Zurich
- *Senozon*, entreprise suisse avec une filiale en Allemagne

Dans VirtualBelgium, MATSim est également utilisé pour créer le réseau utilisé par le programme, à partir des données extraites du site OpenStreetMap [32]. *OpenStreetMap est une carte du monde librement modifiable, collaborative où les données sont libres d'être téléchargées et utilisées, sous les termes d'une licence ouverte.* Une section sera consacrée au réseau à la fin de ce chapitre.

1.3 Coeur du programme

Regardons à présent l'architecture du programme VirtualBelgium. Celle-ci est représentée en annexe, à la figure A.1, qui décrit les différentes classes implémentées. Une classe est un principe utilisé dans la programmation orientée objet. Elle est constituée d'un ensemble de variables et de fonctions, appelées méthodes.

Sur cette figure, chaque cadre correspond à une classe du programme dont le nom est inscrit en haut de ce cadre. Ce dernier est divisé en deux parties : la première contient les variables de la classe et la deuxième les méthodes principales. De plus, des liens indiquent les relations éventuelles existantes entre les différentes classes.

Parcourons les différentes classes et résumons leurs rôles et leurs principales caractéristiques.

1.3.1 Classes générales

La classe `Data` est chargée de lire et récupérer toutes les données nécessaires au programme VirtualBelgium. Les données utilisées sont les suivantes :

- *propriétés des modèles* telles que l'évolution spatiale ou temporelle ;
- *données socio-démographiques* : pyramide des âges par commune pour les hommes et les femmes, probabilité de naissance par âge, probabilité de mort par âge et sexe,

probabilité d'avoir un garçon ou une fille ;

- *données relatives au réseau* : noeuds représentant chaque commune et réseau routier ;
- *données relatives au modèle d'activités* : codes pour les activités, paramètres de distribution concernant les activités (distance, durée du voyage, durée de l'activité et heure de départ et d'arrivée).

La classe `RandomGenerators` contient des générateurs [5] de nombres pseudo-aléatoires suivant une loi de probabilité, utilisés tout au long du programme.

La classe `Model` se charge principalement de l'initialisation du modèle et décrit la procédure d'évolution que subit la population via la méthode `step`. La méthode `step` est utilisée pour itérer sur tous les agents du modèle afin qu'ils accomplissent les tâches demandées par le modèle, telles que vieillir, mourir, déménager, se rendre à leurs activités de la journée, se marier ou divorcer, ...

1.3.2 Individus et Ménages

VirtualBelgium comporte deux différents types d'agents : les ménages représentés par la classe `Household` et les individus par la classe `Individual`.À chaque agent sont associés plusieurs variables ainsi qu'un identifiant unique, déterminé par le logiciel Repast.

Un ménage contient les identifiants de chaque individu le composant, l'identifiant du noeud du réseau correspondant au domicile du ménage ainsi que le code INS [6] de sa commune. La classe `Household` contient également des attributs concernant le nombre d'enfants (de 0 à 5), le nombre d'adultes supplémentaires (de 0 à 2) et le type de ménage parmi :

- Homme seul sans enfant
- Femme seule sans enfant
- Homme seul avec enfant(s)
- Femme seule avec enfant(s)
- Couple sans enfant
- Couple avec enfant(s)

5. Un générateur de nombres pseudo-aléatoires est un outil permettant de ressortir une suite de nombre suivant une loi demandée à partir d'un nombre qu'on appelle graine (seed). Le générateur n'est pas valide à 100%.

6. Le code INS est un code à 5 chiffres attribué à chaque commune par l'Institut National de la Statistique. Cette définition provient de la référence [28]

Un individu est décrit par son genre (féminin ou masculin), son âge et la classe d'âge associée, sa catégorie professionnelle (actif, inactif ou étudiant), son niveau d'éducation (aucun diplôme, diplôme primaire, diplôme secondaire ou diplôme supérieur), l'identifiant du ménage dont il fait partie et la place qu'il y occupe (chef de ménage, conjoint, enfant ou adulte supplémentaire) ainsi que son obtention ou non du permis de conduire.

De plus, les individus possèdent également une chaîne d'activités, c'est-à-dire une liste d'activités qu'ils doivent effectuer pendant leur journée. Par exemple, l'agenda d'un étudiant peut se résumer à deux activités : la première étant d'aller à l'école et la deuxième, de retourner au domicile.

Les différentes activités possibles sont les suivantes :

- Déposer, reprendre quelqu'un
- Activités à la maison
- Se déplacer pour le travail
- Travail
- École
- Manger à l'extérieur
- Faire des courses
- Activités personnelles (banque, docteur)
- Rendre visite à la famille ou aux amis
- Promenade
- Loisirs
- Autre
- Aucune

Toute activité d'un individu se voit imposer un lieu et une durée. Une telle modélisation par chaines d'activités permet de simuler les déplacements des individus dans le réseau, en d'autres mots leur évolution spatiale. Les chaînes d'activités nous permettent donc d'analyser la mobilité belge en vue de mieux la comprendre.

1.3.3 Réseau

Le réseau utilisé dans VirtualBelgium est le réseau routier belge entier constitué de tous les carrefours et routes du pays et extrait du site OpenStreetMap. Le réseau est

ensuite modifié par OSMOSIS [7] pour conserver uniquement les données nécessaires au programme VirtualBelgium correspondant au réseau routier belge avec comme conséquence, par exemple, que les noeuds représentant les gares dans OpenStreetMap ne sont pas pris en considération. Ensuite, ce réseau est converti dans un type de fichier XML exploitable par VirtualBelgium et MATSim.

Dans le programme, le réseau est représenté par un *graphe*. Un graphe est un objet mathématique constitué de deux ensembles finis : le premier, noté V, contient n points appelés *noeuds* ou *sommets* labellisés de 1 à n ; le deuxième, noté E, contient des liens entre certaines paires de noeuds. Un lien entre deux noeuds est appelé *arc* et est noté (i, j) lorsqu'il relie les noeuds i et j.

Notre réseau est constitué d'environ 262.000 noeuds et 830.000 arcs. Les noeuds et arc représentent respectivement les carrefours du réseau et les routes. Les arc ne sont pas orientés. La construction de ce réseau est expliquée plus en détail dans la section 4.1 du chapitre 4.

1.4 Programmation parallèle

Vu la quantité de données traitées dans VirtualBelgium avec ses 10 262 160 individus et ses 4 326 202 ménages, ce programme ne peut s'exécuter sur une unique machine possédant un seul processeur. En effet, si on exécute le programme uniquement en se restreignant à la population de Namur sur un ordinateur personnel avec deux processeurs, il faut déjà attendre deux jours pour obtenir les résultats.

Une solution à ce problème est la programmation parallèle, traitable par le logiciel Repast et par la norme MPI. Ce type de programmation consiste à effectuer différentes tâches grâce à l'utilisation de plusieurs processeurs [8]. L'utilisation de la norme MPI (Message Passing Interface) permet d'exploiter les ordinateurs multiprocesseurs en assignant l'exécution d'une suite séquentielle d'opérations à chaque processeur. Cette assignation est réalisée en utilisant des opérateurs d'envoi et de réception de messages. Chaque processeur possède son propre espace mémoire. Si on veut une interaction entre les processeurs, il existe des messages spécifiques relatifs à la synchronisation.

Dans le programme VirtualBelgium, chaque processeur va posséder son propre ensemble de ménages et donc son propre ensemble d'individus composant ces ménages.

7. OSMOSIS est une application java pour la manipulation des données OSM, extraites d'OpenStreetMap.

8. Un processeur est la partie centrale d'un ordinateur qui effectue les opérations arithmétiques et logiques. Cette définition a été inspirée du site de FUTURA- SCIENCES [18].

Actuellement, aucune synchronisation n'est effectuée entre ces processeurs, il n'y a donc pas de lien entre les ménages de chaque processeur. Le réseau de VirtualBelgium n'est pas commun. En effet, chaque processeur possède son propre réseau, ce qui veut dire que si on modifie un noeud dans un processeur, il n'est pas modifié dans le réseau des autres processeurs.

En conclusion, il faut garder en mémoire le fait que le programme VirtualBelgium doit être adapté à la programmation parallèle et que chaque processeur possède son propre ensemble d'individus et son propre réseau.

1.5 Conclusion

En conclusion, le programme VirtualBeglium possède une multitude d'outils, autant pour comprendre l'évolution de la démographie que pour comprendre les déplacements effectués par les habitants de la Belgique.

Cette introduction au projet VirtualBelgium a été écrite en collaboration avec W. Henrotin, G. Picard et E. Ramelot.

Chapitre 2

Chaines d'activités

Afin d'analyser et donc de comprendre la mobilité belge, la plateforme VirtualBelgium possède des outils permettant l'étude de l'évolution des individus virtuels le long des routes du réseau, en d'autres mots, l'évolution spatiale des individus.

Cette évolution est réalisée en utilisant la modélisation par chaines d'activités qui tient compte du comportement des individus tout au long d'une journée de 24 heures. En effet, cette méthode impose aux individus de la population de posséder une liste d'activités qu'ils devront effectuer pendant leur journée comme une sorte d'agenda.

Ce chapitre permet de comprendre les caractéristiques de cette modélisation et le travail de J. Barthelemy et Ph. Toint sur les chaines d'activités. Dans un premier temps, nous nous attarderons sur les principes de cette modélisation. Par la suite, nous décrirons en quelques mots la méthode permettant d'attribuer une chaine d'activités à un individu avant de réaliser une description d'une activité et de ses caractéristiques, y compris les méthodes d'initialisation. Nous terminerons par une présentation de l'activité qui nous intéresse dans ce travail : « École » .

2.1 Principes de la modélisation par chaines d'activités

La modélisation par chaines d'activités est une approche pour l'analyse de la mobilité qui a fait son apparition dans les années 70 afin de répondre à une nouvelle demande à laquelle aucune méthode antérieure illustrant l'approche "Trip-based" ne pouvait y répondre convenablement. En effet, avant cette période, les modèles utilisant cette approche, tenaient compte uniquement des déplacements effectués par les individus sans mettre en évidence les objectifs de ces derniers et leurs interactions et ne pouvaient donc pas modéliser la complexité observée.

La modélisation par chaines d'activités est donc utilisée pour faire face à ces inconvénients en mettant l'accent sur l'analyse des différentes actions effectuées par les individus le long d'une journée. Cette méthode permet donc, au premier plan, la compréhension du comportement général de l'individu et au second, celle des déplacements et ainsi de la mobilité en extrayant les déplacements effectués par les individus pour atteindre les lieux de leurs activités.

Pour utiliser cette méthode sur une population synthétique, chaque individu virtuel doit posséder une liste d'activités, aussi appelée chaine d'activités, qu'il doit effectuer tout le long d'une journée de 24h. En d'autres mots, cette liste représente l'agenda de l'individu pour une journée. L'attribution d'une chaine d'activités doit dépendre des différentes caractéristiques de l'individu et doit tenir compte des données réelles. En effet, toutes les chaines d'activités proviennent de données récoltées par des enquêtes.

La modélisation par chaines d'activités doit respecter les règles citées ci-dessous, extraites du site [34] :

1. Le mot « activité » est utilisé fréquemment dans les paragraphes précédents mais n'a pas été clairement défini. Dans notre cas, « une activité est une interaction continue avec l'environnement physique, avec un service, une personne et qui a de l'importance pour l'individu »[1]. Par exemple, le travail, l'école ou les loisirs sont considérés comme des activités. On peut également considérer la promenade à vélo ou à pied comme une activité à part entière car elle consiste à apporter du plaisir à l'individu et donc, elle a de l'importance pour ce dernier. Par exemple, voyager du domicile au bureau n'est pas considéré comme une activité car le voyage en lui-même n'est pas important pour la personne mais simplement le fait d'arriver à destination.

   Une activité doit posséder différents paramètres qui la caractérisent tels que
   - le type d'activité ;
   - la durée ;
   - l'heure de départ et/ou l'heure de fin ;
   - le lieu.

- Lorsqu'on parle de déplacement avec cette approche, on considère seulement ceux effectués entre deux activités. Par exemple, déposer ses enfants à l'école est considéré comme une activité et non comme un déplacement

1. Cette définition provient de la référence [34]

- Les individus doivent effectuer leurs activités en tenant compte de différentes contraintes telles que
  - leur budget ;
  - leur niveau social ;
  - la composition de leur ménage ;
  - les ressources du ménage (par exemple : le nombre de voitures, de vélos dans le ménage) ;
  - le lieu de leur travail ou celui de leur école.

  Le lieu de travail ou celui de l'école doit être unique pour chaque individu. En effet, on suppose qu'un étudiant ne peut pas aller dans une école le matin et dans une autre l'après-midi. Les activités concernant l'école et le travail doivent donc être assignées à un lieu précis.

- L'évolution d'un individu doit se dérouler sur une journée de 24h.

Ces règles permettent une meilleure utilisation de la modélisation par chaines d'activités et une interaction entre l'individu et les co-habitants du ménage. Cependant, pour l'instant aucune étude utilisant cette approche ne respecte entièrement ces règles. Cela est dû à la difficulté de récolter toutes les données nécessaires sur la population souhaitée.

2.1.1 Avantages

Un des avantages de mettre l'accent sur les activités est l'analyse du comportement de certains types d'individus. En effet, d'après la référence [1], le travail de Ferguson and Jones sur la modélisation par chaines d'activités a permis de mettre en évidence le rythme de vie des personnes âgées ou des personnes handicapées et ainsi pouvoir mieux adapter les infrastructures mises à leur disposition.

Un autre avantage est la représentation des individus travaillant chez eux (télétravail). En effet, dans les approches concernant uniquement les déplacements, ces individus ne pouvaient être analysés à travers leurs déplacements pour leur travail.

2.1.2 Conclusion

L'analyse de la mobilité en utilisant la modélisation par chaines d'activités possède plusieurs avantages et permet autant la compréhension du comportement humain que de leurs déplacements.

2.2 Modélisation par chaines d'activités dans VirtualBelgium

Afin d'analyser la mobilité belge en utilisant la modélisation par chaines d'activités, tous les individus de la population de VirtualBelgium doivent posséder une liste d'activités, comme expliqué dans la section précédente.

Chaque activité de cette liste doit respecter les conditions établies par la modélisation. Il existe au total 12 activités possibles pour un individu. Elles sont décrites dans la table 2.1. Ces dernières ont été choisies sur base de l'enquête sur la mobilité en Belgique, nommée MOBEL et effectuée en 1999.

Numéro	Activité
1	Déposer / reprendre quelqu'un
2	Activités à la maison
3	Se déplacer pour le travail (aller voir des clients par exemple)
4	Travail
5	Ecole
6	Manger à l'extérieur
7	Faire les courses
8	Activités personnelles (banque, docteur)
9	Rendre visite à la famille / aux amis
10	Promenade
11	Loisirs
12	Autre
13	Aucune

TABLE 2.1 – Les activités dans VirtualBelgium

Dans le programme VirtualBelgium, on ne tient pas encore compte des contraintes liées aux niveaux sociaux et aux budgets, ni aux ressources du ménage car tous ces paramètres ne sont pas encore inclus dans les caractéristiques des individus et des ménages. Les points concernant les paramètres déterminant une activité et la contrainte liée aux lieux du travail et de l'école, seront abordés juste après l'explication de la méthode d'assignation d'une chaine d'activités aux individus de VirtualBelgium.

2.2.1 Méthode d'attribution d'une chaine d'activités

Chaque individu de la population synthétique possède une chaine d'activités, excepté les enfants âgés de moins de 6 ans. Ces derniers sont considérés comme inactif dans la plateforme et ne possèdent donc pas de chaine d'activités. La méthode d'assignation d'une

chaine d'activités utilisée par J.Barthelemy et Ph. Toint [5] pour le reste de la population est composée de deux parties.

Dans un premier temps, les individus de la population virtuelle, excepté les enfants âgés de moins de 6 ans, sont classés dans différentes catégories selon leurs caractéristiques. Chaque catégorie possède un ensemble de chaines d'activités potentielles déterminées en fonction des données récoltées dans l'enquête concernant la mobilité belge. Cette étape est décrite dans la référence [5]. Pour un étudiant de l'enseignement primaire, un exemple de chaine d'activités pourrait être :

1. École ;
2. Activités à la maison.

La dernière activité de chaque chaine d'activités doit reconduire l'individu chez lui. En d'autres termes, chaque chaine d'activités doit se terminer par l'activité « être à la maison ». Chaque chaine d'activités doit aussi posséder un poids, c'est-à-dire une constante permettant de connaitre leur fréquence d'apparition dans les données réelles. Grâce à ce poids, on peut calculer, dans chaque catégorie, la probabilité des chaines d'être choisie.

Dans chaque catégorie, le nombre de chaines doit être supérieur à cinq. Si cela n'est pas garanti, ce nombre doit être augmenté en ajoutant des nouvelles chaines. Les explications sur la méthode permettant de créer une nouvelle chaine d'activités peuvent se trouver dans l'article de référence [5].

Dans un deuxième temps, chaque individu se voit attribuer, de manière aléatoire, une chaine de sa catégorie tout en tenant compte des poids.

Pour illustrer cette méthode, prenons l'exemple de l'article de référence. Soit une femme, entre 18 et 39 ans, ayant un diplôme de l'enseignement supérieur mais continuant ses études et ne possédant pas de permis de conduire, il existe 6 chaines d'activités potentielles (voir la table 2.2) dont chacune d'elles possède un poids qui permet de connaitre leur probabilité d'être choisie lors du choix aléatoire.

Par exemple, une femme de la population virtuelle possédant les caractéristiques citées ci-dessus, a 8 % de chance de posséder la chaine d'activités « 2 5 10 2 », ce qui signifie le planning suivant :

1. Activités à la maison
2. École ;

3. Promenade ;

4. Activités à la maison

Chaines d'activités	Poids	Probabilités
2 5 10 2	0.272	0.08
2 2 2 9 2	1.025	0.31
2 5 2	0.913	0.28
2 5 2 10 2	0.412	0.12
2 5 6 5 2	0.412	0.12
2 11 2	0.284	0.9

TABLE 2.2 – Ensemble des chaines d'activités potentielles pour une femme âgée de 18 à 39 ans, ayant un diplôme de l'enseignement supérieur mais continuant ses études et ne possédant pas de permis de conduire. Chaque nombre représente une activité (voir la table 2.1).

2.2.2 Description d'une activité dans VirtualBelgium

Conformes aux règles de la modélisation par chaines d'activités, les activités effectuées par les individus de VirtualBelgium possèdent des paramètres [2] qui les identifient tels que :

- **le type** permettant de connaitre l'objectif de l'activité ;
- **la distance parcourue** pour atteindre cette activité par rapport à l'activité précédente ;
- **la durée du trajet ;**
- **le lieu** où se déroule l'activité ;
- **la durée de l'activité ;**
- **l'heure de fin.**

Chaque individu de VirtualBelgium possède ses propres activités afin de construire le planning de sa journée. Les différentes valeurs de ces paramètres sont déterminées au fur et à mesure de leur journée. La durée du trajet, la distance parcourue et la durée pour chaque activité doivent être déterminées en fonction des données réelles récoltées. L'heure de fin est calculée en fonction de ces trois paramètres. En ce qui concerne la localisation de l'activité, elle est choisie en fonction de la distance parcourue comme décrit ci-dessous.

2. Les détails sur ces paramètres, le nom des variables correspondantes ainsi que leurs caractéristiques peuvent être consultées dans la table B.1.

2.2.2.1 Distance parcourue, durée du trajet et durée de l'activité

1. Création de fonctions de densité

Afin de pourvoir déterminer la distance parcourue par un individu pour atteindre une de ses activités et la durée de l'activité, J. Barthelemy, Ph. Toint et E. Cornelis [5] ont calculé les fonctions de densité empiriques [3] et les fonctions de répartition empiriques [4] relatives à ces items pour chaque type d'activités sur base des informations récoltées lors de l'enquête MOBEL. Cependant, l'utilisation de ces dernières n'est pas très efficace du point de vue du temps d'exécution. J. Barthelemy et Ph. Toint ont donc modélisé chaque fonction en déterminant une loi de probabilité et ainsi obtenir une meilleure performance du temps d'exécution. Pour minimiser les erreurs dues à la modélisation, la différence entre la fonction de densité de la loi de probabilité trouvée et la fonction densité empirique doit être minimale. Après des tests, les fonctions de densité empiriques peuvent être approximées par une combinaison de fonctions de densité de différentes lois log-normales [5]. La loi de probabilité d'une telle fonction de densité est appelée mixture de loi log-normale dont la définition est citée ci-dessous.

Définition 2.1. [6] *La fonction de densité f(x) d'une mixture de loi log-normale est construite à partir de plusieurs lois log-normales indépendantes.*

Soient n *le nombre de lois log-normales qui forment la mixture*

$w_i \geq 0$ *la proportion de la* $i^{ième}$ *composante.*

$$\sum_{i=1}^{n} w_i = 1$$

$f_i(x, \mu_i, \sigma_i^2)$ *le fonction de densité de la loi log-normale de paramètre* μ_i *et* σ_i^2 *telle que* $p_i(x, \mu_i, \sigma_i^2) = \frac{1}{x\sigma_i\sqrt{2\pi}} \exp(-\frac{(\ln(x)-\mu_i)^2}{2\sigma_i^2})$

$$f(x) = \sum_{i=1}^{n} w_i * f_i(x, \mu_i, \sigma_i^2) \quad \forall \ x \in I\!R$$

Pour la distance parcourue, les fonctions de densité empiriques de chaque type d'activités se rapprochent d'une fonction de densité d'une seule loi log-normale contrairement aux fonctions de densité empiriques des durées des activités qui se modélisent par une

3. La fonction de densité empirique d'un ensemble de données est calculée à partir des différentes valeurs observées et de leur fréquence.

4. La fonction de répartition correspond aux probabilités cumulées associées à la variable aléatoire continue sur l'intervalle d'étude. Cette définition est extraite du site de référence [25]. La fonction de répartition empirique d'un ensemble de données est calculée à partir des différentes valeurs observées de ces données et de leur fréquence cumulée.

5. Une variable aléatoire X suit une loi de probabilité log-normale de paramètre μ et σ^2 si la variable aléatoire Y=Ln(X) suit la loi de probabilité normale de paramètre μ et σ^2. En d'autres mots, le logarithme népérien des valeurs de X doit suivre une loi normale de moyenne μ et de variance σ^2.

6. Cette définition a été inspirée de l'article [5]

combinaison de deux ou trois fonctions de densité d'une loi log-normale.

Pour un souci de précision, J. Barthélemy et Ph. Toint ont également modélisé les fonctions relatives à la durée de l'activité en fonction de l'heure à laquelle elle commence et ainsi pouvoir déterminer la durée de l'activité en fonction de son type et de l'heure de départ. Les graphiques représentant les fonctions de répartition liées aux fonctions de densités, réalisés par J. Barthelemy et Ph. Toint, sont mis à votre disposition dans la section B.2.

Afin de pouvoir calculer la durée du trajet pour atteindre une activité, J. Barthélemy, Ph. Toint et E. Cornelis ont créé des fonctions de densité regroupant les durées des trajets en fonction de la distance parcourue. Pour plus d'information concernant ces fonctions de densité, la référence [5] est mise à votre disposition.

2. Détermination des valeurs

Après avoir approximé les fonctions de densités empiriques par des lois de probabilité, il suffit d'utiliser un générateur de nombres pseudo-aléatoires suivant la loi demandée pour obtenir aléatoirement une distance ou une durée pour une activité tout en respectant au maximum les données réelles. Ces générateurs sont déjà construits dans la plateforme et prennent en argument les paramètres de des lois qu'ils doivent suivre. Par exemple, il existe des générateurs de loi log-normale, de mixture de lois log-normales, de loi uniforme,...La construction de ces générateurs est expliquée dans [6].

Les distances parcourues pour l'activité « École » sont approximées par une loi log-normale de paramètres μ= 7.9579 et σ = 1.3879841498. Pour connaitre la distance parcourue par un étudiant virtuel, il suffit de générer une distance avec le générateur de nombres pseudo-aléatoires suivant la loi log-normale μ= 7.9579 et σ = 1.3879841498, donnée par la classe « RandomGenerateur ».

3. Remarques

Deux remarques concernant les durées peuvent être mises en évidence :

- Comme une journée se déroule en 24h, la somme des durées de chaque activité d'un individu et des durées de chacun de ses déplacements doit être inférieure ou égale 24h.

- Avec les durées des déplacements et des activités, nous pouvons déterminer le planning de la journée d'un individu. Cependant, tant que nous ne connaissons pas

l'heure de départ du domicile pour chaque individu, nous ne pouvons pas assigner chaque activité du planning à un moment précis de la journée de l'individu et donc connaitre l'heure de début et de fin de chaque activité. J. Barthelemy et Ph. Toint ont donc suivi les mêmes étapes, expliquées précédemment, pour modéliser une fonction de densité empirique construite avec les différentes heures de départ de la journée pour chaque type d'activités. Ainsi dès que l'heure de départ est connue pour un individu, l'heure de début et de fin de chacune de ses activités sont calculées via les durées de ces dernières et des déplacements.

2.2.2.2 Localisation des activités

1. Sélection d'un nœud

Nous allons maintenant nous intéresser à la manière dont le programme lié à la plateforme sélectionne un nœud dans le réseau pour localiser l'activité d'un individu. Pour un individu, le choix du lieu, c'est-à-dire du nœud du réseau où se déroule l'activité, va dépendre de la distance parcourue par l'individu pour effectuer l'activité mais également de son lieu actuel, c'est-à-dire de sa précédente activité.

Soient un individu et une activité qu'il doit réaliser,

1. on calcule aléatoirement une distance que l'individu doit parcourir pour effectuer son activité en utilisant la méthode expliquée dans la section 2.2.2.1. Cette distance est notée d.

2. on sélectionne tous les noeuds du réseau virtuel qui sont à une distance d du lieu de l'activité précédente, à 250 mètres près. Cette valeur permet une certaine marge lors de la sélection des noeuds. Si la distance est égale à 1000m alors les noeuds sélectionnés seront à une distance entre 750 m et 1250 m du lieu précédent (voir la figure 2.1). Cette étape est réalisée grâce à l'algorithme de Dijkstra qui calcule le plus court chemin entre 1 noeud et les autres.

3. on choisit un nœud de manière uniformément aléatoire parmi ceux sélectionnés à l'étape précédente. Ce nœud sera donc désigné comme étant le lieu où l'individu effectuera son activité.

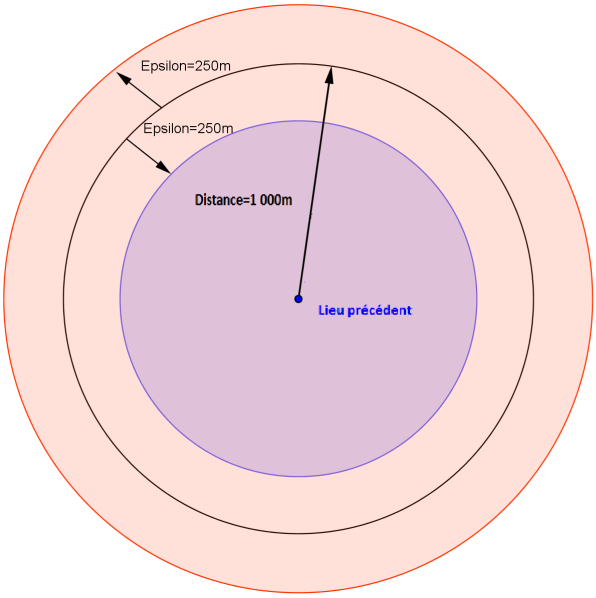


FIGURE 2.1 – Signification de la marge égale à 250m avec une distance choisie de 1 km. L'aire rouge [7]représente la zone où les noeuds sont sélectionnés.

2. Exemple

Prenons un individu, dont le ménage se situe au point rouge de l'image 2.2. Suivant son planning, l'individu doit se rendre à son travail dès le début de la journée. Pour connaitre le lieu de ce dernier, le programme VirtualBelgium va, tout d'abord, calculer aléatoirement une distance en fonction de la fonction de répartition relative au travail. Cette distance vaut, pour cet exemple, 500 m.

Ensuite le programme sélectionne les nœuds du réseau virtuel qui se trouvent à une distance entre 250 m et 750 m du domicile de l'individu. On distingue onze nœuds.

Pour terminer, il ne reste plus qu'à choisir un de ces onze nœuds pour y assigner l'activité de cet individu. Ce choix est fait de manière uniformément aléatoire, c'est-à-dire que tous les noeuds ont la même probabilité d'être choisi. Au final, l'individu travaillera au nœud n°5 (voir l'image 2.3)

7. Dans ce graphe, la distance et l'epsilon sont illustrés en vol d'oiseau. Or il faut garder en mémoire que les distances en VirtualBeglium sont calculées à partir de la longueur des routes. Ce graphe est juste ajouté à titre informatif

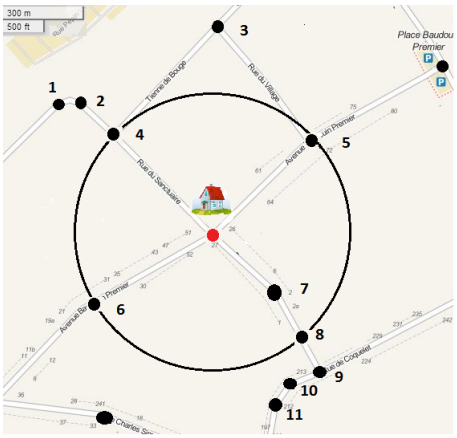


FIGURE 2.2 – Exemple pour la localisation de l'activité (1). Cette image a été inspirée du site d'OpenStreetMap [32]

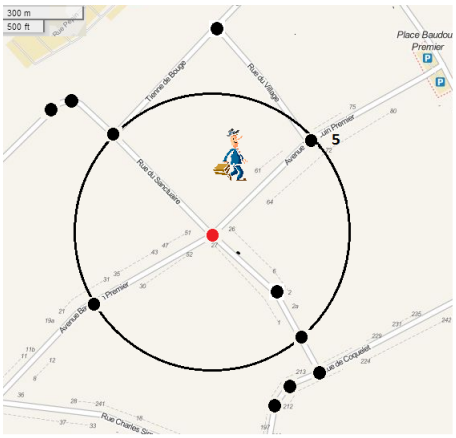


FIGURE 2.3 – Exemple pour la localisation de l'activité (2). Cette image a été inspirée du site d'OpenStreetMap [32]

3. Remarque

Pour l'instant, dans le programme VirtualBelgium, il n'y a aucune contrainte concernant les lieux où se déroulent les activités. Or pour obtenir une bonne analyse de la mobilité en utilisant la modélisation par chaines d'activités, il est précisé que les activités « Travail » et « École » doivent être assignées à un lieu précis pour chaque individu.

2.2.3 Simulation

Afin de simuler les déplacements des individus grâce au logiciel MATSIM, présenté dans le chapitre 1, le planning de chaque individu de la population virtuelle est inscrit dans un fichier XML lisible par le logiciel. Les éléments essentiels à cette simulation sont le type, le lieu et l'heure de fin de chaque activité. Le fichier contient également l'heure

de départ du domicile pour chaque individu.

Pour chaque déplacement d'un individu, MATSIM calcule un itinéraire qui correspond au plus court chemin entre le nœud de départ et celui d'arrivée Le logiciel essaye de respecter les horaires des activités données. Cependant, un imprévu peut avoir lieu, comme par exemple un bouchon, ce qui entraine une perte de temps et un retard dans le planning. Pour contrer ce problème, MATSIM peut décider de modifier l'itinéraire afin de minimiser le temps de déplacement.

Le logiciel MATSIM essaye de respecter au mieux les horaires donnés mais il se peut que cela soit impossible. Dans ces conditions, MATSIM modifie les horaires des individus tout en optimisant la durée des activités.

2.3 Activité liée à l'école

La méthode de localisation d'une activité, présentée précédemment, est également utilisée pour l'attribution d'un établissement scolaire à un élève ou à un étudiant. Ce qui veut dire qu'un individu allant à l'école, peut se retrouver dans n'importe quel endroit du réseau. En d'autres mots, cela reviendra à dire que les étudiants en Belgique peuvent aller suivre leurs cours dans n'importe quel bâtiment. La modélisation de l'activité « École » n'est pas représentative de la réalité.

Ce travail a pour but d'améliorer la localisation de l'activité liée à l'école en réalisant un réseau d'écoles dans la plateforme et en attribuant une école à un enfant de la population virtuelle tout en respectant les données réelles.

Avant de créer des modèles tenant compte de la remarque précédente, il faut tout d'abord récolter des données. Ce travail a donc commencé par une recherche de données concernant les établissements scolaires belges. Les différentes démarches et les résultats obtenus sont expliqués dans le prochain chapitre.

Chapitre 3

Récolte des données

Comme présenté dans le chapitre précédent, le travail relatif au mémoire commence par une recherche de données concernant les établissements scolaires en Belgique. Les informations obtenues doivent aider à créer des établissements scolaires dans le réseau virtuel et à améliorer la distribution des étudiants dans ces écoles. Ce chapitre contient d'une part, une explication permettant de comprendre la nature des données nécessaires à ce travail, d'autre part, les différentes démarches effectuées pour l'obtention de ces renseignements.

3.1 Données

Le nom de l'école, sa localité, son code postal et le nombre d'élèves qui y sont inscrits, forment l'ensemble des renseignements qui doivent être récoltés pour chaque établissement scolaire en Belgique. Par exemple, l'école Gemeentelijke Largere School est un établissement scolaire situé dans la commune de Bruxelles dont le code postal est 1000. 3673 élèves viennent y suivre des cours chaque jour.

Pour pouvoir géocoder [1] de manière précise, une école dans le réseau routier de la simulation, son adresse exacte (rue et numéro) est nécessaire. Cependant, cette information ne sera pas intégrée dans ce mémoire en raison de la gestion du temps. En effet, il existe plus de 7000 écoles belges et pour les géocoder, il faudrait pour chacune, connaitre leurs coordonnées géographiques et trouver le noeud du réseau virtuel correspondant le mieux à ces dernières. Néanmoins, pour préparer l'évolution du sujet dans un futur travail, l'adresse sera récoltée à titre informatif dans la cas où il y a moyen de l'obtenir.

L'emplacement des écoles dans la virtualisation va donc dépendre uniquement du code postal. Mais comme le plus petit niveau de désagrégation dans le programme Vir-

1. Géocoder une école signifie la localiser en fonction de son adresse dans un réseau virtuel proche de celui de la Belgique.

tualBelgium, est la commune, les institutions seront donc placées dans le réseau virtuel en fonction de leur commune et plus précisément en fonction du code INS de cette dernière. Dès lors, il n'y aura pas de différences entre une école de la commune de Namur donc le code postal est 5000 et une école de la commune de Namur mais avec le code postal 5001.

Dans la suite du travail, les codes postaux des écoles belges seront utilisés pour créer des écoles virtuelles dans les 589 communes du réseau virtuel. Le nombre d'élèves inscrits permettra une meilleure distribution des étudiants virtuels de VirtualBelgium dans les écoles. Les méthodes utilisées pour réaliser ces objectifs sont expliquées dans le chapitre 4.

L'ensemble des données portent sur les deux types d'enseignement existant : obligatoire ou non obligatoire. D'une part, l'enseignement obligatoire se compose de l'enseignement fondamental qui est lui-même constitué des enseignements préscolaire et primaire, et de l'enseignement secondaire. L'obligation scolaire s'applique seulement aux enfants âgés de 6 à 17 ans (compris). D'autre part, l'enseignement non obligatoire, communément appelé l'enseignement supérieur, regroupe les Universités, les Hautes Écoles, les Écoles supérieures des Arts et d'architecture. Il n'est accessible qu'aux personnes ayant obtenu leur certificat d'enseignement secondaire supérieur (CESS).

Cependant, lors du déroulement de ce travail, seules les données concernant l'enseignement primaire et secondaire pour l'enseignement obligatoire et celles concernant l'enseignement supérieur, sont collectées. En effet, les enfants en âge d'aller à l'école préscolaire, sont considérés comme inactifs dans la conception de la simulation de la population dans VirtualBelgium.

3.2 Démarches et résultats

Étant donnée la configuration de la Belgique en trois communautés (voir figure 3.1) ayant leurs propres Ministres de l'enseignement, la recherche des informations spécifiques fût divisée en trois parties : la première concernant la communauté française, la seconde correspondant à la communauté flamande et la dernière, à la communauté germanophone. En ce qui concerne les écoles de la région Bruxelles-Capitale, elles sont soit prises en charge par la communauté française, soit par la communauté flamande.

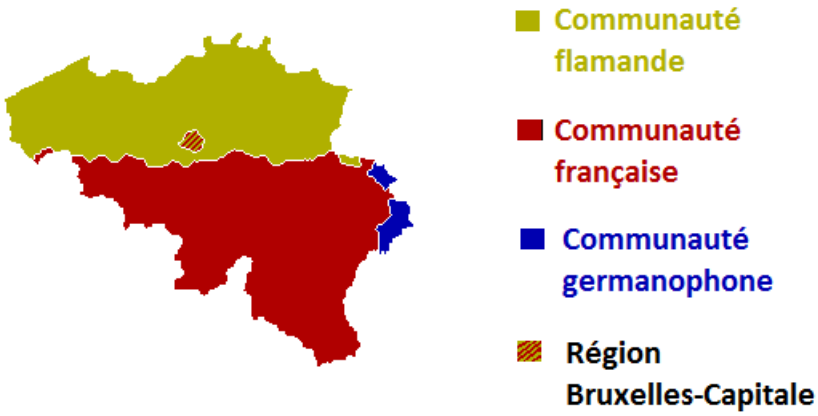


FIGURE 3.1 – Répartition des communautés en Belgique. Cette image est inspirée du site de référence [41]

3.2.1 Communauté française

Dans cette sous-section, nous énumérons toutes les données récoltées pour cette communauté en séparant les deux types d'enseignement.

Enseignement obligatoire

Dans un premier temps, une recherche sur le moteur de recherche Google a pu mettre en évidence :

- trois listes concernant les écoles sous la direction de la communauté française pour l'année académique 2012-2013, téléchargeables sur le site de référence [17] :
  - une liste des établissements primaires (ordinaires et spécialisés),
  - une liste des établissements et des implantations secondaires pour la première année d'étude,
  - une liste des établissements secondaires qui sont reliés à un ou plusieurs établissements primaires en Wallonie et à Bruxelles.

  Les tables D.1, D.2, D.3 représentant les premières lignes de chaque liste sont mises à votre disposition dans l'annexe D.1.1.

- un tableau contenant le nombre d'élèves fréquentant les établissements préscolaires, primaires et secondaires ainsi que les établissements supérieurs hors universités par arrondissement. Cette information provient du site de référence [16] de l'Entreprise publique de Technologies nouvelles de l'Information et de la Communication (ET-

NIC [2]).

Ces données sont intéressantes pour connaitre les noms et les adresses des établissements primaires et secondaires pris en charge par la communauté française et pouvoir également les classer par commune mais elles ne fournissent pas l'information concernant le nombre d'étudiants inscrits. Ces renseignements, par année académique, ont été obtenus suite à une demande officielle transmise à l'intention de Madame Marie-Dominique Simonet, Ministre en charge de l'enseignement obligatoire de Fédération Wallonie-Bruxelles par l'intermédiaire de Madame Christelle Ladavid, attachée aux affaires générales et intergouvernementales de la direction générale de l'enseignement obligatoire. En effet, les données souhaitées ne peuvent être transmises en dehors des services du Gouvernement et du ministre compétent et d'après Madame Ladavid, la communication de ces informations doit tenir compte de l'article 8, du paragraphe 4 du « Décret portant diverses mesures en matière de culture, de santé, d'enseignement et de budget du 27 décembre 1993 ». Ce dernier impose que toute demande transmise au ministre concernant chaque établissement scolaire doit être introduite par des personnes précises dont des chercheurs qualifiés (paragraphe 3 du décret). Cette demande a donc été appuyée par le Professeur Toint, vice-recteur à la recherche. Toutes les informations concernant cette demande et le décret se trouvent dans l'annexe D.4. Il n'existe pas d'annexe contenant ces données pour des raisons de confidentialité.

Les données récoltées suite à cette demande permettent de connaitre le nombre d'étudiants inscrits dans chaque école de la Fédération Wallonie-Bruxelle pour l'année académique 2011-2012. Cependant, les écoles sont uniquement distinguables par leur numéro de FASE [3]. Nous ne pouvons pas connaitre le nom de l'école ou son adresse. Grâce aux listes récupérées précédemment, nous pouvons assigner une adresse à certaines écoles. Cependant, nous ne pouvons retrouver l'adresse de toutes les écoles composant ces données.

Enseignement non obligatoire

Pour l'enseignement supérieur, seulles des informations concernant les universités francophones (voir table 3.1) ont été trouvées via le site [12] du conseil des recteurs des universités francophones de Belgique regroupant différentes statistiques sur les étudiants, à partir de l'année 1996. Les données concernant l'année académique 2010-2011 sont mises en évidence dans ce travail puisqu'elles sont les plus récentes.

2. D'après le site de référence [16], « l'ETNIC est l'entité autonome et spécialisée en matière informatique de la Fédération Wallonie Bruxelles ». ETNIC s'occupe également des statistiques de l'enseignement en Fédération Wallonie-Bruxelles.

3. Le numéro de FASE est une identification administrative de chaque établissement scolaire. Cette définition provient du site de référence [39]

Université de Liège	ULg
Université catholique de Louvain	UCL
Université libre de Bruxelles	ULB
Université de Mons	UMONS
Facultés universitaires Notre-Dame de la Paix à Namur	FUNDP
Facultés universitaires Saint-Louis à Bruxelles	FUSL
Facultés universitaires catholiques de Mons	FUCaM

TABLE 3.1 – Liste des universités francophones.

Plus précisément, le site contient un tableau [4] contenant le nombre d'étudiants effectuant des études dans les universités francophones en Belgique. Dans les données reçues, une distinction est faite au niveau de la nationalité. Comme cette étude est concentrée sur la population belge, seules les statistiques concernant les étudiants belges sont utilisées.

En ce qui concerne les Hautes Écoles francophones, plusieurs démarches ont été effectuées mais aucune donnée n'a pu être transmisse. Nous obtenons juste le tableau, cité précédemment, contenant le nombre d'étudiants par arrondissement.

3.2.2 Communauté flamande

Comme pour les données au sein de la communauté française, les recherches concernant les établissements scolaires flamands se sont déroulées en deux phases.

Enseignement obligatoire

Les premiers éléments pertinents trouvés sont

- des tableaux concernant le nombre d'élèves dans les établissements primaires et secondaires par rapport aux arrondissements et aux provinces sur le site des statistiques de l'éducation de la Flandre [30].

- des listes des établissements primaires et secondaires se trouvant en Flandre via le site de référence [29]. Ces dernières contiennent les informations sur le numéro de l'établissement, le nom, le siège central de l'école et l'adresse. Elles peuvent également fournir le numéro de téléphone et de fax de l'établissement, une adresse email ainsi que le site Web officiel de l'école. Les premières lignes de ces listes sont représentées dans la table D.5 de l'annexe D.2.1 pour l'enseignement primaire et dans la

4. Pour plus de compréhension et pour plus de détails sur la manière de lire le tableau, l'annexe D.1.2 est mise à votre disposition.

table D.9 de l'annexe D.2.2 pour le secondaire.

Comme dans la première partie, ces données ne suffisent pas à notre étude. Après d'autres recherches, une liste contenant le nombre d'élèves dans les établissements secondaires ordinaires pour l'année 2011-2012 est trouvée via le site des statistiques de l'éducation.

Une deuxième liste concernant les chiffres de l'année 2012-2013 relatifs aux établissements primaires, est mise en évidence sur le site de référence [3]. Cette dernière contient également le nombre d'élèves par établissement préscolaire. Les premières lignes de ces listes se trouvent dans les tables D.6, D.7, D.10 de l'annexe D.2.1. Nous pouvons remarquer que le nombre d'étudiants peut parfois être un nombre décimal. En effet, certaines écoles sont affiliées à des centres de soins pour les écoliers. Le nombre d'étudiants dans ces écoles est une moyenne par mois et non par année vu que les écoliers n'y restent pas un temps prédéfini.

Cependant ces deux listes ne tiennent pas compte des écoles spécialisées flamandes. Pour obtenir les informations nécessaires sur ces dernières, des tableaux contenant ces informations pour l'année académique 2011-2012, sont trouvés sur le site [31]. Ces derniers tiennent compte des deux types d'enseignement recherchés (primaire et secondaire) Cependant, ce site propose seulement le nombre d'étudiants dans les écoles spécialisées de chaque commune. Par example, l'ensemble des écoles primaires spécialisées de la commune d'Alost ont accueilli 346 étudiants. Un modèle d'un tableau pour la commune d'Alost est disponible dans la table D.8.

Enseignement non obligatoire

Sur le même site de référence [29] que celui où nous avons trouvé les listes des établissements primaires et secondaires de Flandre, il existe une liste contenant des informations pratiques sur l'ensemble des Hautes Ecoles de Flandre et sur les Universités. Cependant, pour obtenir le nombre d'étudiants inscrits dans chaque établissement, il faut consulter le site de la référence [20] contenant des fichiers pdf relatifs aux inscriptions et plus précisément, les inscriptions pour les écoles supérieures durant l'année 2010-2011.

3.2.3 Communauté germanophone

Pour terminer la liste complète des écoles en Belgique, il faut y ajouter la liste des établissements scolaires dans la communauté germanophone de la Belgique se trouvant

sur le site de référence [13]. Cette communauté est composée de neuf communes dont Eupen. Les écoles de cette communauté doivent transmettre le nombre d'étudiants inscrits dans leur établissement, deux fois par an au ministère correspondant. La première récolte s'effectue le dernier jour de classe du mois de septembre et la seconde, au mois de janvier.

Enseignement non obligatoire

Les données récoltées reprennent le nombre d'étudiants inscrits dans les établissements primaires et secondaires en octobre de l'année académique 2011-2012. Ils sont divisés en plusieurs types de système scolaire :

- l'enseignement subventionné gratuit (FSUW)
- l'enseignement subventionné officiel (OSUW)
- l'enseignement communautaire (GUW)

Enseignement obligatoire

Dans cette communauté, il existe également une Haute École, nommée Autonome Hochschule, située dans la commune d'Eupen. Les informations relatives à cet établissement se trouvent également sur le site de référence cité précédemment.

3.3 Création des fichiers

Avec les données récoltées, il a fallu créer trois fichiers contenant les différentes informations. Le premier, nommé PrimarySchool.txt dans le programme VirtualBelgium, contiendra les informations relatives à l'enseignement primaire telles que le numéro de Fase de l'établissement, le nombre d'étudiants qui y sont inscrits et le code INS[5] de sa commune. Le deuxième fichier, basé sur la même structure mais pour l'enseignement secondaire, se nommera SecondarySchool.txt. Pour finir, le dernier fichier, HigherSchool.txt se rapportera à l'enseignement supérieur.

Avant la création de ces derniers, il y a quelques points de modifications à réaliser sur le jeu de données :

- Nous pouvons remarquer que nous n'avons pas réussi à récolter les données nécessaires pour la même année scolaire. Cela n'est pas un choix délibéré mais une contrainte des données. Nous allons donc essayer de régler ce problème afin que notre modèle soit le plus pertinent possible et le mieux ajusté.

5. Le code INS d'une commune est une combinaison de plusieurs chiffres qui permettent d'identifier de manière unique, la commune.

Afin de détecter les données à modifier, une table récapitulative de nos données est proposée ci-dessous :

Typed'enseignement	Région	Année scolaire récoltée
Primaire	Fédération Wallonie- Bruxelles	2011-2012
Primaire ordinaire	Communauté flamande	2012-2013
Primaire spécialisé	Communauté flamande	2011-2012
Primaire	Communauté germanophone	2011-2012
Secondaire	Fédération Wallonie- Bruxelles	2011-2012
Secondaire	Communauté flamande	2011-2012
Secondaire	Communauté germanophone	2011-2012
Supérieur	Fédération Wallonie- Bruxelles	2010-2011
Supérieur	Communauté flamande	2010-2011
Supérieur	Communauté germanophone	2010-2011

TABLE 3.2 – Récapitulation des données récoltées

Pour chaque type d'enseignement, nous avons essayé de prendre la même année scolaire afin d'être le plus pertinent possible. Cependant pour les établissements flamands de l'enseignement primaire ordinaire, nous n'avons pas pu récupérer les informations concernant l'année académique 2011-2012. Nous avons donc essayé de comprendre et ensuite de modifier ces données.

Pour lire ces dernières, il y a une subtilité à comprendre. Effectivement, le nombre d'étudiants indiqué pour chaque école ne correspond pas exactement à l'année 2012-2013. Certaines écoles ont un nombre d'étudiants inscrits, déclaré au 1er février 2012 et d'autres au 1er octobre 2012 comme nous pouvons le voir dans les tables D.6 et D.7 représentant une petite partie des écoles flamandes de l'enseignement primaire ordinaire.

En réalité, ce tableau contient le nombre d'étudiants estimé pour l'année 2012-2013, déterminé soit en fonction l'année précédente, soit en fonction de l'année en cours. Ce qui signifie que le nombre d'étudiants dans la plupart des écoles correspond à l'année scolaire de référence 2011-2012. Nous allons donc seulement modifier le nombre d'étudiants inscrits pour les écoles qui se sont enregistrées au 1er octobre 2012.

Pour réaliser cette étape, nous comparons le nombre total des étudiants dans les écoles flamandes primaires ordinaires en 2011-2012 et 2012-2013. Entre ces deux

années, nous avons une augmentation de 0.0145% d'étudiants flamands. Or nous avons les nombres d'étudiants pour l'année 2012-2013 et nous les voulons pour l'année 2011-2012. Nous avons donc dû diminuer le nombre d'étudiants des écoles qui se sont enregistrées au 1er octobre 2012, de 0.0145 % afin d'obtenir leur nombre d'étudiants pour l'année 2011-2012.

- En ce qui concerne l'enseignement supérieur, les données de l'année 2011-2012 n'ont pas pu être récoltées parce que les informations relatives aux universités francophones ne sont pas disponibles pour cette année. Nous choisissons, dès lors pour référence, l'année scolaire 2010-2011 pour les écoles supérieures.

  De plus, le nombre d'étudiants par Haute École de la Fédération Wallonie-Bruxelles n'a pas pu être mis à notre disposition suite au décret cité précédemment. Seul les nombres d'étudiants par arrondissement ont été récoltés. Nous utilisons donc ces données en supposant qu'il existe une haute école dans chaque arrondissement, établie dans la commune principale de ce dernier. Ainsi, la réalité sera respectée au moins au niveau des arrondissements pour ces écoles.

Au final, nous pouvons créer les trois fichiers cités ci-dessus. Un récapitulatif du nombre d'étudiants et du nombre d'écoles est donné par la table 3.3. Pour lire ce tableau, il faut toujours avoir en tête que le nombre d'écoles supérieures est erroné en raison du deuxième point cité ci-dessus.

Typed'enseignement	Nombre d'écoles	Nombre d'étudiants
Primaire	5140	749689
Secondaire	1988	828198
Supérieur	55	356023

TABLE 3.3 – Nombres d'écoles récoltés avec le nombre respectif d'étudiants

Type d'enseignement	Nombre d'étudiants en réalité [10]	Nombre d'étudiants virtuels
Primaire	749689	1727134
Secondaire	828198	40353
Supérieur	356023	435980
Total	1933910	2203467

TABLE 3.4 – Nombres d'étudiants par type d'enseignement

Maintenant que nous avons créé nos fichiers, nous allons les comparer aux nombres d'étudiants virtuels du programme VirtualBelgium. Ces derniers sont stockés dans la table 3.4

Nous remarquons dans la table 3.4, que le nombre d'étudiants belges et le nombre d'étudiants virtuels est fortement différent. Nous observons une hausse important des étudiants virtuels de l'enseignement primaire comparé aux données récoltées. J. Barthelemy et Ph. Toint [7], ont utilisé les données provenant de l'enquête Mobel de 2001 pour créer la population. Il est impossible que le nombre d'étudiants primaires ait baissé autant entre 2001 et 2011. D'autant qu'il y a une hausse de la population belge entre 2003 et 2013 dans la catégorie des personnes âgées de moins de 18 ans. Cette information provient du site de référence [15]

En conclusion, la différence entre ces chiffres nous amène à une réflexion sur la construction de la population virtuelle réalisée par J. Barthelemy et Ph. Toint. Après une analyse de la population initiale, nous nous sommes rendu compte de certaines constatations erronées. Par exemple, le fait qu'il existe beaucoup d'étudiants virtuels âgés de plus de 40 ans allant toujours à l'école primaire.

Ces problèmes ont comme origine les corrélations entre les données lors de la création de la population virtuelle. En effet, par exemple, les croisements entre l'âge des individus, le type d'enseignement et le nombre d'étudiants n'ont pas été réalisés. Nous pouvons donc conclure que la population virtuelle ne peut correspondre à la réalité au point de vue des étudiants par type d'enseignement.

La modification de la population initiale ne peut être réalisée dans ce travail par un manque de temps. Toutefois, nous pouvons toujours créer un modèle d'assignation des écoles avec cette population en modifiant les données récoltées. Cela permettrait d'ajuster le modèle en attendant le changement de la population initiale dans un futur travail.

Les données récoltées ont été réajustées en fonction de la population de VirtualBelgium tout en gardant la même répartition des étudiants dans les écoles. Ce qui signifie que pour chaque école d'un type d'enseignement, son nombre d'étudiants va être augmenté ou diminué en fonction d'un même coefficient, propre au type.

Par exemple, en ce qui concerne les écoles primaires, leur nombre d'étudiants va être multiplié par le coefficient calculé ci-dessous :

10. Ces nombres dépendent de l'année scolaire choisie lors de la récolte

$$\text{coefficient}_{\text{primaire}} = \frac{1727134}{749689} = 2.303080$$

L'ensemble des coefficients par type d'enseignement est repris dans la table 3.5 ainsi que les nombres finaux d'étudiants dans les écoles. Ces derniers ne correspondent pas exactement aux nombres d'étudiants virtuels dû aux erreurs d'arrondis.

Type d'enseignement	Nombre d'étudiants virtuels	Nombre d'étudiants en réalité	Coefficient	Nombre final de places dans écoles
Primaire	749689	1727134	2.303080	1727158
Secondaire	828198	40353	0.0487	40580
Supérieur	356023	435980	1.2246	435987
Total	1933910	2203467		2203725

TABLE 3.5 – Coefficients d'ajustement du nombre d'étudiants dans les écoles récoltées

En ce qui concerne les écoles secondaires, nous avons dû effectuer une diminution de places disponibles ce qui a entrainé une suppression de 34 écoles dont leur quota est devenu nul.

Suite à ces modifications, nous pouvons créer de nouveaux fichiers qui seront utilisés dans le programme VirtualBelgium. Lors de l'utilisation de ces fichiers, il faut toujours garder en mémoire les différentes opérations effectuées.

3.4 Conclusion

Comme on peut l'imaginer, cette collecte d'informations a demandé un certain temps, tant dans la recherche initiale, que dans les courriers échangés pour les demandes officielles, que dans l'attente des résultats.

Le traitement des données relatives aux établissements scolaires et leurs inscriptions se sont concrétisés par la création de 3 fichiers. Avec ces trois fichiers, le programme va pouvoir contenir toutes les informations nécessaires pour implanter les écoles dans le réseau de VirtualBelgium et assigner une de ces écoles à chaque étudiant en fonction de leur domicile et de leur diplôme. Les modèles créés pour la réalisation de ces tâches sont développés dans le chapitre suivant.

En ce qui concerne le géocodage des écoles, nous n'avons pas su récupérer les adresses de tous les établissements scolaires.

Chapitre 4

Établissements scolaires dans VirtualBelgium : Partie théorique

Cette étude a pour but d'améliorer l'activité liée à l'école dans le plateforme VirtualBelgium. Comme expliqué dans le second chapitre, l'établissement scolaire de l'individu est actuellement choisi aléatoirement, de manière uniforme, parmi tous les noeuds du réseau virtuel. Ce chapitre permet d'expliquer les différentes méthodes théoriques testées pour améliorer cette partie ainsi que leurs résultats. Le prochain chapitre contiendra les différentes techniques de programmation employées pour simuler ces méthodes.

Avant de décrire la méthode de distribution des établissements scolaires dans le réseau virtuel et celle d'assignation d'une école aux étudiants, appelées respectivement dans ce travail, méthode de distribution des écoles et modèle d'assignation, il est important de préciser les bases du fonctionnement du réseau virtuel.

4.1 Réseau de VirtualBelgium

VirtualBelgium possède un réseau proche de celui de la Belgique, qui est construit grâce à un fichier extrait d'OpenStreetMap [1][32]. Ce dernier est composé :

- d'arcs (Links dans le programme) qui symbolisent les routes de la Belgique,
- de noeuds (Nodes dans le programme) qui représentent
  - soit un carrefour en Belgique,
  - soit un lieu où la route change de direction.

Pour mieux comprendre le choix des noeuds sur les routes d'OpenStreetMap, un exemple est mis à votre disposition à la figure 4.1, concernant une partie de la route de Bruyère. Cette portion possède trois noeuds :

1. Le site OpenStreetMap a été présenté dans le premier chapitre.

- un pour décrire l'emplacement du carrefour entre les trois routes ,
- les deux autres pour signaler les changements de direction de la route.

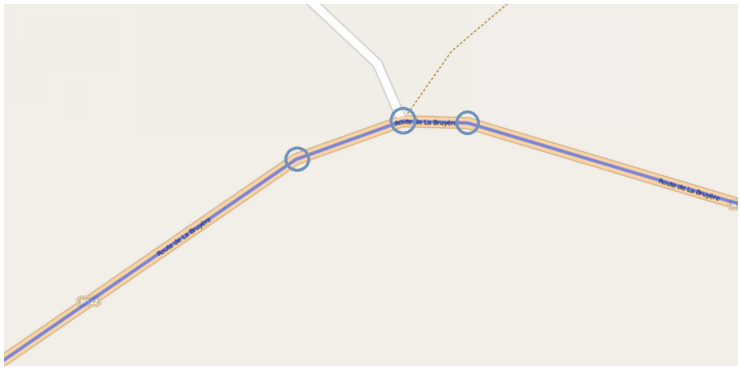


FIGURE 4.1 – Route de la Bruyère. Cette figure est inspirée du site d'OpenStreetMap [32]

Chaque noeud d'Openstreetmap est décrit, principalement, par un identifiant unique et par ses coordonnées géographiques. Par exemple, le noeud désignant le carrefour sur l'image 4.1 est représenté par
<node id="1041627727" visible="true" version="2" changeset="7103883" timestamp="2011-01-27T14 :01 :58Z" user="Renaud Michel" uid="74476" lat="50.5780362" lon="4.8913779" /> [2].

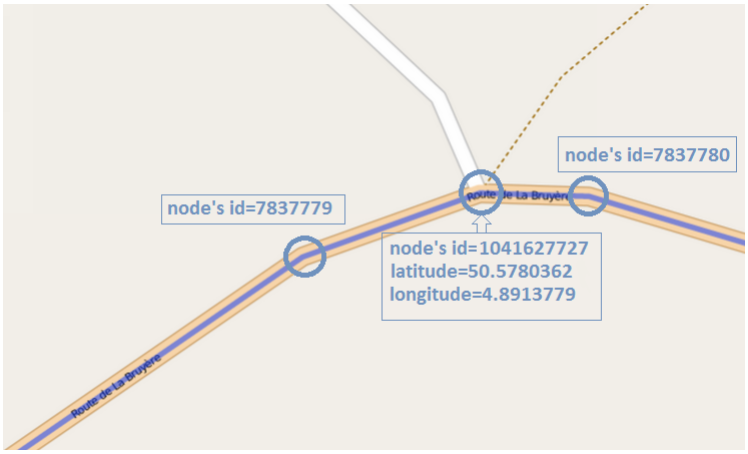


FIGURE 4.2 – Route de la Bruyère avec l'identifiant d'un noeud

Chaque arc est représenté par un identifiant unique et par l'ensemble des noeuds qui le constituent. Il peut également posséder des propriétés précédées par le mot clé "tag", telles que le nom de la route, le type de voie de circulation, ... Toutes les informations concernant ces propriétés peuvent être consultées via le site OpenStreetMap [24].

2. Cette description provient du site de référence [32]

La portion de route représentée à la figure 4.2, fait partie d'un arc d'OpenStreetMap qui est décrit par l'expression suivante :

<way id="232660138" visible="true" version="1" changeset="17254689" timestamp="2013-08-07T14 :42 :13Z" user="Roger1959" uid="1704305">
<nd ref="7837781"/>
<nd ref="7837780"/>
<nd ref="1041627727"/>
<nd ref="7837779"/>
<nd ref="7837778"/>
<nd ref="7837777"/>
<nd ref="299691089"/>
<nd ref="7837776"/>
<nd ref="7837775"/>
<nd ref="7837774"/>
<tag k="highway" v="secondary"/>
<tag k="name" v="Route de La Bruyère"/>
<tag k="ref" v="N912"/>
</way>

Une adaptation doit cependant être réalisée avant que le fichier soit exploitable par le programme VirtualBelgium. En effet, les noeuds doivent posséder des coordonnées cartésiennes. Pour effectuer ces transformations, le logiciel MATSIM est utilisé car il permet d'obtenir un fichier d'extension xml contenant les noeuds décrits sous la forme adéquate. Cette étape a été réalisée par J. Barthélemy et Ph. Toint[6].

Après ce changement, le fichier peut être lu et utilisé pour la création des noeuds et des liens dans VirtualBelgium. En terme de programmation orientée objet, cela revient à créer des objets de la classe Node et de la classe Links en référence au premier chapitre. Toutes les informations relatives à ces deux classes peuvent se trouver dans l'annexe C.

En résumé, une route de Belgique est décrite par un ensemble d'arcs dans le réseau de VirtualBelgium. Ces derniers sont composés de noeuds. Cependant, le choix des noeuds d'un arc ne dépend pas des bâtiments ou des habitations qui se situent le long de ce dernier mais de la direction de celui-ci ou des croisements avec d'autres arcs. On peut donc en conclure que sur un route, le nombre de noeuds est indépendant du nombre de bâtiments. Par exemple, sur une route de campagne, suite aux changements fréquents de direction, le nombre de noeuds est plus important que le nombre d'habitations. À contrario, une rue dans un quartier résidentiel est caractérisée par un petit nombre de noeuds comparé au nombre de maisons.

Cette remarque met en évidence le fait que deux ménages virtuels peuvent être associés au même noeud lors du modèle d'assignation d'un noeud à un ménage[6]. Cette constatation s'étend également aux activités et donc aux écoles. Deux écoles se situant dans la même rue en Belgique, peuvent être assignées au même noeud dans le réseau virtuel.

En conclusion, un noeud dans VirtualBelgium peut abriter plusieurs ménages et/ou plusieurs activités : par abus de langage, un noeud peut être dès lors une habitation, un lieu de travail, une école et un lieu de loisir en même temps.

4.2 Méthode de distribution des écoles

Comme expliqué dans le chapitre 3, le géocodage des écoles de Belgique dans le réseau virtuel ne peut être réalisé dans ce travail dû à un manque de temps. Une nouvelle méthode doit être trouvée afin d'implanter les écoles dans ce réseau. Dans ce travail, cette dernière sera appelée méthode de distribution et doit prendre en considération deux points.

- D'une part, la méthode doit tenir compte du plus petit niveau de désagrégation : la commune. Cela signifie que le respect de la réalité doit aller jusqu'au niveau communal et qu'il n'est pas possible de distinguer les villes et les villages dans les communes du réseau de VirtualBelgium.

  Toutes les communes virtuelles doivent posséder un nombre d'établissements scolaires primaires, identique à celui calculé à partir des données récoltées. Ceci est également valable pour les autres types d'enseignement.

  Prenons un exemple : si la commune de Namur en Wallonie possède 54 écoles pour l'année académique 2011-2012, alors la commune virtuelle de Namur doit également posséder 54 écoles primaires dans le réseau de VirtualBelgium.

- D'autre part, chaque nœud désigné comme une école dans une commune virtuelle doit correspondre à un établissement scolaire réel de cette commune, analysé dans les données récoltées. En d'autres termes, une école virtuelle possède un nombre total d'étudiants qu'elle peut accueillir et ce dernier représente le nombre d'étudiants inscrits dans l'école réelle correspondante.

La méthode créée pour désigner l'emplacement de l'école dans sa commune, tout en respectant les deux points cités ci-dessus, consiste en un tirage aléatoire, de manière uni-

forme, d'un nœud de la commune.

En conclusion, **la méthode de distribution des écoles** se résume :

- à prendre aléatoirement autant de nœuds dans une commune qu'il y a d'établissements scolaires dans cette dernière,
- à faire correspondre chaque nœud à une école de sa commune, décrite dans les données récoltées.

Comme expliqué dans la section précédente concernant le réseau de VirtualBelgium, les nœuds peuvent contenir plusieurs ménages et peuvent être un lieu où se déroulent plusieurs activités telles que l'école. Lors du choix des nœuds dans une commune, un nœud peut donc être sélectionné plusieurs fois et représenter plusieurs fois le même type d'établissement. Il n'y a donc aucune restriction dans le tirage des nœuds. De plus, dans la réalité, un bâtiment peut contenir plusieurs écoles différentes. Par exemple, il y a deux écoles situées rue Pierre Longin n°1 à Bruxelles :

- Ecole primaire communale n°18 Les Etangs avec le numéro de FASE égal à 15
- Gemeentelijke Basisschool avec le numéro de FASE égal à 3988

Chaque école virtuelle possède donc le nombre souhaité d'étudiants dans cette dernière. Ce nombre est appelé « quota ».

Afin de permettre une compréhension rapide de la méthode, un exemple est mis à votre disposition. À l'aide du fichier PrimarySchool.txt [3], les établissements scolaires primaires de 7 communes de la communauté flamande sont réunis dans le tableau 4.1. où le nombre d'écoles par commune est mis en évidence.

Prenons une carte [4] représentant les 7 communes ainsi que les nœuds qui les caractérisent. La première étape de la méthode de distribution des écoles expliquée ci-dessus, consiste à choisir aléatoirement, de manière uniforme, un certain nombre de nœuds dans chaque commune. Par exemple, la commune de Ternat possède, dans la réalité, 4 établissements primaires pour l'année 2011-2012. Après un choix uniformément aléatoire, il y a 4 nœuds du réseau virtuel, inclus dans la commune de Ternat qui deviennent des écoles primaires. En ce qui concerne la commune d'Asse, on sélectionne 7 nœuds dans la commune virtuelle. On peut effectuer le même raisonnement pour les autres communes.

3. Les lignes du fichier PrimarySchool.txt aidant à remplir les lignes du tableau 4.1 se trouvent dans l'annexe...

4. Cette image est uniquement une illustration aidant à comprendre la méthode. Les nœuds représentés ont été placés aléatoirement et n'indiquent pas un endroit précis dans le réseau de VirtualBelgium. On peut également noter que seulement une minorité de nœuds a été illustré.

Cette étape est représentée par la figure 4.3.

Commune	Ternat	Asse	Dilbeek	Lennik	Roostall	Liederkerke	Affligem
Nombre d'étudiants pour chaque école primaire dans la commune	105 334 240 337	336 275 282 288 146 226 133	148 230 175 239 157 172 378 208 272 264 247	114 168 88 326	227 239 180	182 245 216	147 162 254 169
Nombre d'écoles dans la commune	4	7	11	4	3	3	4

TABLE 4.1 – Informations concernant les écoles primaires dans 7 communes du Brabant Flamand

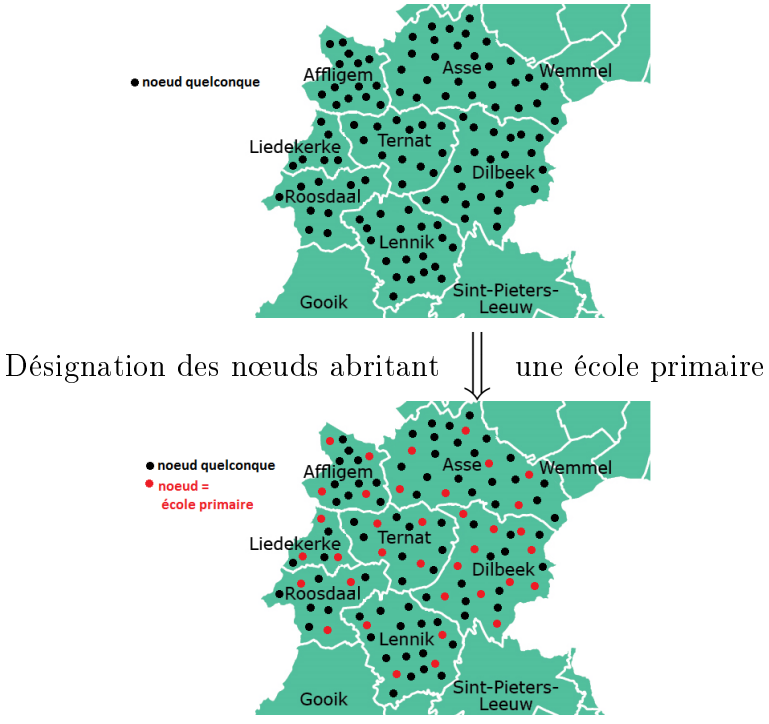


FIGURE 4.3 – Désignation des nœuds abritant une école primaire

Ensuite, la prochaine étape consiste à faire correspondre les nœuds sélectionnés de chaque commune, aux établissements primaires cités ci-dessus. Ce choix est également aléatoire. Cette étape est représentée par l'image 4.4

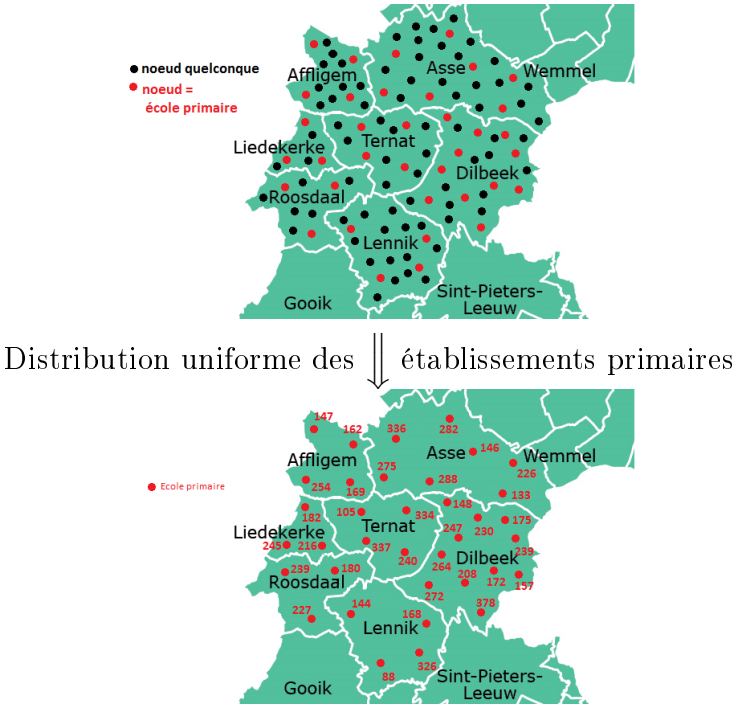


FIGURE 4.4 – Distribution uniforme des établissements primaires

À la fin de cette méthode, le réseau virtuel possède des établissements primaires pour ces 7 communes tout en respectant les données récoltées et analysées.

Maintenant que la méthode de distribution des écoles a été développée, il faut trouver le moyen de distinguer les nœuds représentant une/des école(s) parmi l'ensemble des nœuds constituant le réseau virtuel dans le programme de VirtuelBelgium. Différentes tentatives ont été réalisées. La version finale de l'implémentation de ce modèle est expliquée dans la section 5.2 du prochain chapitre.

4.3 Modèle d'assignation des écoles dans VirtualBelgium

Dans cette section, les différentes méthodes permettant d'assigner une école à un individu en connaissant son domicile, sont détaillées. Ces dernières ne s'appliquent pas à tous les individus du programme VirtualBelgium. En effet, seuls les étudiants virtuels se voient attribuer une école.

Une section sera donc consacrée à distinguer les étudiants parmi tous les individus du programme. Ensuite, une petite explication concernant les différents types d'écoles est donnée. Et pour terminer, les différentes méthodes seront énoncées.

4.3.1 Étudiant de VirtualBelgium

Comme expliqué dans le premier chapitre, les individus virtuels possèdent différentes caractéristiques (disponibles à la section 1.3.2) qui permettent de leur attribuer un certain type, tel que, par exemple, le type FEMME-SEULE-ACTIF. La plus intéressante des caractéristiques dans cette démarcation est la catégorie professionnelle. Cette variable permet de déterminer si l'individu possède ou non un emploi ou s'il suit des cours.

Une seule valeur est possible pour cette variable, cela implique qu'une hypothèse a été formulée lors de la conception du programme par J. Barthélémy et Ph. Toint [6] :

Hypothèse 4.1. *Un individu ne peut pas travailler en même temps que suivre des cours.*

En conclusion, grâce à cette caractéristique, nous pouvons distinguer les individus qui sont considérés comme des étudiants. Le nombre d'étudiants virtuels s'élève à 2203467.

4.3.2 Type d'individu

Avant d'assigner une école à un étudiant, il faut déterminer le type de cette dernière en fonction de ses caractéristiques. Pour rappel, il y a trois enseignements possibles pour les individus virtuels : primaire, secondaire et supérieur. Aucun étudiant virtuel ne sera dirigé vers un établissement pré-scolaire car une hypothèse concernant l'inactivité des enfants de 0-5 a été imposée par J. Barthélemey, Ph. Toint et E. Cornelis.

Hypothèse 4.2. *Les enfants du programme VirtualBelgium, âgés de moins de 6 ans sont inactifs.*

Le choix du type d'enseignement pour un individu est déterminé en fonction de son niveau d'éducation, une caractéristique qui le détermine (décrite dans la section 1.3.2). Cette variable tient compte du dernier diplôme obtenu par l'individu. Il y a 4 possibilités :

- **None :** la personne ne possède pas de diplôme.
- **Primaire :** la personne possède le CEB [5]. Ce diplôme lui permet d'accéder à l'enseignement secondaire.

5. CEB : Certificat d'Etude de Base

- **Secondaire :** l'individu dispose d'un diplôme de l'enseignement secondaire : CESS [6]. Celui-ci permet aux individus de commencer des études supérieures s'ils le souhaitent.
- **Supérieur :** l'individu a entrepris des études supérieures et il les a réussies. Si un étudiant possède ce diplôme, cela signifie qu'il recommence des études supérieures.

En tenant compte de ces variables : « Catégorie Professionnelle » et « Niveau d'éducation », les écoles ne seront assignées qu'aux étudiants et le type des établissements seront déterminés en fonction du diplôme obtenu de ces derniers. Un petit récapitulatif est mis à votre disposition à la table 4.2.

Catégorie Professionnelle	Niveau d'éducation	Type d'enseignement	Nombre d'individus dans VirtualBelgium
Student	none	primaire	1727134
Student	primaire	secondaire	40353
Student	secondaire	supérieur	302012
Student	supérieur	supérieur	133968

TABLE 4.2 – Les individus pris en compte dans le travail avec le type d'enseignement associé.

4.3.3 Caractéristiques essentielles du modèle d'assignation

Notre étude consiste à créer un modèle d'assignation des écoles aux étudiants virtuels de VirtualBelgium. Afin de pouvoir déterminer ce modèle, différentes informations concernant les hypothèses posées, les contraintes, les données et le contexte seront énumérés tout au long de cette section. Ensuite, un examen des différents modèles de la littérature est réalisé dans la prochaine section afin d'en sélectionner un adapté à notre étude.

1. Problématique

Le modèle d'assignation consiste à attribuer un établissement aux étudiants de VirtualBelgium en fonction de leur dernier diplôme obtenu.

2. Données

Les données utiles pour cette méthode sont les suivantes :

- le nombre d'établissements scolaires belges par commune et par type d'enseignement et le nombre souhaité d'étudiants dans chaque école virtuelle.

6. CESS : Certificat d'Enseignement Secondaire Supérieur

Pour chaque école et plus précisément pour chaque nœud du réseau désignant une école, le nombre de trajets dont il est la destination finale est également connu. En effet, ce dernier peut être le quota de l'école, c'est-à-dire, le nombre souhaité d'étudiants dans cette dernière.

Soient : e une école abritée par le nœud j,
D_j^e le nombre de trajets arrivant au nœud j, ayant pour but l'école e,
quota_e le quota d'étudiants de l'école e.

$$D_j^e = \text{quota}_e$$

En d'autres mots, si le quota d'un établissement scolaire, noté e, attribué au nœud noté k, est égal à 200, alors la somme des déplacements dont la destination est cette école, noté D_k^e vaut 200.

- des données réelles concernant la mobilité des belges. Ces dernières proviennent de l'enquête de Mobilité Mobel.

3. Hypothèses

Hypothèse 4.3. *Le modèle d'assignation doit être applicable à tous les étudiants.*

Actuellement, il existe deux sortes d'étudiants dans VirtualBelgium :
- d'une part, il y a les étudiants dont la chaine d'activités contient l'activité « École »,
- d'autre part, il y a les étudiants qui ont été assignés à une chaine d'activités correspondant aux weekends. En d'autres mots, ils ne vont pas à l'école pendant la journée virtuelle.

Le programme VirtualBelgium actuel ne s'occupe que des étudiants possédant l'activité « École » . Cependant, en ce qui concerne le nouveau modèle d'assignation, nous souhaitons qu'il puisse être adapté à tous les étudiants virtuels même si certains ne se déplacent jamais vers ce lieu. Cette hypothèse permet au modèle d'être applicable dans l'optique future de l'assignation de chaines d'activités différentes en fonction du jour de la semaine.

Hypothèse 4.4. ***Hypothèse de domiciliation :***

Le choix de l'école doit être réalisé en fonction du lieu du domicile.

Comme expliqué dans le chapitre 2, J. Barthelemy, Ph. Toint et E. Cornelis ont développé une distribution permettant de connaitre les distances effectuées par les étudiants

virtuels en respectant celles parcourues dans la réalité durant l'année 2001. Plus précisément, elle est construite à partir de tous les déplacements ayant pour motif « École ». Une distance générée via un générateur de nombres pseudo-aléatoires suivant cette distribution, est donc mesurée en partant du lieu de l'activité précédente.

Prenons, par exemple, un individu effectuant ses courses. Pour sa prochaine activité, il souhaite aller à l'école. Le modèle actuel d'assignation, expliqué dans la section 2.2.2.2, génère une distance que l'individu effectuera pour atteindre son école à partir de son lieu actuel. Le lieu de l'activité « école » dépend donc du lieu où l'individu réalise ses courses.

Pour le modèle d'assignation des écoles, il nous parait plus pertinent de déterminer le choix de l'école en fonction du domicile de l'individu. Nous avons donc posé l'hypothèse 4.4.

Cette hypothèse est posée malgré le fait que certains parents peuvent choisir l'établissement de leurs enfants en fonction du lieu de leur travail. Cependant, aucune donnée relative à ce choix n'est à notre portée.

4. Contraintes :

Il existe trois contraintes que le modèle doit respecter.

- Le modèle d'assignation ne peut pas augmenter significativement la durée de l'exécution du programme. En effet, le programme actuel considérant l'évolution spatiale des individus, possède un temps d'exécution égal à plus de 9h en utilisant 500 processus en programmation parallèle. Vu que cette durée est assez importante, il faut donc éviter toute augmentation inutile de temps.

- Le modèle doit respecter les données réelles concernant les écoles.

- Le modèle doit respecter le nombre souhaité d'étudiants dans chaque école virtuelle.

5. Contexte :

Suite aux hypothèses et contraintes posées, nous pouvons enfin définir le contexte dans lequel le modèle doit être applicable.

1. Le modèle doit être adapté à tous les étudiants de VirtualBelgium indépendamment de leur chaine d'activités.
2. Les étudiants virtuels choisissent leur établissement scolaire en fonction de leur domicile.

3. Le type d'établissement scolaire choisi pour un individu doit être déterminé en fonction du dernier diplôme qu'il a reçu (plus de détails dans la section 4.3.2).
4. Le modèle doit respecter le nombre total souhaité d'étudiants dans les écoles virtuelles.

4.3.4 Recherche de modèles

Comme expliqué ci-dessus, notre modèle consiste à attribuer une école aux élèves de VirtualBelgium, en d'autres termes, de distribuer les déplacements scolaires dans le réseau. Le nombre de trajets vers chaque école est connu et le nombre d'étudiants est également en notre possession. Il est donc possible de connaitre le nombre de déplacements scolaires partant et arrivant pour chaque nœud du réseau. Or cette caractéristique est propre aux modèles « Trip distribution ».

Dans la littérature, les modèles relatifs aux « Trip distribution » ne considèrent pas les déplacements comme un ensemble mais divisent ces derniers en fonction de l'endroit du départ et celui d'arrivée en créant une matrice dite O-D où O signifie origine et D, destination. Cette matrice comprend le nombre de trajets effectués entre chaque nœud d'origine et d'arrivée :

$$(\text{matrice } O-D)_{i,j} = T_{i,j}$$

où $T_{i,j}$ est le nombre de trajets effectués entre le nœud i et le nœud j. Elle peut être construite à partir de la matrice O contenant le nombre souhaité de trajets partant de chaque nœud et la matrice D, comprenant le nombre souhaité de trajets arrivant à chaque nœud.

Soient : T le nombre de trajets totaux (connu),
O_i le nombre de trajets partant du nœud i (connu),
D_j le nombre de trajets arrivant au nœud j (connu),
la matrice $O-D$ du réseau de la figure 4.5 est donnée ci-dessous dans la table 4.3.

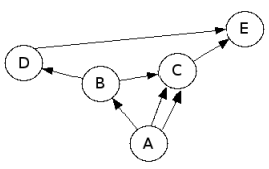


FIGURE 4.5 – Exemple de graphe pour expliquer la matrice O-D

	Destination	Matrice des origines O
Origine	$\begin{pmatrix} 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 2 \\ 1 \\ 1 \\ 0 \end{pmatrix}$
Matrice des destinations D	$\begin{pmatrix} 0 & 1 & 3 & 1 & 2 \end{pmatrix}$	$T = 7$

TABLE 4.3 – O-D matrice associée à la figure 4.5

Une égalité est mise en évidence dans ce type de modèle :

Soient : n_o le nombre de nœuds dans le réseau considérés comme lieu de départ (origine du déplacement)

n_d le nombre de nœuds dans le réseau considérés comme destination.

$$\sum_{d=1}^{n_d} D_d = \sum_{o=1}^{n_o} O_s$$

Les modèles relatifs « Trip distribution » peuvent être classés dans deux catégories distinctes :

- d'une part, les modèles qui ne respectent qu'une seule contrainte telle que

$$\sum_{j=0}^{n_d} T_{i,j} = O_i \tag{4.1}$$

OU

$$\sum_{i=0}^{n_o} T_{i,j} = D_j \tag{4.2}$$

- d'autre part, les modèles qui respectent les deux contraintes citées ci-dessus en même temps.

$$\sum_{j=0}^{n_d} T_{i,j} = O_i$$

ET

$$\sum_{i=0}^{n_o} T_{i,j} = D_j$$

Maintenant que nous connaissons les différentes notations des modèles « Trip distribution », nous pouvons redéfinir le quatrième élément du contexte du modèle d'assignation avec les bons symboles. « Le modèle doit respecter le nombre total d'étudiant souhaités dans les écoles virtuelles » peut être exprimé comme :

soient :
- n le nombre de nœuds dans le réseau virtuel,
- n_e le nombre de nœuds abritant au moins une école,
- $T^e_{i,j}$ le nombre de trajets qui ont pour origine, le nœud i et pour destination, l'école e du nœud j
- D^e_j le quota d'étudiants pour l'école e du nœud j

$$\sum_{i=1}^{n} T^e_{i,j} = D^e_j \ \forall \ j \quad \text{tel que le nœud j abrite au moins une école} \tag{4.3}$$

Cette expression s'applique aux écoles virtuelles mais lorsqu'on se préoccupe uniquement des noeuds virtuels, nous retrouvons l'expression (4.2) à respecter telle que

$$\sum_{i=1}^{n} T_{i,j}^{\text{scolaire}} = D_j^{\text{scolaire}} \tag{4.4}$$

où

- $\sum_{\text{école } e \in \text{ nœud } j} D^e_j = D_j^{\text{scolaire}} \quad \forall \ j$ tel que le nœud j abrite au moins une école
- D_j^{scolaire} représente le nombre souhaité de trajets **scolaires** vers le nœud j.
- $T_{i,j}^{\text{scolaire}}$ représente le nombre de trajets **scolaires** entre le nœud i et le nœud j.

Nous devons donc chercher dans la littérature les différents modèles " Trip distribution" qui respectent :

- soit la contrainte (4.4) relative aux noeuds . Cependant, il faudra trouver une méthode supplémentaire afin de répartir les trajets trouvés avec $T_{i,j}$ entre les écoles du noeud j pour pouvoir respecter l'équation (4.3).
- soit directement la contrainte relative aux écoles (4.3)

La recherche se divise en deux parties distinctes en fonction du rôle joué par la distance entre l'école et le domicile pour chaque individu.

- Les premiers modèles annoncés ci-dessous n'ont aucune contrainte concernant la distance.
- Les autres utilisent une distance entre le domicile et l'école fixée antérieurement.

4.3.4.1 Modèles sans contrainte de distance

Dans cette section, différents modèles simplement contraints ou doublement contraints provenant de la littérature sont exposés tels que le modèle gravitationnel. Les différents modèles de cette section sont principalement inspirés du [33] même s'il y a eu d'autres recherches.

1. Le modèle gravitationnel

Le modèle gravitationnel est un modèle inspiré de la loi de la gravitation, écrite par Isaac Newton [7]. En concordance avec cette dernière, ce modèle permet d'estimer le nombre de trajets effectués entre deux noeuds en émettant les deux hypothèses suivantes :

1. Le nombre de trajets effectués entre deux noeuds, notés o et d , est proportionnel au nombre de trajets d'origine o et au nombre de trajets de destination d.
2. Le nombre de trajets effectués entre deux noeuds, notés o et d est inversement proportionnel à la distance qui les sépare.

Ce modèle permet donc de créer la matrice O-D avec la formule suivante :

soient : n_o le nombre de noeuds dans le réseau considérés comme lieu de départ (origine du déplacement),
n_d le nombre de noeuds dans le réseau considérés comme destination,
O_i le nombre souhaité de trajets partant du noeud i,
D_j le nombre souhaité de trajets arrivant au noeud j,
$F_{i,j}$ le facteur de friction. Cet élément permet de connaitre la distance entre les noeuds i et j.
K constante à calibrer,

$$T_{i,j} = K\, O_i\, D_j F_{i,j} \;\forall\; i \in \{1, \cdots, n_o\} \text{ et } j \in \{1, \cdots, n_d\} \tag{4.5}$$

7. D'après le site de référence [41], la loi de la gravitation écrite par Newton peut être énoncée comme ci-dessous :

« Deux corps ponctuels de masses respectives M_A et M_B s'attirent avec des forces de mêmes valeurs (mais vectoriellement opposées), proportionnelles à chacune des masses, et inversement proportionnelles au carré de la distance qui les sépare. La force exercée sur le corps B par le corps A est vectoriellement donnée par :

$$F_{A/B} = F_{B/A} = G\frac{M_A M_B}{d^2}$$

»

Il existe une deuxième expression de la formule pour ce modèle. Cette dernière est plus attentive à la proportionnalité des trajets telle que

soient :
- n_o le nombre de noeuds dans le réseau considérés comme lieu de départ (origine du déplacement),
- n_d le nombre de noeuds dans le réseau considérés comme destination,
- O_i le nombre souhaité de trajets partant du noeud i,
- D_j le nombre souhaité de trajets arrivant au noeud j,
- $F_{i,j}$ le facteur de friction.
- $K_{i,j}$ Coefficient socio-économique entre les noeuds i et j. Il permet d'ajuster $T_{i,j}$ en tenant compte des caractéristiques socio-économique des deux noeuds.

$$T_{i,j} = O_i \frac{D_j\, F_{i,j}\, K_{i,j}}{\sum_{k=1}^{n_d} D_k\, F_{i,k}\, K_{i,k}} \;\forall\; i \in \{1, \cdots, n_o\} \text{ et } j \in \{1, \cdots, n_d\} \tag{4.6}$$

La matrice O-D est donc construite à partir des valeurs de l'expression (4.6), arrondies au plus proche

Ce modèle peut être utilisé dans notre travail car la particularité de ce dernier, c'est-à-dire, « $T_{i,j}$ inversement proportionnel à la distance », est respectée par les données réelles. En effet, la fonction de densité relative aux distances récoltées durant l'enquête, a été construite via le logiciel MATLAB [8]. Nous avons pu observer qu'il y avait de moins en moins de déplacements de longue distance.

Ce modèle est simplement contraint car il respecte la contrainte (4.1).

$$\sum_{j=0}^{n_d} T_{i,j} = O_i \quad \forall\; 1 \leq i \leq n_o$$

Mais il ne satisfait pas toujours la contrainte (4.2) et donc si on l'applique dans notre étude, l'équation (4.4) ne serait pas respectée et nous n'obtiendrions pas le résultat voulu. Or il existe un moyen de satisfaire cette équation tout en utilisant le modèle gravitationnel. En d'autres mots, il suffit de transformer le modèle simplement contraint en un modèle doublement contraint. Pour obtenir ce résultat, il existe deux manières.

1.1. **Première amélioration**

8. MATLAB est un langage avec une interface graphique pour le calcul numérique, la visualisation,...

Pour pouvoir conserver le nombre de trajets vers une destination avec le modèle gravitationnel, l'approche itérative peut être appliquée. Les étapes de cette méthode, adaptée à ce modèle, sont citées ci-dessous et permettent donc d'obtenir des valeurs finales $D_j \ \forall \ 1 \leq j \leq n_d$ assez proches que celles souhaitées. Le nombre d'itérations dépend de la précision voulue. Cette amélioration est inspirée des sites de référence [37] et [38]

Initialisation (Itération 0)

- $D_j \ \forall \ \ 1 \leq j \leq n_d$, $O_i \ \forall \ \ 1 \leq i \leq n_o$ sont connues.
- $F_{i,j}, K_{i,j} \quad \forall \ \ 1 \leq i \leq n_o, \forall \ 1 \leq j \leq n_d$, sont calculées.
- $T_{i,j}$ sont calculées grâce à la formule (4.6).

Après l'application du modèle, les résultats obtenus concernant les destinations, sont notés

$$D_j^{0,end} \ \forall \ 1 \leq j \leq n_d$$

On va donc vérifier si ces résultats correspondent aux valeurs souhaitées pour les destinations. Si $D_j^{0,end} \neq D_j$ alors une deuxième itération s'impose avec un ajustement des attractivités $D_j \ \forall \ 1 \leq j \leq n_d$.

Itération n°1

- Ajustement des variables $D_j \ \forall \ 1 \leq j \leq n_d$ pour l'itération 1 en fonction de leur valeur à l'initialisation tel que

$$D_j^1 = \frac{(D_j^0)^2}{D_j^{0,end}} \ \forall \ 1 \leq j \leq n_d \tag{4.7}$$

- Application du modèle gravitationnel avec changement des variables D_j par D_j^1 $\forall 1 \leq j \leq n_d$ dans l'équation (4.6). Les résultats obtenus se notent $D_j^{1,end} \ \forall \ 1 \leq j \leq n_d$

Les résultats sont toujours comparés à ceux souhaités et si ces derniers ne conviennent pas, une nouvelle itération doit être réalisée. Le modèle s'arrête lorsque les résultats sont identiques aux valeurs de D_j souhaitées $\forall \ 1 \leq j \leq n_d$.

Admettons que la $m-1^{\text{ième}}$ itération ne donne pas de valeur concluante telle que $D_j^{m-1,end} \neq D_j \ \ \forall \ 1 \leq j \leq n_d$ où $m > 1$, alors l'itération m doit être effectuée.

Itération m

- Ajustement des variables $D_j^{m-1}\ \forall\ 1 \leq j \leq n_d$ pour l'itération m tel que

$$D_j^m = \frac{D_j^0 D_j^{m-1}}{D_j^{m-1,end}}\ \forall\ 1 \leq j \leq n_d \tag{4.8}$$

- Application du modèle gravitationnel avec changement des variables D_j par D_j^m $\forall\quad 1 \leq j \leq n_d$ dans l'équation (4.6). Les résultats obtenus se notent $D_j^{m,end}\ \forall\ 1 \leq j \leq n_d$

Cette méthode itérative permet donc d'obtenir le respect de la contrainte (4.2) si le modèle converge.

Adaptation de ce modèle à notre cas :

Le modèle gravitationnel peut donc être utilisé pour distribuer les trajets à travers les noeuds du réseau virtuel. Cela implique que seuls les trajets scolaires sont analysés. Nous pouvons donc reformuler l'équation (4.6) :

soient :	$n_{domicile}$	le nombre de noeuds considérés comme le domicile d'un étudiant,
	$n_{noeud_{ecole}}$	le nombre de noeuds abritant au moins une école,
	O_i^{scolaire}	le nombre souhaité de trajets scolaires partant du noeud i,
	D_j^{scolaire}	le nombre souhaité de trajets scolaires arrivant au noeud j,
	$F_{i,j}$	le facteur de friction.
	$K_{i,j}$	coefficient socio-économique entre les noeuds i et j

$$\forall\ i \in \{1, \cdots, n_{domicile}\} \text{ et } j \in \{1, \cdots, n_{noeud_{ecole}}\}$$

$$T_{i,j}^{\text{scolaire}} = O_i^{\text{scolaire}} \frac{D_j^{\text{scolaire}} F_{i,j} K_{i,j}}{\sum_{k=1}^{n_{noeud_{ecole}}} D_k^{\text{scolaire}} F_{i,k} K_{i,k}} \tag{4.9}$$

Après l'application de ce modèle pour trouver la matrice O-D, les trajets associés à un noeud abritant au moins une école doivent être répartis à travers toutes les écoles de ce lieu afin de respecter la contrainte (4.3) tel que

$$\sum_{\text{école } e \in \text{ noeud } j} T_{i,j}^e = T_{i,j}^{\text{scolaire}}$$

Cependant, avant de chercher à réaliser cette répartition, les coefficients $F_{i,j}$ et $K_{i,j}$ $\forall\ 1 \leq j \leq n_{noeud_{ecole}},\ \forall\ 1 \leq i \leq n_{domicile}$ doivent être déterminés.

Le coefficient de friction $F_{i,j}$ est une fonction qui peut dépendre du temps de parcours, de la distance ou même du coût du transport entre les noeuds i et j. Ce coefficient apporte une notion de distance dans le modèle. Par exemple, d'après la référence [23], une étude à été réalisée avec ce modèle exprimé par l'équation (4.6) concernant les déplacements dans la ville de Lincoln-Lancester. La fonction calculant le coefficient de friction dans cette étude peut être exprimée de la manière suivante :

soient : $duree_{i,j}$ la durée du trajet reliant le noeud i au noeud j
A, B, C constantes

$$F_{i,j} = A * (duree_{i,j}^{B} * e^{duree_{i,j} * C})$$

Toujours, en concordance avec le site de référence, il existe plusieurs valeurs pour les constantes A, B et C en fonction du motif du déplacement.

D'après [33], les coefficients de friction de l'équation (4.6) peuvent être décrits par une des formes suivantes :
$\forall\ 1 \leq j \leq n_{noeud_{ecole}},\ \forall\ 1 \leq i \leq n_{domicile}$
- $F_{i,j}(d_{i,j}) = \exp(-Gd_{i,j})$
- $F_{i,j}(d_{i,j)} = d_{i,j}^{-n}$
- $F_{i,j}(d_{i,j}) = d_{i,j}^{n} * \exp(-Gd_{i,j})$

où n, G sont des coefficients à calibrer avec les données réelles. Dans chaque équation, la distance peut être remplacée par la durée ou par le coût du voyage.

Nous remarquons donc qu'il faut connaitre la durée ou la distance de chaque trajet pour construire la matrice O-D. Cependant, cette contrainte est un inconvénient majeur pour le temps d'exécution du programme. Supposons qu'on veuille distribuer uniquement les écoles primaires aux étudiants de Namur, alors les variables $n_{domicile}$ et $n_{noeud_{ecole}}$ valent réciproquement 3323 noeuds pour 48862 étudiants et 4626 noeuds pour 5033 écoles primaires. Ce qui signifie que plus de 15 000 000 demandes de distance entre deux noeuds doivent être formulées pour connaitre la valeur des coefficients de friction. Or chaque demande a un prix assez élevé car la fonction « getDistance » parcourt tous les noeuds à partir du domicile jusqu'à trouver celui de la destination. De plus, le nombre de demandes s'accroitra énormément lors de l'assignation d'une école à tous les étudiants de Belgique en fonction des trois types d'enseignement.

En outre, avec ce modèle, une seule itération ne suffit pas toujours. Il faudra donc recalculer les distances à chaque itération. Cela rend ce modèle encore plus handicapant pour la recherche des distances. Nous ne pouvons donc pas appliquer ce modèle si les demandes réalisées sont aussi importantes sous peine d'augmenter le temps d'exécution.

Une possibilité de contrer cet inconvénient est de créer une matrice contenant la distance entre chaque noeud avant l'exécution du programme VirtualBelgium. Cependant, vu le nombre important de noeuds dans le réseau, le calcul de cette dernière n'est pas réalisable en moins de 24 heures pour un processeur. De plus, la matrice aurait une taille plus ou moins équivalente à 262000 X 26200. Une taille aussi imposante ne serait pas adéquate du point de vue de la mémoire du programme.

Une autre raison de ne pas appliquer ce modèle est le calcul des coefficients socio-économiques. En effet, aucune donnée récoltée nous permet de calibrer ces derniers.

Conclusion

Le modèle gravitationnel donné par l'équation (4.9), ne peut être considéré comme le modèle d'assignation principalement suite à la recherche imposante des distances entre deux noeuds.

1.2. Deuxième amélioration

Il existe un autre moyen de permettre au modèle gravitationnel, exprimé par l'équation (4.5), de respecter la contrainte (4.2). D'après la référence [33], cette astuce consiste à remplacer la constante K par deux ensembles de constantes, notées $(A)_i$ et $(B)_j \forall\ i \in \{1, \cdots, n_o\}$ et $j \in \{1, \cdots, n_d\}$. La formule du modèle devient :

soient :
- n_o le nombre de noeuds dans le réseau considérés comme lieu de départ (origine du déplacement),
- n_d le nombre de noeuds dans le réseau considérés comme destination,
- O_i le nombre souhaité de trajets partant du noeud i,
- D_j le nombresouhaité de trajets arrivant au noeud j,
- $F_{i,j}$ le facteur de friction.
- A_i, B_j constantes à calibrer, dépendantes du noeud d'origine ou du noeud de destination,
  avec $A_i = \frac{1}{\sum_{j=1}^{n_d} B_j D_j F_{i,j}}$
  avec $B_j = \frac{1}{\sum_{i=1}^{n_o} A_i O_i F_{i,j}}$

$$T_{i,j} = A_i B_j \, O_i \, D_j F_{i,j} \;\forall\; i \in \{1, \cdots, n_o\} \text{ et } j \in \{1, \cdots, n_d\} \tag{4.10}$$

Grâce à ces constantes, nous pouvons affirmer que le modèle gravitationnel, exprimé avec l'équation (4.10), respecte les deux contraintes (4.1) et (4.2). En effet :

$$\begin{aligned}
\sum_{i=1}^{n_o} T_{i,j} &= \sum_{i=1}^{n_o} A_i B_j \, O_i \, D_j F_{i,j} \\
&= \sum_{i=1}^{n_o} A_i \frac{1}{\sum_{i=1}^{n_o} A_i O_i F_{i,j}} \, O_i \, D_j F_{i,j} \\
&= D_j
\end{aligned}$$

Les valeurs des constantes se déterminent en plusieurs étapes :
- Calculs de $A_i \;\forall\; i \in \{1, \cdots, n_o\}$ en admettant que $B_j = 1 \;\forall\; j \in \{1, \cdots, n_d\}$
- Calculs de $B_j \;\forall\; j \in \{1, \cdots, n_d\}$ en fonction des valeurs de $A_i \;\forall\; i \in \{1, \cdots, n_o\}$ trouvées au point précédent.

Malheureusement, ce modèle doublement contraint ne peut être utilisé dans notre cas d'étude pour la même raison que la première amélioration. En effet, la formule s'exprime toujours en fonction des coefficients de friction. Les différents commentaires et remarques exposés ci-dessus sont également valables pour ce modèle.

Conclusion

Le modèle gravitationnel donné par l'équation (4.10) ne peut être considéré comme le modèle d'assignation principalement dû à une suite de recherche imposante des distances entre deux noeuds.

2. Intervening opportunities model

Il existe également le modèle " Intervening opportunities", conçu pour construire la matrice O-D. Ce modèle ne tient pas compte directement de la distance entre deux noeuds mais se consacre à l'analyse des opportunités manquées qui peuvent également satisfaire les envies du voyageur. Il est donc basé sur la probabilité de choisir un noeud pour une activité tout en connaissant les noeuds intermédiaires liés à cette dernière depuis le lieu d'origine. D'après [33], ce modèle a été développé par Schneider en 1959 mais l'idée principale provient de Stouffer.

Le principe du modèle est expliqué ci-dessous ainsi que la manière de calculer $T_{i,j}\ \forall\ 1 \leq i \leq n_o,\ \forall\ 1 \leq j \leq n_d$.

1. Soit i, un noeud origine quelconque dans le réseau et j, un noeud destination.

   Exemple :
   En concordance avec la figure 4.6, le noeud "$Residence_1$" est choisi. L'individu, habitant à cet endroit souhaite se déplacer pour aller dans un magasin.

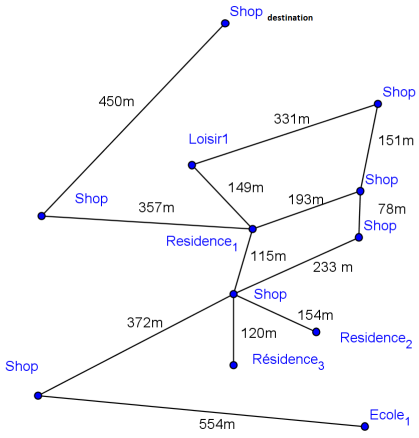


FIGURE 4.6 – Exemple de réseau pour le modèle « Intervening opportunities »

2. Toutes les destinations possibles, à partir de ce noeud i, qui conviennent aux attentes du voyageur, sont listées puis numérotées en fonction de leur distance avec le noeud d'origine. Par exemple, si le noeud j, se retrouve à la $m^{\text{ième}}$ position de cette liste, alors il existe $m-1$ autres destinations qui peuvent également contenter le voyageur mais qui sont plus proches du noeud i que le noeud j. Ces autres destinations, non retenues entre les noeuds i et j, sont appelées les « Intervening opportunities », les

opportunités manquées.

Le modèle ne tient pas compte de la direction des trajets. En d'autres mots, les opportunités manquées sont déterminées juste en fonction de la distance à l'origine i même si elles sont dans la direction opposée au noeud j.

Dans l'exemple, on souhaite calculer $T_{Residence_1,Shop\_destination}$. Il faut donc que les magasins qui sont à une plus petite distance de la résidence que le "Shop_destination", soient numérotés. Cette numérotation est visible sur la figure 4.7.

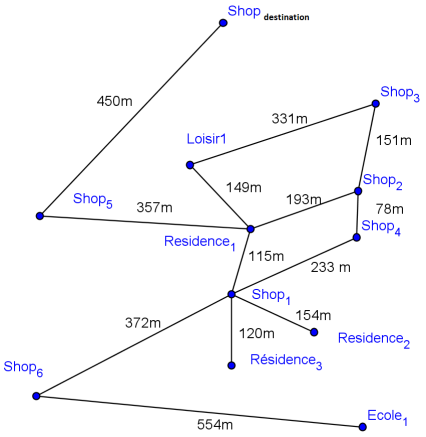


FIGURE 4.7 – Exemple de réseau pour le modèle « Intervening opportunities » avec numérotation

3. Le nombre de déplacements entre le noeud i et le noeud j est déterminé par la formule suivante :

$$T_{ij} = O_i \frac{e^{LD^{\mathbf{cum}}_{i,j-1}} - e^{-LD^{\mathbf{cum}}_{i,j}}}{1 - e^{-LD^{\mathbf{cum}}_{i,j}}} \forall\ i \in \{1, \cdots, n_o\} \text{ et } j \in \{1, \cdots, n_d\}$$

avec :
- n_o le nombre de noeuds dans le réseau considérés comme lieu de départ (origine du déplacement)
- n_d le nombre de noeuds dans le réseau considérés comme destination.
- O_i le nombre souhaité de trajets partant du noeud i
- D_j le nombre souhaité de trajets arrivant au noeud j
- $D^{\text{cum}}_{o,d}$ l'attractivité cumulée
  le nombre de trajets dont la destination est soit une opportunité manquée, soit le noeud d, toujours par rapport à l'origine o
  tel que $D^{\text{cum}}_{o,d} = D^{\text{cum}}_{o,d-1} + D_d$
- L paramètre à calibrer.

Le paramètre L est la probabilité que le voyageur soit satisfait avec une opportunité aléatoire. Ce dernier est donc un paramètre à calibrer en fonction de données récoltées sur les déplacements. L peut être une constante d'après la référence [40] si on suppose que son calcul est réalisé indépendamment du noeud d'origine.

En ce qui concerne la valeur de la probabilité, il y a deux interprétations possibles :

- Si L a une petite valeur, donc proche de 0, alors la distribution des déplacements ne se fera pas toujours localement.
- Si L a une grande valeur, donc proche de 1, alors la distribution des déplacements partant d'une origine sera assez confinée autour de cette dernière.

Remarques :

La distance n'est pas explicitement utilisée dans la formule mais elle y est implicitement. Cela met en évidence l'opinion de certains chercheurs qui supposent que les voyageurs ne se préoccupent pas de la distance parcourue pour atteindre leur objectif mais plutôt des différentes opportunités qui se présentent à eux.

Le paramètre L doit être calibré en fonction de données réelles. Malheureusement, aucune donnée en notre possession sur le transport ne permet de déterminer la valeur de cette probabilité.

De plus, ce modèle demande une recherche importante de données pour calculer la valeur de chaque $T_{i,j}$. En effet, lorsqu'on veut calculer $T_{i,j}$, il faut connaitre tous les noeuds qui sont à une plus petite distance que le noeud j du noeud i. En outre, pour choisir une école à un étudiant du noeud i, il faudra calculer les $T_{i,\text{noeud}}$ pour tous les noeuds abritant au moins une école. Par exemple, si l'étudiant ne possède pas de diplôme, on doit lui assigner une école primaire. Cependant, il existe 5033 établissements primaires en Belgique. Ce qui signifie qu'il faut déterminer 5033 fois la valeur de $T_{i,\text{ noeud}}$.

Conclusion

Le modèle " Intervening opportunities" ne peut être considéré comme le modèle d'assignation principalement en raison du manque de données pour la calibration du paramètre L mais également en raison d'une recherche importante des écoles virtuelles à chaque assignation.

4.3.4.2 Modèles avec distance scolaire parcourue fixée

Comme nous venons de le constater, un inconvénient majeur est relevé dans les deux modèles détaillés ci-dessous : ils ne sont pas adaptés aux grandes bases de données. En effet, d'un côté, il y a une trop grande demande pour le calcul des distances et de l'autre côté, une imposante recherche parmi les écoles « manquées » pour chaque calcul $T_{i,\text{noeud}}$ entre un noeud et le domicile i de l'étudiant.

Afin d'optimiser le temps d'exécution et éviter des recherches de données trop importantes, un compromis doit être généré. En effet, une solution consiste, tout d'abord, à fixer la distante entre le domicile et l'école. Cette méthode permet de réduire le choix parmi les écoles et également d'éviter des calculs de distance entre deux noeuds. La distance est donc calculée indépendamment et antérieurement au modèle d'assignation. Ensuite, il faut s'inspirer des modèles vus précédemment pour construire le modèle souhaité.

1. Distribution liée à la distance

Comme expliqué dans la partie concernant l'hypothèse de domiciliation de la section 4.3.3, J. Barthelemy, P. Toint et E. Cornelis ont développé une distribution permettant de connaitre les distances effectuées par les étudiants en respectant celles parcourues dans la réalité, plus précisément, durant l'année de récolte de l'enquête MOBEL. Cette distribution ne convient plus à cette étude. En effet, elle est calculée à partir de tous les trajets dont la destination est une école. Or nous avons émis l'hypothèse dans ce travail, que les distances doivent être uniquement parcourues entre le domicile et l'école. Cependant, ce concept de distribution peut être utilisé pour fixer les distances dans notre étude.

Une nouvelle distribution liée uniquement aux distances entre le domicile et l'école est donc créée. Cette dernière suit une loi de probabilité log-normale dont les paramètres sont présents dans la table 4.4. Nous pouvons également dire que la distribution suit une mixture de loi log-normale à une composante. Cette mixture est définie dans la section 2.2.2.1

Distance	Domicile-École
Loi de probabilité	Log-normale
μ	8.0259
σ^2	1.8792
Maximum (mètres)	140000
Minimum (mètres)	10

TABLE 4.4 – Informations concernant la distribution Domicile-École

Pour vérifier cette approximation, différentes démarches et différents tests ont été réalisés. Ces derniers sont détaillés dans la section 5.1.1 du chapitre suivant. Au final, les tests sont acceptés et les déplacements entre le domicile et l'école peuvent être modélisés par une loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.8792$

2. Détermination de la valeur de la distance:

Pour connaitre la distance réalisée par l'étudiant pour atteindre son école, il suffit de générer une distance grâce générateur de nombres pseudo-aléatoires suivant la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.8792$.

3. Méthode de la fonction de répartition:

Pour générer un ensemble de nombres pseudo-aléatoires suivant une fonction de répartition F empirique, nous pouvons utiliser la méthode de la fonction de répartition définie ci-dessous. Ses différentes étapes sont reprises ci-dessous :

- Acquisition d'un nombre entre zéro et un, de manière uniformément aléatoire [9]. Notons le u.

- Recherche de tous les points de la fonction F de répartition dont leur abscisse appartient à l'ensemble $\{ y \,|\, F(y) \geq u\}$.

- Prise de la plus petite abscisse de cet ensemble pour déterminer la valeur de la variable.

Définition 4.1. ***La méthode de la fonction de répartition :***
Soient F(x) une fonction de répartition d'une variable aléatoire réelle X, définie sur [a,b] et U une variable aléatoire suivant une loi uniforme sur [0,1],
l'inverse de cette fonction est définie par $F^{-1}(u)$.

$$F^{-1}(u) = inf \{ y \in [a,b] \,|\, F(y) \geq u\}$$

La variable aléatoire X peut être trouvée grâce à l'expression suivante :

$$X = F^{-1}(U)$$

9. Un tirage uniformément aléatoire sur un ensemble signifie que tous les éléments de cet ensemble ont les mêmes chances d'être choisis.

Afin de mieux comprendre l'utilisation de la méthode de la fonction de répartition, un exemple est mis à votre disposition. Prenons un ensemble de points d'une fonction de répartition F d'une loi log_normale, représenté par la figure 4.8. La première étape à réaliser est d'obtenir un nombre aléatoire entre 0 et 1, de manière uniforme.

Sachant que le nombre aléatoire vaut 0.3, l'ensemble $\{\, y \in [0, 15000] \,|\, F(y) \geq 0.3\}$, nommé E, est formé par les points suivants : $\{\, 200, 300, 400, 500, \cdots, 15000\}$.

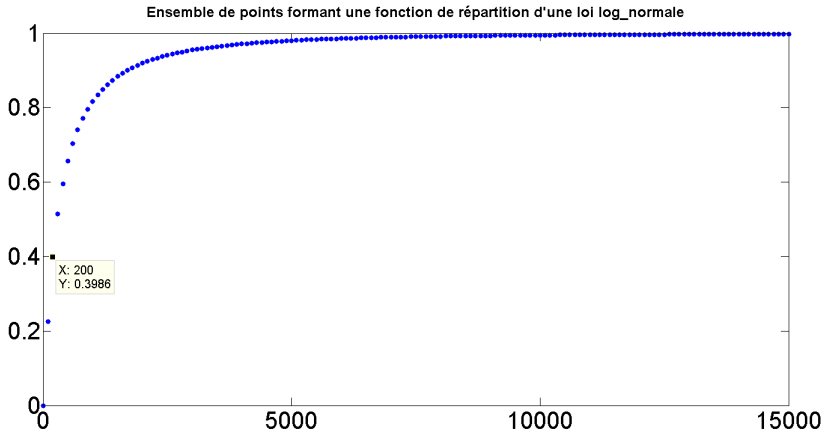


FIGURE 4.8 – Exemple d'une fonction de répartition empirique

Le nombre souhaité est calculé en prenant la borne inférieure de cet ensemble E. Par définition, en mathématique, la borne inférieure d'un ensemble A est le maximum de ses minorants. En d'autres termes, c'est le plus grand réel M, qui respecte la condition suivante :

$$\forall x \in A \quad \mathrm{M} \leq x$$

. En conclusion, le nombre souhaité vaut 200 car l'ensemble des minorants de l'ensemble E est égal à $]-\infty, 200]$.

4. Simplification de la contrainte (4.4):

Dans la section 4.3.3, nous avons posé les contraintes que le modèle d'assignation doit respecter. Il y a, entre autres, la contrainte qui dit que « Le modèle doit respecter le nombre souhaité d'étudiants dans chaque école virtuelle ». Ensuite nous avons traduit cette contrainte avec les notations du modèle Trip Distribution. Cette dernière est représentée par l'équivalence (4.3).

Dans cette partie du travail, nous n'utilisons plus les notations liées aux modèles « Trip Distribution » mais nous traduisons juste la contrainte comme ceci :

Soient : quota_j^e le quota de l'école e du noeud j
$NbEtudiant_j^e$ le nombre d'étudiants inscrits à l'école e du noeud j

À la fin de l'application du modèle d'assignation, nous devons obtenir l'équivalence suivante :

$\forall\ j$ tel que le noeud j abrite au moins une école
$\forall\ e$ une école du noeud j

$$NbEtudiant_j^e = \text{quota}_j^e$$

Dans la suite du travail, nous n'écrivons plus la double boucle $\forall\ j$ et $\forall\ e$ (celle sur les noeuds ayant au moins une école et celle sur les écoles de ces lieux) mais une seule sur les écoles sans tenir compte du lieu. Ceci est un abus de langage mais cela permet de simplifier les notations dans la suite du travail. Nous pouvons faire la même réflexion, avec la variable quota_j^e représentant le quota d'étudiants pour l'école e du noeud j.

En conclusion, la 4ième contrainte de la section 4.3.3 devient :

soient : quota_e le quota d'étudiants, c'est-à-dire,
le nombre souhaité d'étudiants dans l'école e
$NbEtudiant_e$ le nombre d'étudiants virtuels inscrits à l'école e.

$\forall\ e$ une école virtuelle

$$NbEtudiant_e = \text{quota}_e \tag{4.11}$$

Nous pouvons mettre en évidence que les modèles ci-dessous sont créés pour être des modèles simplement contraints par rapport à la destination afin de respecter la quatrième contrainte de la section 4.3.3, aussi exprimée par l'équation (4.3). Or les modèles construits vont être doublement contraints sans y être forcé. En effet, cela revient à prouver que tous les étudiants sont assignés à un déplacement scolaire et donc à une école. Ceci sera donc toujours vrai par construction.

5. Nouveau contexte du modèle d'assignation:

La distance que les étudiants virtuels doivent parcourir depuis leur domicile pour atteindre l'école, est donc connue. Il faut donc maintenant créer un modèle inspiré des précédents qui tienne compte de la distance fixée par la distribution Domicile-École.

En conclusion, le modèle doit respecter les éléments suivants :

- Le modèle est applicable à tous les étudiants de VirtualBelgium.
- Le type d'établissement choisi doit être déterminé en fonction du dernier diplôme obtenu.
- Les distances entre le domicile et l'école de l'individu doivent suivre la distribution construite au point « 1. Distributions liés à la distance : » de cette section.
- soient : quota_e le quota d'étudiants, c'est-à-dire, le nombre souhaité d'étudiants dans l'école e
  $NbEtudiant_e$ le nombre d'étudiants déjà inscrits à l'école e.

$$\forall\ e \text{ une école virtuelle } NbEtudiant_e = \text{quota}_e \tag{4.12}$$

Plusieurs modèles ont été testés, puis améliorés tout au long de ce travail. Cependant, ces derniers conservent l'idée principale de la proportionnalité, c'est à dire que chaque école possède une attractivité qui va dépendre du nombre d'étudiants qui peuvent s'y inscrire et des possibilités d'inscription dans les écoles environnantes.

6. Premier modèle testé:

6.1. Modèle

Les différentes étapes de ce modèle sont expliquées ci-dessous :

Soit un étudiant virtuel et soit le noeud o, son domicile :

1. Détermination du type de l'école dans laquelle l'individu doit être assigné en suivant l'explication donnée dans la section 4.3.2. Notons ce type, « type ».
2. Évaluation de la distance à laquelle doit se trouver approximativement l'école de l'étudiant, notée d, à l'aide d'un générateur de nombres pseudo-aléatoires suivant la

loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.8792$.

3. Création d'un ensemble de noeuds qui sont à une distance d du noeud o, à un epsilon près, égal à 250 mètres. En d'autres mots, si la distance est égale à 1000m alors les noeuds de l'ensemble créé seront à une distance entre 750 m et 12500 m du domicile (voir la figure 4.9).

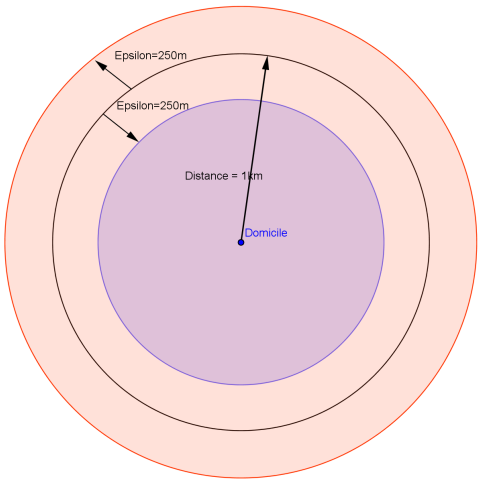


FIGURE 4.9 – Signification de la variable « epsilon », égal à 250m avec une distance choisie de 1 km. L'aire rouge [10]représente la zone où les noeuds sont sélectionnés lors de l'étape 3.

4. Sélection des noeuds représentant une école du même type que la variable « type » dans cet ensemble. Si un noeud abrite deux écoles du même type, alors il sera sélectionné deux fois et sera distingué par les caractéristiques de ces dernières.

   Cette liste contient $n_{\text{noeud}_{\text{selectionné}}}$ noeuds. Les écoles abritées par ces noeuds sont classées. Il y a n_E écoles avec quota_{e_i}, le quota de la $i^{\text{ième}}$ école de la liste, notée e_i, $\forall\ 1 \leq i \leq n_E$.

   Si l'ensemble des noeuds représentant une école est vide alors on retourne à l'étape n°2 où on régénère une nouvelle distance, sinon on passe à l'étape suivante.

5. Déduction de l'école et donc du noeud où l'étudiant effectuera ses études en deux étapes :

   (a) Calcul de la probabilité pour chaque école de la liste d'être choisie par l'étudiant.

10. Comme expliqué pour le graphe 2.1, la distance et l'epsilon sont illustrés en vol d'oiseau. Or il faut garder en mémoire que les distances en VirtualBelgium sont calculées à partir de la longueur des routes.

Ces probabilités sont déterminées en fonction de la formule suivante :

$$P_i = \frac{\text{quota}_{e_i}}{\sum_{j=1}^{n_E} \text{quota}_{e_j}} \quad \forall\ 1 \leq i \leq n_E$$

(b) Utilisation de la méthode de la fonction de répartition.

Cette dernière permet de sélectionner une école comme expliqué dans le point « 3. Méthode de la fonction de répartition ». Pour définir la fonction de répartition utilisée, il faut se baser sur les probabilités calculées précédemment.

Tout d'abord, formons une fonction, notée f, dite en escalier à partir des probabilités cumulatives telle que :

$$\begin{aligned} f(x) &= 0 && \forall\ x \in\ ]-\infty, 1[\\ f(x) &= \sum_{j=1}^{i} P_j && \forall\ x \in [i, i+1[\quad \forall\ i \in \{1, \cdots, n_E - 1\} \\ f(x) &= 1 && \forall\ x \in [n_E, \infty[\end{aligned}$$

Il reste à vérifier que cette fonction est une fonction de répartition empirique. Pour effectuer ce test, la définition d'une fonction de répartition, extraite de la référence [19], est introduite :

Définition 4.2. *Une fonction F est une fonction de répartition si*

i. *F est croissante*

ii. *F est continue à droite*

iii. *F admet une limite à gauche en chaque point*

iv. $\lim_{x \to -\infty} F(x) = 0$

v. $\lim_{x \to \infty} F(x) = 1$

La fonction f, définie plus tôt respecte les cinq conditions. En effet,

i. f est croissante puisque nous utilisons les probabilités cumulatives pour la créer,

ii. f est continue à droite par construction,

iii. f admet une limite à gauche à chaque point par construction,

iv. $\lim_{x \to -\infty} f(x) = 0$ car $f(x) = 0\ \forall\ x \in\ ]-\infty, 1[$

v. $\lim_{x \to \infty} f(x) = 1$ car $f(x) = 1 \ \forall \ x \in [n_E, \infty[$

Nous pouvons donc utiliser la méthode de la fonction de répartition sur la fonction f créée ci-dessous. Cette méthode choisira l'indice de l'école qui sera assignée à l'étudiant. Pour rappel, cette méthode s'effectue en trois étapes :

i. Acquisition d'un nombre entre 0 et 1, de manière uniformément aléatoire. Notons le u.

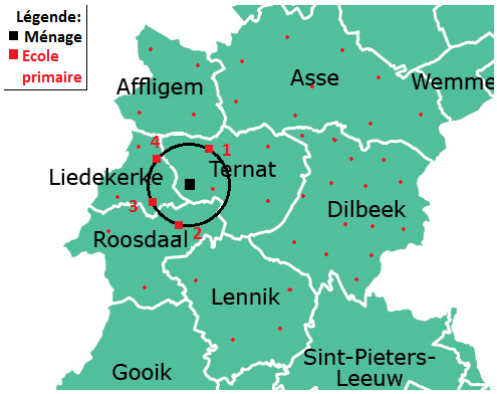
ii. Sélection de tous les points de la fonction f dont l'abscisse z appartient à l'ensemble $\{ z \mid f(z) \geq u \}$.

iii. Recherche de la borne inférieure de cet ensemble pour déterminer l'indice de l'école choisie.

6. Attribution de l'école à l'individu.

6.2. Exemple

Afin de faciliter la compréhension, un exemple est mis à votre disposition avec la figure 4.10.



:

FIGURE 4.10 – Exemple pour le modèle d'assignation

Soit un individu habitant un noeud dans la commune de Ternat (voir la légende de la figure 4.10). Supposons également qu'il ne possède pas de diplôme et que la distance tirée est égale à 20km.

Avec ces données, nous pouvons déjà affirmer que l'individu sera assigné à une école primaire. D'après l'exemple, il existe seulement quatre écoles primaires se trouvant à 20km du domicile, à 250 m près. Elles sont énumérées dans la table 4.5.

Noeuds sélectionnées	Nombres souhaité d'étudiants
1	235
2	300
3	327
4	67

TABLE 4.5 – Choix des écoles pour l'exemple

Après avoir défini l'ensemble des différents choix possibles, il faut déterminer les probabilités. Elles sont reprisent dans la table 4.6. L'ordre des écoles n'a pas d'influence dans ce modèle.

Noeuds sélectionnées	Probabilités
1	0.253
2	0.323
3	0.352
4	0.072

TABLE 4.6 – Probabilités de l'exemple

Maintenant que nous avons les probabilités, nous pouvons construire la fonction de répartition empirique. Elle est représentée à la figure 4.11.

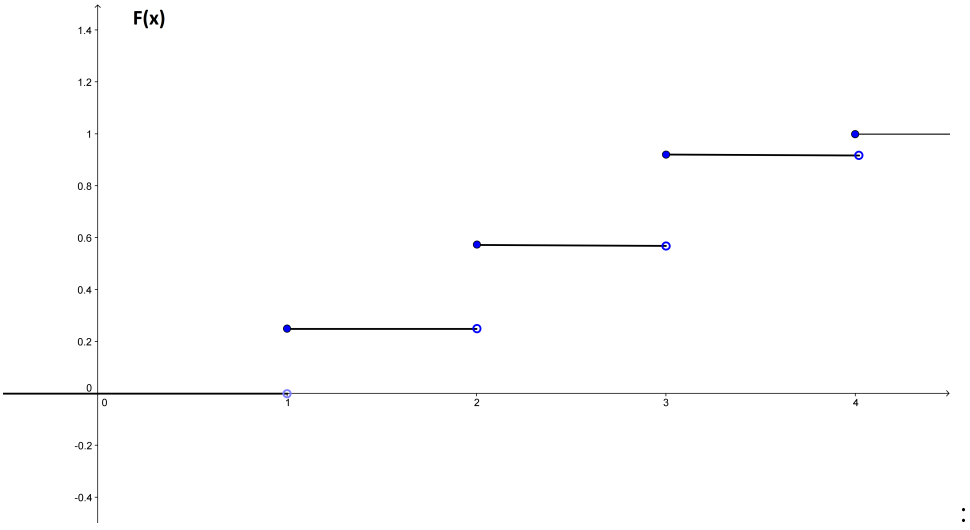


FIGURE 4.11 – Fonction de répartition empirique pour l'exemple

La dernière étape consiste à tirer un nombre aléatoire entre 0 et 1. Supposons que ce nombre soit égal à 0.4, alors la méthode de la fonction de répartition assignera l'école n°2 à l'individu.

6.3. Relation avec le modèle gravitationnel

Le choix entre les écoles abritées par les noeuds sélectionnés à l'étape 4 est basé sur le principe de proportionnalité du modèle gravitationnel. Quelques modifications et quelques adaptations ont été réalisées.

- Premièrement, le modèle gravitationnel, exprimé par le formule (4.9), s'intéresse à tous les noeuds du réseau. Cependant, dans notre modèle d'assignation, on réalise un choix entre des écoles à la même distance du domicile, à un epsilon près. Nous pouvons donc supposer que les coefficients de friction $F_{\text{domicile,noeud}_{\text{selectionné}}}$ sont identiques pour tous les noeuds sélectionnés.

- Nous n'avons aucune donnée pour calibrer les coefficients $K_{\text{domicile,noeud}_{\text{selectionné}}}$ pour les noeuds sélectionnés. On va donc émettre l'hypothèse que les coefficients socio-économiques sont identiques pour tous les noeuds.

- En considérant ces deux hypothèses, nous obtenons un modèle gravitationnel permettant la sélection d'un noeud. Sa formule est décrite ci-dessous :

soient :
- $n_{domicile}$ le nombre de noeuds considérés comme le domicile d'un étudiant,
- $n_{\text{noeud}_{\text{selectionné}}}$ le nombre de noeuds sélectionnés à l'étape 4,
- O_i^{scolaire} le nombre souhaité de trajets scolaires partant du noeud i,
- D_j^{scolaire} le nombre souhaité de trajets scolaires arrivant au noeud j,

$$\forall\ i \in \{1, \cdots, n_{domicile}\} \text{ et } j \in \{1, \cdots, n_{\text{noeud}_{\text{selectionné}}}\}$$

$$T_{i,j}^{\text{scolaire}} = O_i^{\text{scolaire}} \frac{D_j^{\text{scolaire}}}{\sum_{k=1}^{n_{\text{noeud}_{\text{selectionné}}}} D_k^{\text{scolaire}}} \tag{4.13}$$

- Pour respecter la contrainte (4.3), nous avons adapté la formule (4.13) afin qu'elle puisse être utilisée pour déterminer une école dans l'ensemble des écoles abritées par les noeuds sélectionnés en fonction d'un étudiant. Cette formule ne calculera plus le nombre de trajets mais une probabilité. On s'intéresse maintenant à un étudiant et à son ensemble de choix pour les écoles. La valeur de O_i sera égale à 1 pour

représenter l'écolier et la valeur de D_j pour une école est égale au quota de cette dernière (voir le point « 2. Données » de la section 4.3.3).

Soient :
- n_E le nombre d'écoles abritées par les noeuds sélectionnés,
- e_i la $i^{\text{ième}}$ école de la liste des noeuds selectionnés
- D_{e_i} le quota associé à la $i^{\text{ième}}$ école de la liste des noeuds selectionnés., on peut aussi l'écrire quota_{e_i}
- $T_{\text{etudiant},i}$ la probabilité que la $i^{\text{ième}}$ école soit choisie par l'étudiant,

$$\forall\ 1 \leq i \leq n_E$$

$$T_{\text{etudiant},i} = \frac{D_{e_i}}{\sum_{k=1}^{n_E} D_{e_k}} \tag{4.14}$$

6.4. Implémentation

L'implémentation de ce modèle est expliquée dans le chapitre suivant. Il faut toutefois savoir que le programme ressort un fichier par processeur, contenant les informations sur les écoles du réseau de ce dernier dont le nombre d'individus virtuels inscrits dans ces dernières. Ces fichiers sont stockés dans le répertoire VirtualBelgium/Output. Le nombre de ces derniers correspond au nombre de processeurs utilisés pour exécuter le programme.

Un fichier est créé par processeur car comme nous l'avons dit, dans la section 1.4, chacun de ces derniers possèdent un réseau différent et donc des écoles différentes. Nous pourrions effectuer la synchronisation des écoles dans le programme VirtualBelgium afin de ressortir un unique fichier reprenant l'ensemble des individus inscrits dans chaque école. Cependant, la synchronisation dans la programmation parallèle n'est pas facile à mettre en place et engendre des changements importants dans le programme. De plus, un arrêt complet de chaque processeur est nécessaire à chaque synchronisation ce qui augmente le temps d'exécution du programme. De ce fait, un petit programme écrit dans le logiciel MATLAB a été réalisé afin de rassembler tous les fichiers et permettre de connaitre le nombre d'étudiants inscrits dans chaque école virtuelle indépendamment du processus.

La première colonne de ces fichiers reprend l'identifiant des écoles. Ensuite, l'identifiant des noeuds qui abritent les écoles, le code INS de ces noeuds, le numéro de FASE des écoles. Pour finir, nous pouvons trouver le nombre souhaité d'étudiants et le nombre d'étudiants virtuels inscrits dans ces écoles. Notre intention sera principalement portée sur ces deux dernières colonnes afin de vérifier la validité du modèle et sa précision.

6.5. Résultat

Lors du premier essai du programme amélioré, le modèle d'assignation est seulement appliqué aux étudiants de la commune de Namur. En effet, la population virtuelle été réduite aux namurois afin de permettre au programme de s'exécuter dans un laps de temps raisonnable avec peu de processeurs. Plus on demande de processeurs lors de l'envoi du job sur un cluster et plus le risque est grand d'être mis en fin de liste d'attente.

Cependant, les résultats contenant les écoles namuroises ne sont pas concluants et ne respectent pas la contrainte 4.12. Effectivement, les étudiants namurois ne remplissent pas forcement les écoles de leur commune et se déplacent parfois dans d'autres communes. De plus, avec le modèle d'assignation utilisé, la distance entre le domicile et l'école peut aller jusqu'à 140 000m.

Prenons un étudiant namurois et son ensemble d'écoles sélectionnées par le modèle d'assignation. Ces dernières ne se trouvent pas forcément toutes dans la commune de Namur. Or dans la probabilité calculée pour chaque école, seul le quota d'étudiants par école virtuelle est pris en compte. La commune de l'étudiant n'influence pas le modèle. On peut donc en conclure, que cet étudiant peut être assigné à une école autre que namuroise.

Dès lors, si on veut vérifier si le modèle respecte la contrainte 4.12 pour les écoles namuroises, la population utilisée pour tester le modèle doit être élargie.

Un test est donc réalisé sur toute la population virtuelle de la Belgique. Ce test est exécuté en utilisant le Cluster de l'Université Catholique de Louvain, appelé Lemaître2 [11]. Vu que 500 processus ont été mis à disposition de ce programme, le Cluster de l'Université de Namur, appelé Hercules, ne peut être utilisé en raison d'un nombre limité de processeurs.

Durant cette opération, seuls les écoles primaires ont été testées. En d'autres mots, seuls les étudiants virtuels non diplômés ont été assignés à un établissement de l'enseignement primaire. Après une analyse du fichier créé, les écoles primaires namuroises sont davantage choisies qu'au test précédent. Cependant, la contrainte 4.12 n'est toujours pas respectée. Certains établissements ont un taux d'inscription plus important que leur quota et vice-versa, d'autres sont très en dessous. Les résultats obtenus pour les écoles namuroises sont détaillés dans la table 4.7.

Paramètre :	
Modèle	Modèle 1
Type de l'école	Primaire
Nombre de processeurs	100
Nombre d'étudiants	1727134

Identifiant	Numéro de fase	Quota $quota_e$	Nombre d'étudiants inscrits $NbEtudiant_e$
4828	5831	165	156
4829	5833	134	221
4830	5836	754	614
4831	5837	146	297
4832	5838	249	419
4833	5839	452	360
4834	5840	213	194
4835	5841	785	716
4836	5842	194	258
4837	5843	34	140

TABLE 4.7 – Résultats de certaines écoles namuroises pour le modèle 1

6.6. Conclusion

En conclusion, nous ne pouvons pas accepter ce modèle. Si on veut continuer à utiliser la proportionnalité, il va falloir ajuster ce modèle afin de mieux répartir les déplacements scolaires, en d'autres mots, les étudiants. Une première idée est d'éviter aux écoles d'accepter des étudiants au-delà de leur quota. La deuxième idée suit la même logique que la précédente mais en surveillant le nombre d'étudiants tout au long de l'application du modèle.

7. Deuxième modèle testé:

7.1. Modèle

Afin d'éviter un surplus d'étudiants dans les écoles virtuelles, ces dernières doivent refuser des étudiants lorsque leur quota est atteint. En modélisation, cela signifie que la probabilité de ces écoles d'être choisie est nulle. Les différentes étapes de ce modèle sont identiques au premier modèle détaillé au point 5.1. de cette section excepté pour l'étape

5a. En effet, les modifications des probabilités sont exposées ci-dessous.

Soit un étudiant virtuel et soit le noeud o, son domicile :

1. Détermination du type de l'école dans laquelle l'individu doit être assigné en suivant l'explication donnée dans la section 4.3.2. Notons ce type, « type ».
2. Évaluation de la distance à laquelle doit se trouver approximativement l'école de l'étudiant, notée d, à l'aide d'un générateur de nombres pseudo-aléatoires suivant la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.8792$.
3. Création d'un ensemble de noeuds qui sont à une distance d du noeud o, à un epsilon près, égal à 250 mètres. En d'autres mots, si la distance est égale à 1000m alors les noeuds de l'ensemble créé seront à une distance entre 750 m et 1250 m du domicile (voir la figure 4.9).
4. Sélection des noeuds représentant une école du même type que la variable « type » dans cet ensemble. Si un noeud abrite deux écoles du même type, alors il sera sélectionné deux fois et sera distingué par les caractéristiques de ces dernières.

   Cette liste contient $n_{\text{noeud}_{\text{selectionné}}}$ noeuds. Les écoles abritées par ces noeuds sont classées. Il y a n_E écoles avec quota_{e_i}, le quota de la $i^{\text{ième}}$ école de la liste, notée e_i, $\forall\ 1 \leq i \leq n_E$.

   Si l'ensemble des noeuds représentant une école est vide alors on retourne à l'étape n°2 où on régénère une nouvelle distance, sinon on passe à l'étape suivante.
5. Déduction de l'école et donc du noeud où l'étudiant effectuera ses études en deux étapes :
   (a) Calcul de la probabilité de chaque école de la liste d'être choisie par l'étudiant.

   Ces probabilités sont déterminées en fonction de la formule suivante :

soient :	n_E	le nombre d'écoles abritées par les noeuds selectionnés,
	e_i	la $i^{\text{ième}}$ école de la liste déterminée à l'étape précédente,
	$NbEtudiant_{e_i}$	le nombre d'étudiants inscrits dans l'école e_i,
	quota_{e_i}	le nombre souhaité d'étudiants dans l'école e_i
	P_i	la probabilité que l'étudiant choisisse l'école e_i parmi les noeuds sélectionnés à l'étape précédente.

$\forall\ 1 \leq i \leq n_E$ où e_i est une école

Si $NbEtudiant_{e_i} \neq \text{quota}_{e_i}$ alors $P_i = \frac{\text{quota}_{e_i}}{\sum_{j=1}^{n_E} \text{quota}_{e_j}}$

sinon $P_i = 0$

(b) Utilisation de la méthode de la fonction de répartition comme expliqué dans le premier modèle

6. Attribution de l'école à l'individu.

7.1. Implémentation

Comme expliqué précédemment, ce modèle est presque identique au précédent. Cependant, au point de vue de la programmation parallèle, ce modèle impose un énorme changement qui risque de faire apparaitre une différence entre les résultats théoriques attendus avec ce modèle et les résultats obtenus après l'exécution du programme.

Dans le premier modèle, nous nous préoccupions pas de synchroniser le nombre d'étudiants dans chaque école à travers les différents processeurs. En effet, chaque processeur effectuait ses opérations indépendamment des autres et donc remplissait les écoles de son réseau [11] sans prendre en considération le nombre d'étudiants déjà inscrits dans les écoles des autres processeurs. Cependant si nous voulons détecter le moment où le quota d'une école est atteint, il faut pouvoir calculer le nombre total d'étudiants inscrits dans ces écoles à travers tous les processeurs quand on le souhaite. Actuellement, cette opération n'est pas possible. Prenons un exemple avec deux processeurs dont un possède deux étudiants et l'autre trois étudiants. Ces derniers ont la possibilité d'être affectés à deux écoles :

Processeurs	Étudiants	Écoles (quota)	
P_1	S_1, S_2, S_3	E_1	2
		E_2	3
P_2	S_4, S_5	E_1	2
		E_2	3

Supposons qu'on applique le deuxième modèle d'assignation aux étudiants des processeurs. Cette opération sera effectuée de manière simultanée dans les deux processeurs et est représentée dans la table 4.8. Deux attributions renseignées sur la même ligne signifient qu'elles sont effectuées en même temps.

11. Pour rappel, nous avons vu à la section 1.4 que chaque processeur possédait son propre réseau et donc ses propres écoles

Temps	P_1					P_2				
	Étudiants	Écoles attribuées	Écoles de P_1			Étudiants	Écoles attribuées	Écoles de P_2		
T_0				Quota	Nombre d'inscrits				Quota	Nombre d'inscrits
			E_1	2	0			E_1	2	0
			E_2	3	0			E_2	3	0
T_1	S_1	E_1		Quota	Nombre d'inscrits	S_4	E_1		Quota	Nombre d'inscrits
			E_1	2	1			E_1	2	1
			E_2	3	0			E_2	3	0
T_2	S_2	E_2		Quota	Nombre d'inscrits	S_5	E_2		Quota	Nombre d'inscrits
			E_1	2	1			E_1	2	1
			E_2	3	1			E_2	3	1
T_3	S_3	E_1		Quota	Nombre d'inscrits					
			E_1	2	2					
			E_2	3	1					

TABLE 4.8 – Exemple d'application du modèle d'assignation sans synchronisation

Nous pouvons remarquer que l'étudiant S_3 est assigné à l'école E_1 alors que cette dernière était déjà complète avec un étudiant du processeur P_1 et un étudiant du processeur P_2. Dans le premier modèle, cela ne posait pas de problème vu qu'on ne regardait pas si l'école était complète. Cependant, dans le deuxième modèle, nous devons trouver une solution.

La solution à ce problème est une synchronisation entre les processeurs pendant l'application du modèle et ainsi permettre au processeur P_1 de savoir que l'école E_1 est remplie et que l'étudiant $S3$ doit aller ailleurs. Si nous n'effectuons pas cette synchronisation, nous nous retrouverons avec le premier modèle cité-ci dessus, c'est-à-dire l'assignation d'école sans prendre en compte le quota des écoles. Il faut donc changer les probabilités mais également effectuer des synchronisations pour arriver à implémenter le modèle 2.

Pour effectuer cette synchronisation, différentes implémentations ont été réalisées. Ces dernières sont détaillées dans le prochain chapitre. Au final, une seule peut être utilisée dans le programme VirtualBelgium et fonctionne en programmation parallèle avec 500 processeurs. Afin d'interpréter les résultats obtenus par ce modèle, il faut comprendre globalement l'implémentation qui est derrière la synchronisation car cette dernière contraint les résultats du second modèle a être un peu différents de ceux théoriques demandés.

La synchronisation des écoles consiste donc à mettre en commun le nombre d'étudiants inscrits dans ces dernières pour chaque processeur et ainsi obtenir les nombres totaux d'inscrits par école au cours de l'évolution de la répartition. Elle se réalise après l'ensemble des assignations d'une école à un étudiant pour chaque processeur. Ces derniers étant en attente après l'assignation et libérés après la synchronisation. Il y aura donc toujours un petite marge d'erreur lorsqu'on détecte si l'école est complète. En effet, comme les processeurs fonctionnent simultanément, un processeur, assignant un étudiant à une école E, ne peut pas savoir qu'un deuxième assigne un étudiant à cette même école en même temps. En d'autres termes, s'il ne reste plus qu'une seule place dans une école alors deux étudiants de deux processeurs différents peuvent encore y être assignés en même temps. Nous ne pouvons malheureusement pas empêcher ce surplus.

Cela signifie donc que le modèle doit légèrement être changé dans l'implémentation pour mieux détecter le surplus :

soient :	n_E	le nombre d'écoles abritées par les noeuds sélectionnés,
	e_i	la $i^{\text{ième}}$ école de la liste déterminée à l'étape précédente,
	$NbEtudiant^{e_i}$	le nombre d'étudiants inscrits dans l'école e_i,
	D^{e_i}	le nombre souhaité d'étudiants dans l'école e_i
	P_i	la probabilité que l'étudiant choisisse l'école e_i parmi les noeuds sélectionnés à l'étape précédente.

$\forall\ 1 \leq i \leq n_E$ où e_i est une école

Si $NbEtudiant^{e_i} < D^{e_i}$ alors $P_i = \frac{D^{e_i}}{\sum_{j=1}^{n_E} D^{e_j}}$

sinon $P_i = 0$

Nous pouvons donc maintenant refaire la petite application avec notre exemple. Cette dernière est représentée à la table 4.9.

<table>
<tr><td>Temps</td><td colspan="5">P_1</td><td colspan="5">P_2</td></tr>
<tr><td></td><td>Étudiants</td><td>Écoles attribuées</td><td colspan="3">Écoles de P_1</td><td>Étudiants</td><td>Écoles attribuées</td><td colspan="3">Écoles de P_2</td></tr>
<tr><td rowspan="3">T_0</td><td rowspan="3"></td><td rowspan="3"></td><td></td><td>Quota</td><td>Nombre temporaire d'inscrits</td><td rowspan="3"></td><td rowspan="3"></td><td></td><td>Quota</td><td>Nombre temporaire d'inscrits</td></tr>
<tr><td>E_1</td><td>2</td><td>0</td><td>E_1</td><td>2</td><td>0</td></tr>
<tr><td>E_2</td><td>3</td><td>0</td><td>E_2</td><td>3</td><td>0</td></tr>
<tr><td rowspan="3">T_1</td><td rowspan="3">S_1</td><td rowspan="3">E_1</td><td></td><td>Quota</td><td>Nombre temporaire d'inscrits</td><td rowspan="3">S_4</td><td rowspan="3">E_1</td><td></td><td>Quota</td><td>Nombre temporaire d'inscrits</td></tr>
<tr><td>E_1</td><td>2</td><td>1</td><td>E_1</td><td>2</td><td>1</td></tr>
<tr><td>E_2</td><td>3</td><td>0</td><td>E_2</td><td>3</td><td>0</td></tr>
<tr><td>Synchro 1</td><td colspan="5"></td><td colspan="5"><table><tr><td></td><td>Quota</td><td>Nombre total d'inscrits</td></tr><tr><td>E_1</td><td>2</td><td>2</td></tr><tr><td>E_2</td><td>3</td><td>0</td></tr></table></td></tr>
<tr><td rowspan="3">T_2</td><td rowspan="3">S_2</td><td rowspan="3">E_2</td><td></td><td>Quota</td><td>Nombre temporaire d'inscrits</td><td rowspan="3">S_5</td><td rowspan="3">E_2</td><td></td><td>Quota</td><td>Nombre temporaire d'inscrits</td></tr>
<tr><td>E_1</td><td>2</td><td>2</td><td>E_1</td><td>2</td><td>2</td></tr>
<tr><td>E_2</td><td>3</td><td>1</td><td>E_2</td><td>3</td><td>1</td></tr>
<tr><td>Synchro 2</td><td colspan="5"></td><td colspan="5"><table><tr><td></td><td>Quota</td><td>Nombre total d'inscrits</td></tr><tr><td>E_1</td><td>2</td><td>2</td></tr><tr><td>E_2</td><td>3</td><td>2</td></tr></table></td></tr>
</table>

TABLE 4.9 – Exemple d'application du modèle d'assignation avec synchronisation

Comme une synchronisation demande que l'ensemble des processeurs aient assigné une école à un étudiant, nous arrivons dans notre exemple à un problème pour l'étudiant S_3. En effet, le processeur P_2 n'ayant plus d'étudiants à assigner, il effectue d'autres tâches et il ne répond donc plus aux conditions nécessaires pour la synchronisation et cela engendre une erreur.

Cette situation se présentera couramment étant donné que chaque processeur possède un nombre différent d'individus et donc d'étudiants malgré le fait d'avoir plus ou moins le même nombre de ménages en raison de leur composition. Une solution serait d'avoir par processeur, le même nombre d'étudiants, ce qui est impossible. Dès lors, nous devons fixer, un nombre de synchronisations au préalable.

Nous décidons également par processeur, d'autoriser autant d'assignations d'une école entre chaque synchronisation que le résultat de la division entière du nombre d'étudiants du processeur par le nombre souhaité de synchronisations. Cependant comme le nombre d'étudiants par processeur n'est pas forcément un multiple du nombre demandé de synchronisations, nous ajoutons avant la première synchronisation, un nombre d'assignations qui correspond au reste de la division effectuée ci-dessus. De cette manière, nous obtenons le même nombre de synchronisations pour tous les processeurs mais avec un nombre différent d'assignations.

Reprenons l'exemple : en fixant à 2, le nombre de synchronisations, les nombres d'assignations par synchronisation deviennent respectivement 1 et 1 pour les processeurs P_1 et P_2 mais comme le nombre d'étudiants du processeur P_1 n'est pas un multiple de 2, le reste de la division entière vaut 1. On ajoute donc une assignation supplémentaire au nombre d'assignations de P_1, effectuées avant la première synchronisation. Le schéma de cette étape est représenté par la table 4.10.

Temps	P_1					P_2				
	Étudiants	Écoles attribuées	Écoles de P_1			Étudiants	Écoles attribuées	Écoles de P_2		
				Quota	Nombre temporaire d'inscrits				Quota	Nombre temporaire d'inscrits
T_0			E_1	2	0			E_1	2	0
			E_2	3	0			E_2	3	0
				Quota	Nombre temporaire d'inscrits				Quota	Nombre temporaire d'inscrits
T_1	S_1	E_1	E_1	2	1	S_4	E_1	E_1	2	1
			E_2	3	0			E_2	3	0
				Quota	Nombre temporaire d'inscrits					
T_2	S_2	E_2	E_1	2	1					
			E_2	3	1					
				Quota	Nombre total d'inscrits					
Synchro 1			E_1	2	2					
			E_2	3	1					
				Quota	Nombre temporaire d'inscrits				Quota	Nombre temporaire d'inscrits
T_3	S_3	E_2	E_1	2	2	S_5	E_2	E_1	2	2
			E_2	3	2			E_2	3	2
				Quota	Nombre total d'inscrits					
Synchro 2			E_1	2	2					
			E_2	3	3					

TABLE 4.10 – Exemple d'application du modèle d'assignation avec synchronisation (2)

Nous pouvons maintenant voir que grâce à la synchronisation, l'école E_1 n'est plus surpeuplée. L'étudiant S_3 a été redirigé vers une autre école.

En résumé, avec l'aide de la synchronisation, nous pouvons implémenter notre modèle, tout en sachant qu'il n'est pas parfait. Pour rappel, il reste :

- une marge d'erreur lors d'une assignation simultanée entre les processeurs.
- une marge d'erreur car le nombre de synchronisations ne correspond pas toujours au nombre d'étudiants pour chaque processeur.

L'implémentation de ce modèle met en évidence une autre difficulté : imposer aux écoles le refus d'inscription lorsque leur quota est atteint, entraîne un non respect du temps d'exécution. En effet, les derniers étudiants auxquels on assigne une école n'auront plus beaucoup de choix et devront peut être se déplacer plus loin pour aller suivre des cours. En mode opératoire, plusieurs distances vont être générées en faisant appel au générateur, avant de trouver la distance où l'ensemble des noeuds sélectionnés n'est pas vide. Cette opération risque de pendre un certain temps principalement pour le dernier étudiant vu que le nombre total de places dans les écoles virtuelles est presque identique au nombre total d'étudiants virtuels.

Pour corroborer cette hypothèse, nous avons effectué un premier test sur la population de Namur avec 50 processeurs. Le programme tournait toujours après 3 jours. Or une des conditions du modèle est de respecter le temps d'exécution. Pour contrer cette perte de temps, nous allons modifier la variable epsilon de l'étape 3. Pour rappel, cette variable permet de sélectionner les noeuds dans un anneau autour d'un cercle formé avec le domicile comme centre et la distance comme rayon (voir la figure 4.9). Actuellement, elle vaut 250 m.

Si on agrandit cette valeur, la conséquence sera l'éloignement de la distance initialement générée à l'étape 2 du modèle, pour trouver un noeud abritant une école du type souhaité. Par conséquence, en appliquant ce principe aux derniers étudiants, le temps de la recherche d'une école devrait diminuer.

Pour utiliser cette méthode de changement d'epsilon, il faut ajuster différents paramètres : tout d'abord, le moment où la valeur de la variable « epsilon » est modifiée et ensuite, sa nouvelle valeur. Nous envisagerons également de modifier plusieurs fois l'epsilon au cours de l'application du modèle afin de réduire au maximum le temps d'exécution.

Pour conclure cette section, nous pouvons mettre en évidence les trois paramètres que l'on devra calibrer pour obtenir la meilleure distribution des écoles tout en utilisant le

modèle 2 :

- Le nombre de synchronisations
- Le moment où l'epsilon est modifié
- La nouvelle valeur de l'epsilon

Comme nous effectuons des synchronisations pendant l'application du modèle, seul un fichier est suffisant pour contenir les différents résultats. Nous ne devons donc plus utiliser le fichier du logiciel MATLAB créé pour le premier modèle. Trois fichiers, contenant les informations sur les écoles virtuelles, réciproquement pour chaque type d'enseignement sont disponibles dans le répertoire OUTPUT, portant les noms :

- PrimarySchool
- SecondarySchool
- HigherSchool

7.3. Résultats

Pour tester ce modèle une première fois, nous avons choisi de n'utiliser que la population estudiantine des écoles primaires. En effet, pour tester ce modèle et calibrer les paramètres, le travail sur l'entièreté de la population VirtualBelgium est inutile. Premièrement, le programme prendrait trop de temps pour un simple test. Ensuite, le modèle d'assignation des écoles aux étudiants est le même pour les trois types d'enseignement. De plus, cela évite une utilisation abusive du cluster Lemaitre2.

Les résultats obtenus concernant certaines écoles namuroises sont visibles dans la table 4.11 ainsi que les paramètres liés au programme. Si le nombre de places restantes dans une école est positif, alors l'école n'est pas complète. Dans le cas où l'école est surpeuplée, ce nombre est négatif. La valeur de l'epsilon égale à 250 000m lors du dernier changement correspond plus ou moins au rayon du cercle qui englobe toute la Belgique et dont le centre est celui de la Belgique. Ainsi on est sûr que pour les deux dernières itérations, où il reste très peu de places, les derniers étudiants peuvent aller n'importe où en Belgique sans devoir régénérer indéfiniment une distance. Comme expliqué dans la partie implémentation du deuxième modèle, le fait de chercher les dernières écoles impose un temps considérable à cause de la manière dont le modèle est construit.

<table>
<tr><td>Paramètres</td><td></td></tr>
<tr><td>Modèle</td><td>Modèle 2</td></tr>
<tr><td>Type de l'école</td><td>Primaire</td></tr>
<tr><td>Nombre d'étudiants</td><td>1727134</td></tr>
<tr><td>Nombre de processeurs</td><td>100</td></tr>
<tr><td>Nombre de synchronisations</td><td>1200</td></tr>
<tr><td>Changement Epsilon</td><td>
<table>
<tr><th>Changement</th><th>Numéro de la synchronisation lors du changement</th><th>Epsilon (mètres)</th></tr>
<tr><td>1</td><td>1180</td><td>5 000</td></tr>
<tr><td>2</td><td>1190</td><td>10 000</td></tr>
<tr><td>3</td><td>1195</td><td>50 000</td></tr>
<tr><td>4</td><td>1198</td><td>250 000</td></tr>
</table>
</td></tr>
<tr><td>Temps d'exécution</td><td>± 15 heures</td></tr>
</table>

Identifiant	Numéro de fase	Quota $quota_e$	Nombre d'étudiants inscrits $NbEtudiant_e$	Nombre de places restantes
4828	5831	165	165	0
4829	5833	134	135	-1
4830	5836	754	754	0
4831	5837	146	146	0
4832	5838	249	251	-2
4833	5839	452	453	-1
4834	5840	213	213	0
4835	5841	785	786	-1
4836	5842	194	194	0
4837	5843	34	34	0

TABLE 4.11 – Paramètres et résultats de quelques écoles namuroises pour le modèle 2

Nous pouvons remarquer que la distribution des étudiants dans ces 10 écoles namuroises est nettement meilleure que pour le premier modèle. En raison de la marge d'erreur due à l'implémentation de la synchronisation à travers les processeurs, même si certaines écoles ont un étudiant ou deux en plus, on peut considérer que la répartition respecte la contrainte du modèle d'assignation . Nous pouvons donc laisser une marge d'erreur de quelques étudiants en plus ou en moins dans les écoles tout en considérant la contrainte (4.12) respectée par le modèle.

Cependant, même si ces écoles sont proches de leur quota, d'autres le sont moins. En effet, la table 4.12 reprend certaines écoles qui empêchent le modèle n°2 de respecter la contrainte (4.12).

Identifiant	Numéro de fase	Quota	Nombre d'étudiants inscrits	Nombre de places restantes
4329	6333	689	272	417
4555	5369	723	472	251
4556	5374	529	306	223
4637	5525	677	692	-15
4677	5592	548	557	-9
4847	92094	1319	1427	-108
4875	5776	979	1044	-65

TABLE 4.12 – Ensemble d'écoles qui ne respectent pas la condition (4.12)

Nous essayons de connaitre la précision de ce modèle. Pour cela, nous calculons différents paramètres qui sont stockés dans la table 4.13.

Notions [1]	Valeur
Pourcentage d'écoles dont le quota est atteint	66.6%
Pourcentage d'écoles dont le quota a été dépassé	32.4%
Pourcentage d'écoles dont il reste au moins une	0.99%
Pourcentage d'écoles dont le quota a été dépassé de 5 étudiants	4.03%
Pourcentage d'écoles dont il reste au moins 5 places non attribuées	0.033%
Pourcentage d'étudiants qui s'inscrivent dans une école dont le quota est déjà atteint	0.003%
Pourcentage de places dans les écoles qui ne sont pas occupées	0.19%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la moins remplie	43%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la plus remplie	108.1%
Le nombre de places restants en moyenne dans une école	0 (0.014)
L'écart-type [2] du nombre de places restantes dans les écoles	17 (16.80)

TABLE 4.13 – Analyse des résultats du modèle 2

1. Pour lire ce tableau, il faut garder en mémoire que les pourcentages sont arrondis au centième près.
2. L'écart-type d'une série statistique ou d'un variable aléatoire X mesure la dispersion des valeurs de X autour de la moyenne.

Si on regarde les trois premiers pourcentages de la table 4.13, on peut voir qu'il y a plus 66 % des écoles qui ont le bon nombre d'étudiants comparé à 32.4% qui ont un surplus d'étudiants. Vu notre marge d'erreur, il est plus correct de prendre en compte le pourcentage d'écoles dont le surplus est supérieur à 5 étudiants. Comme ce dernier est égal à 4.03% et que le pourcentage d'écoles où il reste plus que 5 places à attribuer est de 0.03%, on peut dire que ce modèle est assez proche de celui souhaité.

Si on considère la contrainte (4.12) que le modèle d'assignation doit respecter et qu'on laisse une petite marge d'erreur due à l'implémentation égale à 5 étudiants, nous pouvons conclure qu'un peu moins de 96% des écoles ont atteint le nombre souhaité d'étudiants à 5 étudiants près.

Cependant, nous pouvons quand même constater dans la table 4.12, qu'il existe des écoles moins remplies que d'autres et dans la table 4.13, qu'il existe au moins une école qui n'a pas dépassé la moitié de son quota alors que d'autres sont en surplus de 8 % (ce qui correspond à plus 24 étudiants dans une école dont la limite est de 300 étudiants). De plus, l'écart-type de l'ensemble des places restantes est égal à 17. Même si le modèle est déjà acceptable, nous pouvons l'améliorer, et diminuer l'écart-type, avec l'ajustement des paramètres tels que le nombre de synchronisations, la valeur de l'epsilon,... et ainsi mieux répartir les étudiants.

Avant de commencer à faire des tests pour le calibrage des paramètres, nous nous demandons si le modèle ne serait pas meilleur en travaillant avec une probabilité d'assignation liée aux places restantes plutôt que liée aux quotas. Nous allons donc essayer de vérifier cette hypothèse avec le modèle n°3. Après la comparaison entre les deux modèles, on utilisera le meilleur pour effectuer tous les tests et comprendre l'influence des valeurs des paramètres dans le modèle.

8. Troisième modèle testé:

8.1. Modèle

Les différentes étapes de ce modèle sont identiques au premier modèle détaillé au point 5.1. de cette section excepté pour l'étape 5a. Afin d'éviter un surplus d'étudiants dans les écoles virtuelles et une meilleure distribution des étudiants, ces dernières doivent refuser des étudiants lorsque leur quota est atteint. Cependant, le nombre d'élèves déjà inscrits dans les écoles peut être surveillé tout au long du processus contrairement au deuxième modèle testé. En d'autres mots, une école qui a presque atteint son quota d'étudiants a moins de chance d'être à nouveau choisie qu'une ayant peu d'étudiants inscrits. En modé-

lisation, cela signifie que la probabilité de ces écoles d'être choisie va dépendre du nombre de places restantes dans ces dernières.

Soit un étudiant virtuel et soit le noeud o, son domicile :

1. Détermination du type de l'école dans laquelle l'individu doit être assigné en suivant l'explication donnée dans la section 4.3.2. Notons ce type, « type ».
2. Évaluation de la distance à laquelle doit se trouver approximativement l'école de l'étudiant, notée d, à l'aide d'un générateur de nombres pseudo-aléatoires suivant la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.8792$.
3. Création d'un ensemble de noeuds qui sont à une distance d du noeud o, à un epsilon près, égal à 250 mètres. En d'autres mots, si la distance est égale à 1000m alors les noeuds de l'ensemble créé seront à une distance entre 750 m et 1250 m du domicile (voir la figure 4.9).
4. Sélection des noeuds représentant une école du même type que la variable « type » dans cet ensemble. Si un noeud abrite deux écoles du même type, alors il sera sélectionné deux fois et sera distingué par les caractéristiques de ces dernières.

   Cette liste contient $n_{\text{noeud}_{\text{selectionné}}}$ noeuds. Les écoles abritées par ces noeuds sont classées. Il y a n_E écoles avec quota_{e_i}, le quota de la $i^{\text{ième}}$ école de la liste, notée e_i, $\forall\ 1 \leq i \leq n_E$.

   Si l'ensemble des noeuds représentant une école est vide alors on retourne à l'étape n°2 où on régénère une nouvelle distance, sinon on passe à l'étape suivante.
5. Déduction de l'école et donc du noeud où l'étudiant effectuera ses études :
   (a) Calcul du nombre de places restantes dans les écoles virtuelles des noeuds sélectionnés.

   soient :

n_E	le nombre d'écoles abritées par les noeuds sélectionnés,
e_i	la $i^{\text{ième}}$ école de la liste déterminée à l'étape précédente,
$NbEtudiant_{e_i}$	le nombre d'étudiants inscrits dans l'école e_i,
quota_{e_i}	le nombre souhaité d'étudiants dans l'école e_i
$NbPlace_{e_i}$	le nombre de places restant dans l'école e_i avant qu'elle atteigne son quota.

   $$\forall\ 1 \leq i \leq n_E \text{ où } e_i \text{ est une école}$$

   $$NbPlace_{e_i} = \text{quota}_{e_i} - NbEtudiant_{e_i}$$

(b) Calcul de la probabilité de chaque école de la liste d'être choisie par l'étudiant.

Ces probabilités sont déterminées en fonction de la formule suivante :

$$\forall\ 1 \leq i \leq n_E \text{ où } e_i \text{ est une école}$$

$$P_i = \frac{NbPlace_{e_i}}{\sum_{j=1}^{n_E} NbPlace_{e_j}}$$

(c) Utilisation de la méthode de la fonction de répartition comme expliqué dans le premier modèle

6. Attribution de l'école à l'individu.

8.2. Implémentation

Au point de vue de l'implémentation de ce modèle, il n'y a pas beaucoup de changement par rapport au modèle précédent excepté que l'on calcule les probabilités avec le nombre de places restantes dans les écoles. Les marges d'erreurs concernant la synchronisation sont toujours présentes, nous devons donc légèrement modifier le modèle du point de vue de l'implémentation. En effet, nous avons vu dans le modèle précédent que les processeurs agissent simultanément et donc plusieurs peuvent assigner la dernière place d'une école à un étudiant sans tenir compte des autres processeurs qui font la même démarche. En conclusion, le nombre de places restantes n'est pas toujours nul dans une école et peut devenir négatif. Or les probabilités négatives n'existant pas, nous devons donc imposer une condition sur le nombre de places restantes :

1. On calcule le nombre de places restantes :

$$\forall\ 1 \leq i \leq n_E \text{ où } e_i \text{ est une école}$$

$$NbPlace_{e_i} = \text{quota}_{e_i} - NbEtudiant_{e_i}$$

2. On impose une condition afin de permettre à la probabilité d'une école complète d'être nulle même si cette dernière a un surplus.

$$\text{Si } NbPlace_{e_i} < 0 \text{ alors } NbPlace_{e_i} = 0 \text{ et donc } P_i = 0$$

8.3. Résultats

Vu que nous ne connaissons pas le meilleur changement de valeur pour la variable afin d'obtenir une meilleure distribution des étudiants dans les écoles, nous avons décidé de tester ce modèle avec des changements de la valeur d'epsilon le plus tard possible. Cette décision est aussi prise dans le but que les distances Domicile-École soient les plus proches possibles des distances générées par le générateur de nombre speudo-aléatoire à l'étape 2 du modèle. En effet, on suppose que tant que l'epsilon vaut 250 m, les distances entre le domicile et l'école vont suivre la loi de probabilité donnée dans le point « 1. Distribution liée à la distance » puisqu'elles sont assez proches des distances tirées qui elles-même suivent cette loi. Cette hypothèse sera testée plus tard dans cette partie.

Cela signifie que plus la valeur de l'epsilon sera grande, plus la distance entre l'école choisie et le domicile de l'étudiant sera éloignée de la distance générée et donc il y aura moins de chance pour que l'ensemble des distances Domicile-École choisie suive la loi de probabilité imposée. Nous allons donc essayer d'éviter l'augmentation de la valeur de l'epsilon si on n'y est pas obligé.

Au début, nous voulions tester le modèle avec les paramètres de la table 4.14 et ensuite avec ceux de la table 4.15.

Cependant, aucun des deux tests n'a pas pu s'exécuter en moins de 3 jours [12]. Nous allons donc utiliser les changements effectués lors du travail sur le modèle 2. Cette opération réduit le temps d'exécution à plus ou moins 15h avec 100 processeurs. Nous pouvons dés lors conclure que ce sont les 20 dernières synchronisations qui prennent plus de 2 jours dans les deux premiers tests. En effet, lorsqu'on agrandit la valeur de l'epsilon aux 10 dernières itérations, le programme ne s'exécute pas en moins de 3 jours mais par contre, si l'epsilon est modifié à partir des 20 dernières itérations, le temps d'exécution devient raisonnable.

12. Le cluster Lemaitre2 impose un temps d'exécution inférieur à trois jours sinon le programme s'arrête de lui-même.

Paramètres			
Modèle	Modèle 3		
Type de l'école	Primaire		
Nombre d'étudiants	1727134		
Nombre de processeurs	100		
Nombre de synchronisations	1200		
Changement Epsilon	Changement	Numéro de la synchronisation	Epsilon
	1	1195	100 000
	2	1198	250 000

TABLE 4.14 – Premier test pour le modèle 3

Paramètres			
Modèle	Modèle 3		
Type de l'école	Primaire		
Nombre d'étudiants	1727134		
Nombre de processeurs	100		
Nombre de synchronisations	1200		
Changement Epsilon	Changement	Numéro de la synchronisation	Epsilon
	1	1190	10 000
	2	1195	50 000
	3	1198	250 000

TABLE 4.15 – Deuxième test pour le modèle 3

Paramètres			
Modèle	Modèle 3		
Type de l'école	Primaire		
Nombre d'étudiants	1727134		
Nombre de processeurs	100		
Nombre de synchronisations	1200		
Changement Epsilon	Changement	Numéro de la synchronisation	Epsilon
	1	1180	5 000
	2	1190	10 000
	3	1195	50 000
	4	1198	250 000
Temps d'exécution	± 15 heures		

TABLE 4.16 – Paramètres du troisième test modèle 3

Les paramètres et les résultats du troisième test du modèle 3 sont repris , respectivement dans la table 4.16 et dans la table 4.17.

Identifiant	Numéro de fase	Quota $quota_e$	Nombre d'étudiants inscrits $NbEtudiant_e$	Nombre de places restantes
4828	5831	165	165	0
4829	5833	134	134	0
4830	5836	754	754	0
4831	5837	146	146	0
4832	5838	249	249	0
4833	5839	452	452	0
4834	5840	213	213	0
4835	5841	785	785	0
4836	5842	194	194	0
4837	5843	34	34	0

TABLE 4.17 – Resultat du troisième test modèle 3

Cependant, même si des écoles sont complètes, d'autres le sont moins, voire surpeuplées. En effet, la table 4.18 reprend certaines écoles qui empêchent le modèle n°3 de respecter la contrainte (4.12).

Identifiant	Numéro de fase	Quota	Nombre d'étudiants inscrits	Nombre de places restantes
142	885	694	731	-37
583	25	979	1044	-65
677	171	778	862	-84
4329	6333	689	394	295
4555	5369	723	604	119
4556	5374	529	442	87

TABLE 4.18 – Une partie de l'ensemble d'écoles qui ne respectent pas la condition 4.12 pour le test 3 du modèle 3

Nous allons essayer de connaitre la précision de ce modèle. Pour cela, nous calculons différents paramètres qui sont stockés dans la table 4.19.

Notions	Valeur
Pourcentage d'écoles dont le quota est atteint	75%
Pourcentage d'écoles dont le quota a été dépassé	23.6%
Pourcentage d'écoles dont il reste au moins une place à attribuer	1.3%
Pourcentage d'écoles dont le quota a été dépassé de 5 étudiants	2.98%
Pourcentage d'écoles dont il reste au moins 5 places non attribuées	1.15%
Pourcentage d'étudiants qui s'inscrivent dans une école dont le quota déjà est atteint	0.23%
Pourcentage de places dans les écoles qui ne sont pas occupées	0.23%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la moins remplie	58.5%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la plus remplie	110.79%
Le nombre de places restantes en moyenne dans une école	0 (0.029)
L'écart-type du nombre de places restantes dans les écoles	12 (11.65)

TABLE 4.19 – Analyse des résultats du troisième test du modèle 3

Si on regarde les trois premiers pourcentages de la table 4.19, on peut voir qu'il y a 75 % des écoles qui ont le bon nombre d'étudiants comparé à 23% qui ont un surplus d'étudiants. On peut déjà remarquer que ces pourcentages sont meilleurs que pour le test du modèle n°2, effectué avec les mêmes paramètres. On a plus d'écoles avec un nombre d'étudiants correspondant à leur quota et moins d'écoles avec un surplus. En ce qui concerne le nombre d'écoles dont il reste des places, il n'y a pas de changement significatif.

Comme expliqué dans la partie concernant le deuxième modèle, il est plus juste de regarder le pourcentage d'écoles dont le surplus est supérieur à 5 étudiants. Ce dernier est égal à 2.98%. Il a très légèrement augmenté. Cela signifie qu'on a une diminution des écoles avec un surplus d'étudiants entre 0 et 5 mais on a pas vraiment d'amélioration concernant les écoles qui ont un surplus non négligeable.

Si on considère que le modèle d'assignation doit respecter la contrainte (4.12) et qu'une petite marge d'erreur égale à 5 étudiants est autorisée, nous pouvons conclure qu'un peu moins de 96% des écoles ont atteint le nombre souhaité d'étudiants à 5 étudiants près. Ce qui correspond au test du modèle n°2 mais avec une amélioration au niveau des écoles avec un surplus entre 0 et 5 mais également un meilleur écart-type, qui permet de dire que le nombre de places restantes dans une école se rapproche plus de la moyenne 0. On peut donc en conclure que ce modèle est meilleur que le modèle 2. De plus, l'application

des deux modèles prend le même temps.

Malheureusement, il existe toujours des écoles moins remplies que d'autres, au moins une école qui a à peine plus 50% de son quota ainsi que d'autres ayant un surplus de 10 % (ce qui correspond à plus 84 étudiants dans une école dont la limite est 778 étudiants). Même si le modèle est déjà acceptable et mieux que le modèle n°2, nous pouvons l'améliorer avec l'ajustement des paramètres tels que le nombre de synchronisations, la valeur de l'epsilon,... et ainsi mieux répartir les étudiants et également diminuer le temps d'exécution.

8.4 Différents tests sur le modèle 3

Comme nous venons de le voir, nous avons choisi le modèle 3 pour continuer les tests. Nous allons donc tout d'abord regarder l'influence du nombre de synchronisations sur la distribution des étudiants à travers les écoles, puis l'influence du nombre de processeurs et ensuite trouver la valeur adéquate. Pour finir, nous allons regarder à quel moment, il est le plus intéressant de changer la valeur de la variable epsilon. Cependant, pour identifier le meilleur changement, nous devons regarder si les distances entre le domicile et l'école choisie respectent la distribution de probabilité fixée au début.

8.4.1. Influence du nombre de synchronisations

Les différents tests effectués dans ce point sont réalisés avec l'ensemble des étudiants virtuels qui doivent être assignés à une école primaire mais qui ont, également une chaine d'activités avec l'activité "Ecole". Cet ensemble a été choisi afin de réduire un peu le temps d'exécution du programme pour les différents tests présentés ci-dessous mais également pour la section suivante. Précisons que la réduction du nombre d'étudiants de 1727134 à 1005053 n'influence pas les conclusions des deux comparaisons suivantes.

Avant de commencer à effectuer les différents tests, nous devons rappeler certains éléments. Premièrement, le nombre d'étudiants entre chaque synchronisation doit être non nul sinon la synchronisation ne pourra se faire. Deuxièmement, le reste de la division entière du nombre d'étudiants dans le processeur par le nombre de synchronisations est ajouté au nombre d'étudiants auquel on assigne une école avant la première synchronisation. Par exemple, si on a 27 étudiants dans un processeur et que le nombre de synchronisations vaut 5. La répartition des étudiants à travers les synchronisations pour ce processeur est réalisée dans la table 4.20.

Temps	Action
1	Affectation d'une école aux 7 premiers étudiants
2	Synchronisation 1
3	Affectation d'une école aux 5 étudiants suivants
4	Synchronisation 2
5	Affectation d'une école aux 5 étudiants suivants
6	Synchronisation 3
7	Affectation d'une école aux 5 étudiants suivants
8	Synchronisation 4
9	Affectation d'une école aux 5 étudiants suivants
10	Synchronisation 5

TABLE 4.20 – Exemple de répartition des étudiants à travers les synchronisations pour un processeur

Dans cette section, nous allons essayer de comprendre la signification d'un changement de synchronisation. Pour commencer cette comparaison, nous prenons comme référence le test n°4 dont les paramètres se trouvent dans la table 4.21. 600 synchronisations sont donc utilisées, ce qui correspond à assigner une école à entre 10 et 20 étudiants par processeur. Nous nous sommes demandé si on pouvait réduire ce nombre d'étudiants à 1 afin de réduire la marge d'erreur due aux synchronisations. En d'autres termes, nous aimerions réduire au minimum le nombre d'étudiants entre synchronisation pour permettre aux écoles des processeurs de connaitre, le plus souvent possible, le nombre exact d'étudiants inscrits et ainsi éviter une marge d'erreur trop grande. Pour déterminer ce nombre, nous nous basons sur le minimum des nombres d'étudiants de chaque processeur qui correspond à 6215 dans notre cas et nous décidons de prendre un nombre inférieur, 6000 synchronisations, sachant que la différence (215) ne change que la durée d'exécution et non la précision des attributions. Ce nombre permet de respecter la condition que chaque synchronisation ne peut se faire sur un nombre nul d'étudiants. En effet, si on divise entièrement pour chaque processeur son nombre d'étudiants par le nombre défini ci-dessus, on obtient toujours un résultat supérieur à 1 et donc au moins un étudiant par synchronisation.

Exemple, avec 6000 synchronisations pour les 6215 étudiants du processeur P, nous calculons la division entière de $\frac{6215}{6000} = 1$, ce qui signifie que qu'il y aura qu'une seule assignation d'une école entre les synchronisations mis à part avant la première où 1+215 étudiants vont être assignés à une école par le processeur P.

Paramètres	Test 4	Test 5
Modèle	Modèle 3	Modèle 3
Type de l'école	Primaire	Primaire
Nombre d'étudiants	1005053	1005053
Nombre de processeurs	100	100
Nombre de synchronisations	600	6000
Division entière [1] (min-max)	8-20	1-2
Maximum du reste [2] de la division	600	6000
Changement Epsilon	Numéro de la synchronisation / Epsilon (mètres): 500 / 50 000; 550 / 100 000; 595 / 250 000	Numéro de la synchronisation / Epsilon (mètres): 5000 / 50 000; 5500 / 100 000; 5950 / 250 000
Temps d'exécution [3]	2 h 51	9h00

TABLE 4.21 – Paramètres pour les tests 4 et 5

En ce qui concerne les changements de la valeur d'epsilon, les itérations référentes ne doivent pas être les mêmes pour les deux tests. En effet, les changements sont calculés en fonction du nombre de synchronisations. Donc si ce dernier augmente, alors les itérations référentes où on change la valeur d'epsilon, augmentent proportionnellement.

Test	Nombre de synchronisations	Itérations référentes		
Test4	600	500	550	595
Test5	6000	5000	5500	5950

Nous pouvons maintenant analyser les résultats et ainsi déterminer si le modèle avec une assignation d'une école pour un 1 ou 2 étudiants entre chaque synchronisation et par processeur est meilleur malgré l'augmentation assez conséquente du temps d'exécution.

1. Division entière représente le nombre d'étudiants auxquels on assigne une école entre chaque synchronisation. Ce nombre dépend du processeur.

2. Le reste de division entière représente une partie des étudiants auxquelles on assigne une école avant la première synchronisation.

3. Le temps d'exécution de ces tests est seulement écrit à titre informatif. En effet, pour connaitre le temps d'exécution précis, il faudra faire tourner le programme plusieurs fois et prendre la moyenne. En effet, le temps varie en fonction du nombre de noeud utilisé du cluster et ce dernier varie à chaque fois.

Notions	Test 4	Test 5
Pourcentage d'écoles dont le quota est atteint	81.48%	88%
Pourcentage d'écoles dont le quota a été dépassé	12.2%	8.25%
Pourcentage d'écoles dont il reste au moins une place à attribuer	6.23%	3.3%
Pourcentage d'écoles dont le quota a été dépassé de 5 étudiants	0%	3.5%
Pourcentage d'écoles dont il reste au moins 5 places à attribuer	0.5%	2%
Pourcentage d'étudiants qui s'inscrivent dans une école dont le quota déjà est atteint	0%	0.4%
Pourcentage de places dans les écoles qui ne sont pas occupées	0.07%	0.4%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la moins remplie	92.46%	44.9%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la plus remplie	101.8%	221%
Le nombre de places restantes en moyenne dans une école L'écart-type [13] du nombre de places restantes dans les écoles	0 1(1.2)	0 9 (9.19)

TABLE 4.22 – Analyse des résultats pour les tests 4 et 5 du modèle 3

Nous pouvons tout d'abord observer qu'il existe plus d'écoles qui respectent la contrainte (4.12) sans considérer la marge d'erreur. En considérant, maintenant, la marge d'erreur de 5 étudiants, nous obtenons cependant un plus petit pourcentage d'écoles dont le nombre d'étudiants inscrits est égal au quota de l'école à 5 étudiants près ($\pm$ 94.5% avec 6000 synchronisations contre $\pm$99.5% avec 600 synchronisations). On peut expliquer cette différence par le fait qu'utiliser un grand nombre de synchronisations c'est bien pour éviter les petits surplus dans les écoles mais cela entraine un reste de division plus élevé et donc un plus grand nombre d'étudiants auxquels on assigne une école sans effectuer une synchronisation.

Vu qu'il y a 6000 synchronisations, le reste de la division entière peut donc aller jusqu'à 5999 ce qui est assez important et conduit à des assignations dans les mêmes écoles sans vérification du caractère complet avec des étudiants des autres processeurs. Par exemple, nous avons deux processeurs, donc chacun possède, respectivement 10133 et 11137 étudiants. Nous pouvons voir dans la table 4.23, que le processeur P_{60} assigne une école à 4133 étudiants sans qu'il y a une synchronisation et dès lors sans tenir compte du nombre d'étudiants répartis dans les écoles par les autres processeurs et il en va de même avec le processeur P_{82} et ses 11137 étudiants. Donc s'il existe une école avec un quota de 300 étudiants, il se peut que 200 étudiants de processeur P_{60} soient assignés à cette école mais également 200 étudiants du processeur P_{82}. Ce qui nous amène à des écoles avec des

taux de surpeuplement élevés.

Processeur	P_{60}	P_{82}
Nombre d'étudiants	10133	11137
Nombre d'étudiants entre chaque synchronisation	$\frac{101331}{6000} \underset{\text{entière}}{\overset{\text{Division}}{=}} 1$	$\frac{11137}{6000} \underset{\text{entière}}{\overset{\text{Division}}{=}} 1$
Reste de la division	$\frac{101331}{6000} \underset{\text{division}}{\overset{\text{Reste}}{=}} 4133$	$\frac{11137}{6000} \underset{\text{division}}{\overset{\text{Reste}}{=}} 5137$
Nombre d'étudiants avant la première synchronisation	4133+1	5137+1

TABLE 4.23 – Exemple de deux processeurs pour montrer l'importance du reste de la division

Au final, pour un grand nombre de synchronisations, on risque d'avoir un grand reste dans la division et donc beaucoup d'étudiants auxquels on assigne une école avant d'effectuer la première synchronisation et cela se ressent dans la distribution des étudiants avec beaucoup plus d'écoles surpeuplées au-delà de 5 étudiants. Il faut donc trouver un bon nombre de synchronisations qui évite un trop grand nombre d'étudiants avant la première synchronisation et qui ne donne pas un trop grand nombre d'étudiants entre les synchronisations.

Nous avons réalisé différents tests pour tenter de trouver le juste milieu entre le reste de la division et le nombre d'étudiants dont on assigne un école entre chaque synchronisation par processeur. Le nombre de synchronisations va être déterminé en fonction du nombre de processeurs utilisés et également en fonction du nombre total d'étudiants de la population puisque ce sont les deux paramètres qui influencent la division entière. Les tests suivants sont donc effectués avec le même ensemble d'étudiants et le même nombre total d'étudiants afin de déterminer le juste milieu entre le résultat de la division entière (le nombre d'étudiants entre chaque synchronisation [14]) et le reste (le nombre d'étudiant avant la première synchronisation [15]).

. Nous avons choisi de tester le nombre de synchronisations égal à 600, 1200 et 1800, comme nous pouvons le voir dans la table 4.24 avec 100 processeurs et l'ensemble réduit des étudiants. En ce qui concerne le test avec 600 synchronisations, nous prenons le test 4, expliqué dans la table 4.21.

14. Par abus de langage, nous utilisons le terme : « nombre d'étudiants entre chaque synchronisation< » mais en réalité nous devrions dire « le nombre d'étudiants dont on assigne une école entre chaque synchronisation ».

15. Par abus de langage, nous utilisons le terme : « nombre d'étudiants avant la première synchronisation< » mais en réalité nous devrions dire « le nombre d'étudiants dont on assigne une école la première synchronisation ».

Paramètres	Test 6	Test 7
Modèle	Modèle 3	Modèle 3
Type de l'école	Primaire	Primaire
Nombre d'étudiants	1005053	1005053
Nombre de processeurs	100	100
Nombre de synchronisations	1200	1800
Division entière (min-max)	5-10	4-6
Maximum du reste de la division	1200	1800
Changement Epsilon	Numéro de la synchronisation / Epsilon 1000 / 50 000 1100 / 100 000 1190 / 250 000	Numéro de la synchronisation / Epsilon 1500 / 50 000 1650 / 100 000 1785 / 250 000
Temps d'exécution	3h 12	3h 21

TABLE 4.24 – Paramètres pour les tests 6 et 7

Notions	Test 6	Test 7
Pourcentage d'écoles dont le quota est atteint	88%	90.3%
Pourcentage d'écoles dont le quota a été dépassé	7.60%	6%
Pourcentage d'écoles dont il reste au moins une place à attribuer	4.35%	3.5%
Pourcentage d'écoles dont le quota a été dépassé de 5 étudiants	0%	0%
Pourcentage d'écoles dont il reste au moins 5 places à attribuer	0.4%	0.2%
Pourcentage d'étudiants qui s'inscrivent dans une école dont le quota déjà est atteint	0%	0%
Pourcentage de places dans les écoles qui ne sont pas occupées	0.07%	0.03%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la moins remplie	95.6%	95.3%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la plus remplie	103.5%	140%
Le nombre de places restantes en moyenne dans une école L'écart-type[16] du nombre de places restantes dans les écoles	0 1(0.83)	0 1(0.76)

TABLE 4.25 – Analyse des résultats pour les tests 6 et 7 du modèle 3

Nous observons dans les tables des résultats 4.22 et 4.25, que le meilleur des trois tests est le test n°6 car il possède plus d'écoles complètes et en tenant compte de la marge d'erreur de 5 étudiants, ce dernier possède le plus grand pourcentage d'écoles qui respectent la contrainte 4.12. Donc 4-6 étudiants par synchronisations et un maximum de 1800 étudiants en plus avant la première synchronisation parait être un bon juste milieu pour ce nombre réduit d'étudiants de la population synthétique et pour 100 processeurs.

Cependant, plus on va avoir un nombre important d'étudiants dans la population synthétique, plus il sera difficile de trouver un petit nombre de synchronisations qui permettra d'assigner une école à 4-6 étudiants entre chaque synchronisation par processeur tout en évitant que la reste de la division soit trop importante comme dans le test n°5.

Nous allons donc choisir dans les prochains tests, un nombre de synchronisations qui permet d'assigner une école à 8-20 étudiants par processeur entre les synchronisations. En effet, le test n°4, est moins bon que les deux autres, mais il a ± 99.5% des écoles qui respectent la contrainte (4.12) à 5 étudiants près comme les deux autres tests et il permet un nombre de synchronisations plus bas, ce qui évite des erreurs et une plus petite augmentation du temps d'exécution.

En conclusion, vu que le modèle d'assignation n°3 ne respectera jamais totalement la contrainte 4.12 dû à l'implémentation, il faut plutôt se focaliser sur le pourcentage d'écoles dont le quota a été dépassé de 5 étudiants et le pourcentage d'écoles dont il reste encore 5 places à attribuer. Or ces pourcentages sont plus ou moins les mêmes pour les trois tests. De plus, les écart-type des ces trois tests, ne sont pas fortement différents pour justifier l'augmentation du temps d'exécution. Le choix 8-20 étudiants par synchronisation nous permet ainsi d'éviter un trop grand nombre de synchronisations.

Pour l'ensemble de tous les étudiants de la population synthétique, équivalent à 1727134 étudiants et pour 100 processeurs, nous fixons le nombre de synchronisations en tenant compte du processeur qui possède le plus petit nombre d'étudiants. Dans notre cas, ce dernier possède 10540 étudiants. Nous divisons ce nombre par 8 et nous obtenons 1317,5. Nous choisissons donc de prendre 1200 synchronisations pour ce paramètre.

8.4.2. Influence du nombre de processeurs

Avant d'effectuer les différents tests pour regarder l'influence du nombre de processeurs, nous pouvons émettre l'hypothèse suivante :

Hypothèse 4.5. *Plus le nombre de processeurs augmente, plus le temps d'excéution diminue.*

Pour vérifier cette hypothèse, nous allons comparer quatre modèles exécutés avec les mêmes paramètres à l'exception du nombre de processeurs utilisés. Les différents paramètres concernant les deux tests sont stockés dans la table 4.26. Les valeurs de la variable epsilon ont été choisies aléatoirement vu qu'on ne connait pas encore la meilleure solution.

Paramètres	Test 8		Test 9	
Modèle	Modèle 3		Modèle 3	
Type de l'école	Primaire		Primaire	
Nombre d'étudiants	1005053		1005053	
Nombre de processeurs	25		50	
Nombre de synchronisations	600		600	
Changement Epsilon	Numéro de la synchronisation lors du changement	Epsilon	Numéro de la synchronisation lors du changement	Epsilon (mètres)
	500	50 000	500	50 000
	550	100 000	550	100 000
	595	250 000	595	250 000
Temps d'exécution	9h 28		6h 07	

Paramètres	Test 10		Test 11	
Modèle	Modèle 3		Modèle 3	
Type de l'école	Primaire		Primaire	
Nombre d'étudiants	1005053		1005053	
Nombre de processeurs	100		200	
Nombre de synchronisations	600		600	
Changement Epsilon	Numéro de la synchronisation lors du changement	Epsilon	Numéro de la synchronisation lors du changement	Epsilon
	500	50 000	500	50 000
	550	100 000	550	100 000
	595	250 000	595	250 000
Temps d'exécution	2 h 51		1h37	

TABLE 4.26 – Paramètres pour le test 8 à 11

Premièrement, nous pouvons voir une différence au point du vue de temps d'exécution. Le programme va de plus en plus vite lorsque le nombre de processeurs augmente. Cependant, le changement n'est pas linéaire comme nous pouvons le voir sur le graphe 4.12. Cela s'explique par le fait que plus il y a de processeurs, plus l'organisation entre les processeurs prend du temps. De plus, plus le nombre de processeurs augmentent, plus les synchronisations prennent du temps car il y a plus d'écoles à synchroniser.

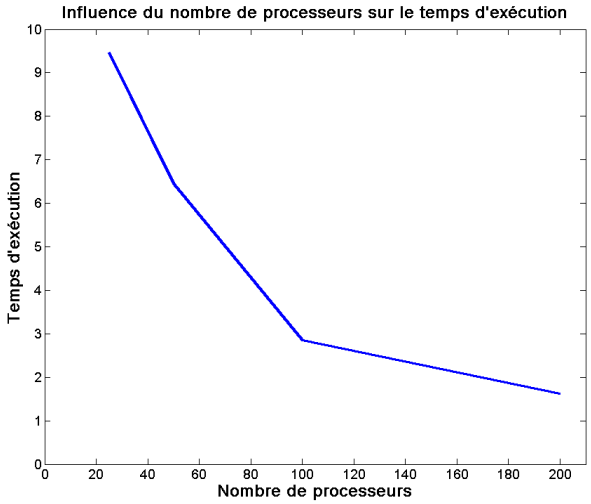


FIGURE 4.12 – Influence du nombre de processeurs par rapport au temps d'exécution

Au point de vue de la précision, nous pouvons comparer la table 4.27 qui comporte les différents paramètres qui nous aiderons à comprendre l'influence du nombre de processeurs. Nous observons une très légère amélioration de la précision en utilisant 25 processeurs à la place de 200 mais si nous regardons les résultats en considérant la marge d'erreur de 5 individus alors, nous obtenons les mêmes résultats à 1% près.

Au final, nous pouvons dire qu'il n'y a pas de changement significatif entre les résultats des différents tests. Le nombre de processeurs n'influence que très légèrement et non significativement la précision du modèle.

Notions	Test 8	Test 9	Test 10	Test 11
Pourcentage d'écoles dont le quota est atteint	82%	81.9 %	81.48%	79.69%
Pourcentage d'écoles dont le quota a été dépassé	11,18%	11.48%	12.2%	13.3%
Pourcentage d'écoles dont il reste au moins une place à attribuer	6%	6.35%	6.23%	7.5%
Pourcentage d'écoles dont le quota a été dépassé de 5 étudiants	0%	0%	0%	0%
Pourcentage d'écoles dont il reste au moins 5 places à attribuer	0.45%	0.5%	0.9%	0.6%
Pourcentage d'étudiants qui s'inscrivent dans une école dont le quota déjà est atteint	0%	0 %	0%	0%
Pourcentage de places dans les écoles qui ne sont pas occupées	0.07%	0.07%	0.07%	0.08%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la moins remplie	95.3 %	94.01 %	92.46%	92%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la plus remplie	102.2%	101.29 %	101.8%	118%
Le nombre de places restantes en moyenne dans une école	0	0	0	0
L'écart-type du nombre de places restantes dans les écoles	1	1	1	1

TABLE 4.27 – Analyse des résultats des tests 8 à 11 du modèle 3

La légère différence entre les résultats liés aux différents processeurs est due aux restes des divisions de chaque processeur entre son nombre d'étudiants et le nombre de synchronisations. En effet, le nombre d'étudiants dont on assigne une école avant la première synchronisation est augmenté considérablement en fonction du nombre de processeurs alors que le nombre d'étudiants entre les autres synchronisations ne varie presque pas. Si on regarde plus en détails, les résultats les tests 9 et 10, on peut en ressortir les chiffres suivants :

	Nombres d'étudiants avant la première synchronisation	Nombre d'étudiants entre chaque synchronisation
Test 9	12653	1654
Test 10	33056	1620

L'influence du nombre de processeurs n'est seulement visible que si on tient compte du nombre d'étudiants dont on assigne une école avant la première synchronisation. Cette différence permet de moins contrôler la répartition si on prend un nombre plus élevé de

processeur. Cependant, comme le nombre d'étudiants par synchronisation est presque identique dans les deux tests, le contrôle est rétablit par la suite ce qui évite une trop grande différence entre les résultats des tests lorsqu'on change le nombre de processeurs.

8.4.3 Influence du changement de la valeur d'epsilon

Un des éléments à déterminer reste les différents moments auxquels nous devons changer la valeur de l'epsilon pour que le programme puisse tourner dans un laps de temps raisonnable. Actuellement, pour que le programme s'exécute en moins de trois jours, nous avons déjà utilisé les changements de la valeur d'epsilon à 4 reprises. Cependant, le temps d'exécution, égal à 15h, doit encore être réduit afin de satisfaire une des conditions du modèle qui consiste à ne pas trop augmenter le temps d'exécution du programme modélisé par rapport au programme initial. De plus, les tests ne sont effectués que sur la population estudiantine des écoles primaires. Au final, le programme VirtualBelgium doit tenir compte des trois types d'enseignement. Nous devons donc réduire au maximum le temps. Il a déjà été réduit en choisissant un nombre de synchronisations pas trop élevé et donc il ne reste plus qu'à jouer sur les valeurs de l'epsilon.

Cependant, en modifiant la valeur de l'epsilon, on va augmenter la différence entre la distance générée par un générateur de loi log-normale à l'étape 2 du modèle et la vraie distance que l'étudiant parcourt pour atteindre son école. Pour rappel, la variable epsilon permet, comme montré à la figure 4.9, de trouver des écoles qui se trouvent dans l'anneau formé par la distance et l'epsilon. Nous allons donc essayer de trouver les différents changements idéaux de la valeur d'epsilon pour les dernières synchronisations de la population primaire tout en essayant de garantir le fait que les distances entre le domicile et l'école suivent la loi log-normale demandée au début de cette section.

Malheureusement, si nous utilisons le test n°3 avec le simple changement de la valeur de la variable epsilon donné par la table 4.16, nous pouvons déjà voir, grâce à la figure 4.13 que les distances parcourues par les étudiants entre domicile et l'école ne suivent pas la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$, demandée.

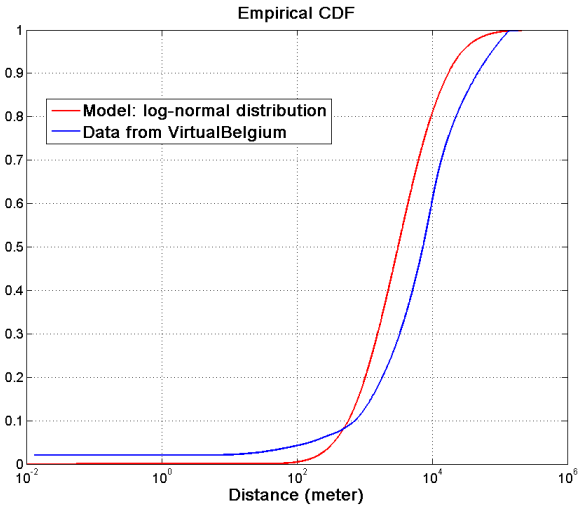


FIGURE 4.13 – Fonctions de répartition permettant d'émettre un avis négatif concernant la loi de probabilité des distances des étudiants virtuels entre leur domicile et leur école.

Pour comprendre ce phénomène et ensuite trouver une solution pour obtenir des distances suivant la loi demandée, nous allons tout d'abord vérifier si les distances générées grâce au générateur de nombres pseudo-aléatoires , implémenté par J. Barthélemy, Ph. Toint et E. Cornelis, suivent bien une loi log-normale de paramètre $\mu = 8.0259$ et $\sigma^2 = 1.879$. Ensuite, la solution trouvée sera expliquée suivie du choix réalisé pour le changement de la valeur d'epsilon au cours de dernières synchronisations.

8.4.3.1. Le générateur de nombres pseudo-aléatoires suivant une loi log-normale

Dans VirtualBelgium, il existe deux générateurs de nombres pseudo-aléatoires suivant une loi log-normale, plus précisément, un qui génère des nombres pseudo-aléatoires suivant une loi log-normale et un autre qui produit des nombres pseudo-aléatoires suivant une mixture de lois log-normales. Ces deux générateurs ne génèrent pas forcement le même résultat car ils sont construits de deux manières différentes.

Pour nos tests, nous utilisons le générateur de la mixture de lois log-normales afin d'être le plus général possible et anticiper un changement futur lorsque la distance réelle entre le domicile et l'école sera modélisée par un mixture de loi log-normale à plus d'une composante.

Pour tester ce générateur, un simple programme est construit qui permet de faire appel à ce générateur pour qu'il génère 10000 nombres pseudo-aléatoires suivant la mixture de

lois log-normales de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$. Comme nous avons pu le voir dans la table 4.4, les étudiants belges ne parcourent pas plus de 140 km pour se rendre dans leur école. Or si nous utilisons le générateur de nombres pseudo-aléatoires sans limiter les bornes, il peut ressortir des nombres allant jusqu'à l'infini. Nous devons donc imposer une condition concernant l'intervalle supérieur de l'ensemble des nombres pseudo-aléatoires. En ce qui concerne la borne inférieure, les nombres pseudo-aléatoires suivant une loi log-normale sont toujours positifs. Nous effectuons donc ce test avec le générateur produisant des nombres entre 0 et 140 000. Le graphique de la figure 4.14 permet de comparer la fonction de répartition empirique créée à partir de ces nombres [17] et celle créée à partir des nombres pseudo-aléatoires suivant la loi demandée.

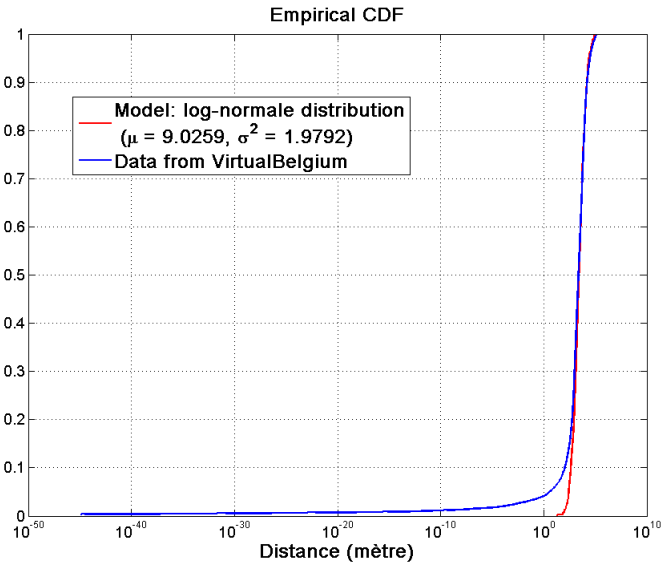


FIGURE 4.14 – Comparaison des deux fonctions de répartition empiriques avec une condition sur la borne supérieure

Nous pouvons déjà remarquer qu'il y a une grande différence dans la première partie du graphique. Ceci est dû au fait que le générateur aléatoire permet de sortir des nombres très petits, allant parfois jusqu'à 10^{-42}. Nous allons donc améliorer le générateur en fixant une borne inférieure afin de limiter les distances générées en dessous de 1m. Or nous pouvons remarquer dans la table 4.4, que les étudiants belges, questionnés lors de l'enquête Mobel, parcourent au minimum 10 m avant d'atteindre leur école. La borne inférieure sera donc fixée à 10.

17. La construction d'une fonction de répartition empirique est expliquée dans le deuxième chapitre à la section2.2.2

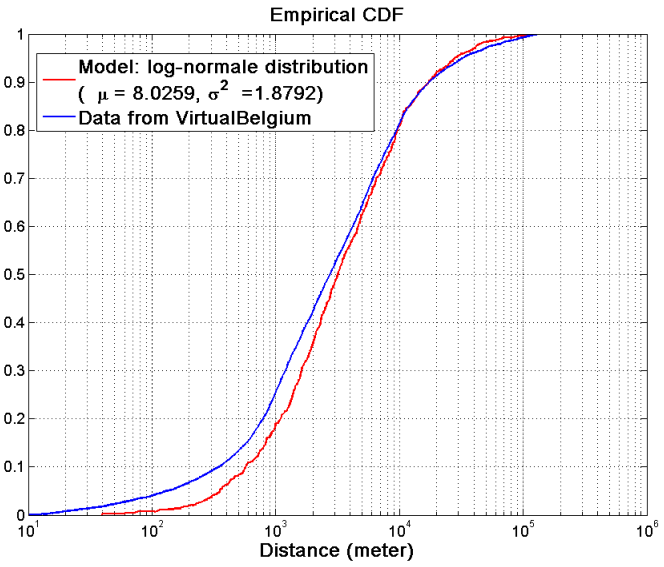


FIGURE 4.15 – Comparaison des deux fonctions de répartition empiriques avec une condition sur la borne supérieure et une condition sur la borne inférieure égale à 10

Nous pouvons voir que les deux courbes du graphe 4.15 ne correspondent pas lorsque la distance est inférieure à 10^4. En effet, lorsqu'on effectue un test statistique pour évaluer si les deux fonctions de répartition empiriques suivent la même loi, ce dernier n'est pas accepté. Le test utilisé est celui de Kolmogorov-Smirnov. Il calcule l'écart entre les deux fonctions de répartition empiriques. Ce dernier est en réalité un test d'hypothèses [18] dont l'hypothèse nulle est le fait que les deux ensembles de données suivent la même loi de probabilité. On détermine le résultat du test grâce à une p-value et un niveau de signification fixé à 0.01. Si cette p-value est supérieure à 0.01 alors l'hypothèse nulle est acceptée sinon elle est refusée.

En réalité, il faut augmenter la borne inférieure pour que le test soit accepté et que l'on puisse dire que le générateur de nombres pseudo-aléatoires génère des nombres suivant une loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$. En effet, si on choisit de fixer la distance minimum à 20 m, le test est accepté avec une p-valeur égale à 0.0574, ce qui est supérieure à 0.01. Nous pouvons néanmoins remarquer que les deux fonctions de répartitions de la figure 4.16 ne sont pas tout à fait semblables. Il faut monter la borne inférieure à 100 m pour obtenir une meilleur concordance dans les courbes. Nous allons garder 20 m comme borne inférieure.

18. D'après [19], « Un test d'une hypothèse statistique H_0 est une règle de décision, fondée sur la valeur réalisée t d'une statistique T, qui nous permet de décider si on rejette H_0 ». Cette décision peut être prise grâce au calcul de la « p-valeur ». Cette dernière « est la probabilité d'observer l'hypothèse nulle H_0, une valeur de la statistique au moins aussi extrême (au sens de H_1) que celle qui a été observée ». On rejette H_0 au niveau α si la p-valeur est inférieure ou égale à α. La valeur pour α vaut en général 0.05 ou 0.01.

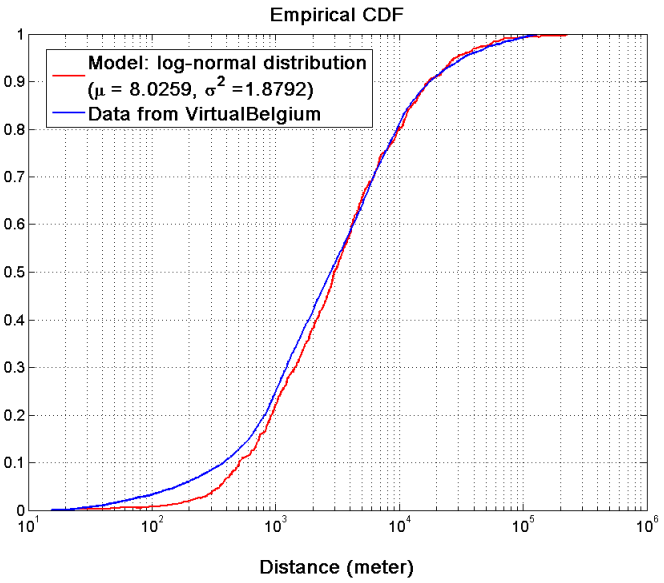


FIGURE 4.16 – Comparaison des deux fonctions de répartition empiriques avec une condition sur la borne supérieure et une condition sur la borne inférieure égale à 20

8.4.3.2. Application du générateur dans VirtualBelgium

Maintenant que nous avons amélioré le générateur afin qu'il puisse ressortir des nombres pseudo-aléatoires suivant une loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$, nous allons l'appliquer à notre modèle. En d'autres mots, nous allons l'utiliser dans l'étape 2 du modèle n°3 d'assignation et nous allons vérifier si les nombres qu'il génère suivent bien la loi demandée. Cependant, il faut garder en tête que le modèle n'utilise que les distances où il y a des écoles primaires. S'il n'y a pas d'écoles primaires à la distance générée du domicile, alors, on redemande une nouvelle distance au générateur. Nous ne garderons en mémoire que les distances utilisées pour attribuer une école.

Dans la suite de ce travail, nous appelons « distance générée », celle qui sont générées à partir du générateur dans l'étape 2 du modèle d'assignation et « distance Domicile-École », celle qui l'étudiant parcoure pour atteindre son école[19]. Elle est déterminée à l'étape 6.

Nous effectuons donc un premier test avec les paramètres de la table 4.28 où il n'y a aucun changement d'epsilon.

19. La distance entre le domicile et l'école choisie n'est pas forcement la même que la distance générée par le générateur suite à la présence de la variable epsilon.

Paramètres	Test 7
Modèle	Modèle 3
Type de l'école	Primaire
Nombre d'étudiants	5883
Nombre de processeurs	1
Epsilon	250 m

TABLE 4.28 – Paramètres pour tester le générateur

Nous allons analyser les distances qui ont été générées par le générateur lors de l'étape 2 du modèle en vérifiant si la fonction de densité créée avec ces distances concorde avec la fonction de densité de la loi log-normale. Pour cela, la figure 4.17 est mise à votre disposition.

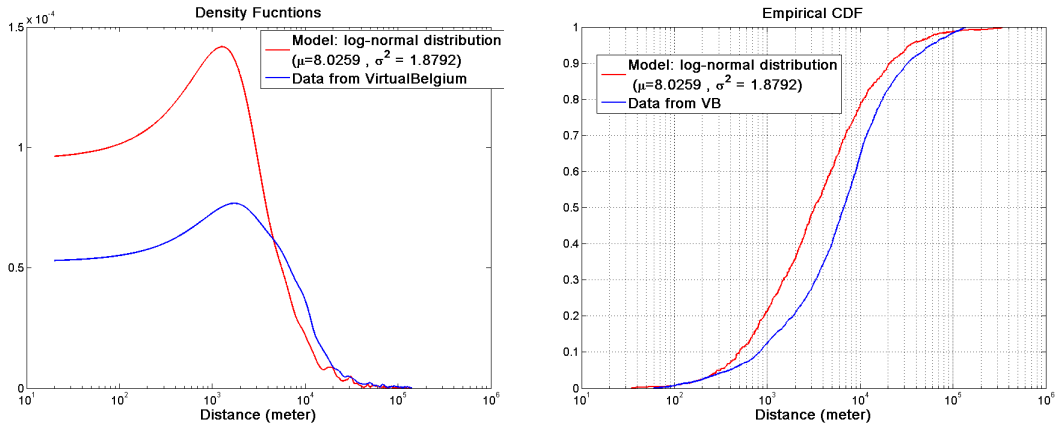


FIGURE 4.17 – Comparaison entre la fonction de densité (fonction de répartition) construite à partir des distances générées par le générateur de VirtualBelgium et la fonction de densité (fonction de répartition) de la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$

Nous pouvons remarquer que les courbes ne sont pas identiques. Or nous avons vérifié que le générateur était acceptable si on utilise les bornes inférieures et supérieures. Nous pouvons voir dans le graphe des fonctions de densité que VirtualBelgium possède moins de distances en dessous de 4472 m que la fonction de densité de la loi log-normale mais plus de distances entre 4472 m et 19650 m. En ce qui concerne les distances au dessus de 19650 m, il y en a plus ou moins le même nombre. Nous pouvons conclure que l'ensemble des distances calculées à l'étape 2 du modèle suit bien une loi log-normale mais avec des paramètres différents.

Après plusieurs recherches, nous supposons que nous obtenons ce résultat en raison du choix parmi les nombres pseudo-aléatoires générés de ne retenir que ceux où l'ensemble

des noeuds sélectionné par l'étape 4 du modèle est non vide. Nous imposons donc une condition sur ces nombres qui conduit à réduire leur caractère aléatoire et à modifier le loi de probabilité. Dans VirtualBelgium, nous observons qu'il y a peu de petites distances. Apparemment c'est assez difficile de trouver des écoles proches du domicile à 250m près. Nous allons donc essayer de comprendre pourquoi.

Comme les écoles virtuelles sont disposées aléatoirement dans le réseau, il y a deux cas possibles. D'une part, les écoles sont placées de manière plus ou moins homogène avec comme conséquence que plus on s'éloigne du domicile, plus il y a des possibilités de trouver un école comme montré avec la figure 4.18. Si elles sont placées de manière homogène, elles ont moins de probabilité de se trouver au même endroit et donc peu de chance que dans l'environnement immédiat du domicile de l'individu, il y ait plusieurs écoles. C'est pourquoi, il y a peu de courtes distances dans la distribution créée avec les distances de VirtualBelgium dans le graphe 4.17. D'autre part, on n'est pas certain que les écoles soient placées de manière homogène dans les communes. Il est tout-à-fait possible de trouver toutes les écoles d'une commune dans un même zone. Cela vaut dire qu'un étudiant qui a son domicile à l'opposé de cette zone, ne trouvera pas d'école dans son voisinage et devra effectuer un plus grand nombre de kilomètres.

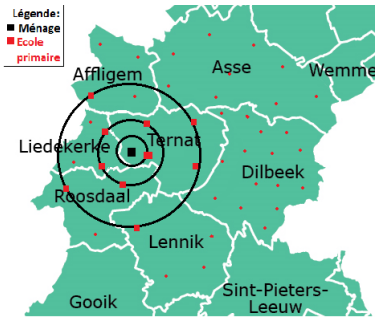


FIGURE 4.18 – Exemple de différentes distances générées

En conclusion, il est assez difficile de trouver une école proche du domicile. Une autre façon de prouver cette hypothèse est de réaliser un test en doublant le nombre d'écoles et de constater qu'avec plus d'écoles dans le réseau belge, la distribution des écoles se porte mieux (voir la figure 4.19).

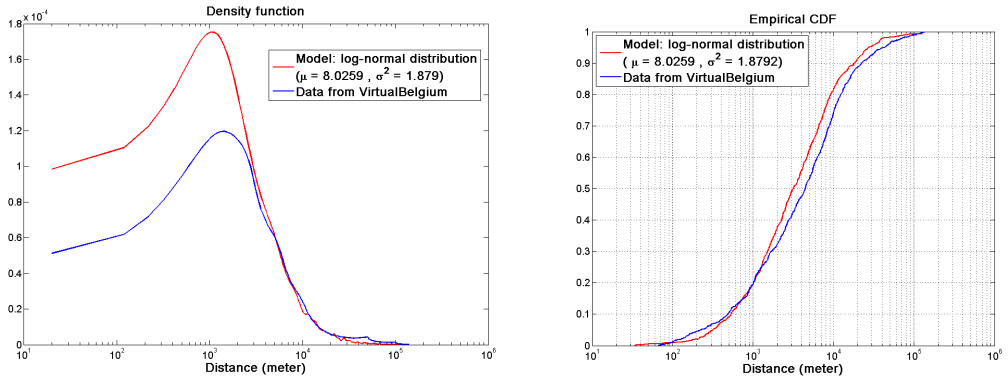


FIGURE 4.19 – Comparaison entre la fonction de densité (fonction de répartition) construite à partir des données provenant de VirtualBelgium où le nombre d'école a doublé et la fonction de densité (fonction de répartition) de la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$

Cependant, nous devons respecter le nombre d'écoles défini dans la section 3.3. Nous allons donc devoir modifier certaines parties du modèle.

Reprenons le modèle n°3 :
Soit un étudiant virtuel et soit le noeud o, son domicile :

1. Détermination du type de l'école dans laquelle l'individu doit être assigné en suivant l'explication donnée dans la section 4.3.2. Notons ce type, « type ».
2. Évaluation de la distance à laquelle doit se trouver approximativement l'école de l'étudiant, notée d.
3. Création d'un ensemble de noeuds qui sont à une distance d du noeud o, à un epsilon près, égal à 250 mètres.
4. Sélection des noeuds représentant une école du même type que la variable « type » dans cet ensemble. Si un noeud abrite deux écoles du même type, alors il sera sélectionné deux fois et sera distingué par les caractéristiques de ces dernières.

   Cette liste contient $n_{\text{noeud}_{\text{selectionné}}}$ noeuds. Les écoles abritées par ces noeuds sont classées. Il y a n_E écoles avec D_{e_i}, le quota de la $i^{\text{ième}}$ école de la liste, notée e_i, $\forall\ 1 \leq i \leq n_E$.

   Si l'ensemble des noeuds représentant une école est vide alors on retourne à l'étape n°2 où on reprend une nouvelle distance sinon on passe à l'étape suivante.
5. Déduction de l'école et donc du noeud où l'étudiant effectuera ses études en trois étapes :

(a) Calcul du nombre de places restantes dans les écoles virtuelles des noeuds sélectionnés.

soient : n_E le nombre d'écoles abritées par les noeuds sélectionnés,
e_i la $i^{\text{ième}}$ école de la liste déterminée à l'étape précédente,
$NbEtudiant_{e_i}$ le nombre d'étudiants inscrits dans l'école e_i,
quota_{e_i} le nombre souhaité d'étudiants dans l'école e_i
$NbPlace_{e_i}$ le nombre de places restantes dans l'école e_i avant qu'elle atteigne son quota.

$$\forall\ 1 \leq i \leq n_E \text{ où } e_i \text{ est une école}$$

$$NbPlace_{e_i} = \text{quota}_{e_i} - NbEtudiant_{e_i}$$

(b) Calcul de la probabilité de chaque école de la liste d'être choisie par l'étudiant à partir du noeud o.

Ces probabilités sont déterminées en fonction de la formule suivante :

$$\forall\ 1 \leq i \leq n_E \text{ où } e_i \text{ est une école}$$

$$P_i = \frac{NbPlace_{e_i}}{\sum_{j=1}^{n_E} NbPlace_{e_j}}$$

(c) Utilisation de la méthode de la fonction de répartition comme expliqué dans le premier modèle

6. Attribution de l'école à l'individu.

Nous allons modifier les étapes 2 et 4. En effet, lorsque l'on génère une distance avec le générateur utilisé, on vérifie à l'étape 4 s'il y a des écoles à cette distance du domicile. S'il y en a alors on continue, sinon on régénère une nouvelle distance à l'étape 2. Or nous avons vu plus tôt que le générateur fonctionne lorsqu'on utilise toutes les distances générées et pas qu'une partie sélectionnée. Nous allons donc faire le maximum pour éviter de retourner à l'étape 2 et ainsi garder un maximum de valeurs générées les unes à la suite des autres.

Une solution testée pour résoudre ce problème est le changement de la valeur de l'epsilon lorsqu'on s'aperçoit que l'ensemble des noeuds représentant une école est vide. Ainsi on garde la distance générée mais on augmente juste la valeur de l'epsilon pour trouver une école plus près ou plus loin mais tout en restant dans le même ordre de grandeur car

nous devons également vérifier que la distance entre le domicile et l'école choisie suive la loi de probabilité donnée.

En conclusion, nous devons donc essayer de modifier la valeur de l'epsilon afin d'éviter de générer une nouvelle distance mais cette valeur ne doit pas être trop grande afin que l'ensemble des distances entre le domicile et les écoles choisies suive la loi de probabilité log-normale demandée.

Différents modes de calcul ont été testés. Par exemple, soit la valeur de l'epsilon est un certain pourcentage de la distance, soit la valeur de l'epsilon dépend de la distance sous la forme suivante : si la distance est inférieure à 5000 m, la valeur de l'epsilon est un certain pourcentage de la distance sinon elle vaut 2500m. Malheureusement, tous ces tests où on ne modifie qu'une seule fois la valeur de l'epsilon ne donnent pas de résultats concluants pour le générateur de distances. Nous décidons donc de modifier graduellement la valeur de l'epsilon pour trouver une école dont la distance entre le domicile de l'étudiant et elle-même est la plus proche possible de la distance générée par le générateur. Ainsi lorsque la distance générée est assez petite, nous essayons de trouver l'école la plus proche.

Après plusieurs tests, on peut mettre en évidence deux ensembles de modifications possibles qui s'effectuent en plusieurs étapes et dépendent de la valeur de la distance générée à l'étape 2.

- Le premier ensemble de modifications tient compte de la valeur de la distance générée. Dans la table 4.29, vous pouvez trouver les modifications de la variable epsilon apportées aux distances inférieures à 1000m. Les autres modifications peuvent être visibles dans l'annexe F.1.

Soient :	dist	la distance générée par le générateur à l'étape 2 du modèle n°3.
	dist2	une deuxième distance générée par le générateur à l'étape 2 du modèle n°3.
	Epsilon	la valeur de l'epsilon,
	$n_{\text{noeud}_{\text{selectionné}}}$	le nombre de noeuds représentant une école, sélectionnés à l'étape 4 du modèle

Si dist $<$ 1000

Étape	Distance générée	Valeur Epsilon	Prochaine action
1	dist	250	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°2 sinon STOP
2	dist	dist * 0.25	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°3 sinon STOP
3	dist	dist * 0.50	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°4 sinon STOP
4	dist	dist * 0.75	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°5 sinon STOP
5	dist	dist	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°6 sinon STOP
6	dist	dist * 1.25	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°7 sinon STOP
7	dist	dist * 1.50	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°8 sinon STOP
8	dist	dist * 1.75	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°9 sinon STOP
9	dist	dist * 2	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°10 sinon STOP
10	dist	dist * 2.25	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°11 sinon STOP
11	dist	dist * 2.50	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°12 sinon STOP
12	dist	2500	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°13 sinon STOP
13	dist	5000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°1 avec distance= dist2 sinon STOP

TABLE 4.29 – Modification de la valeur d'epsilon si la distance générée est inférieure à 1000 m

Avec cet ensemble de modifications, nous allons vérifier les distances générées par le générateur mais également les distances Domicile-École choisies. Pour cette vérification, le test a été réalisé sur la population de Namur afin d'agrandir la population estudiantine allant dans les écoles primaires tout en restant avec un temps d'exécution convenable. Les autres paramètres de ce test sont repris dans la table 4.30.

Paramètres	Test 7
Modèle	Modèle 3
Type de l'école	Primaire
Nombre d'étudiants	11000
Nombre de processeurs	1
Epsilon	250 m

TABLE 4.30 – Paramètres pour le modèle avec l'ensemble de modifications de la valeur de l'epsilon

Nous comparons les fonctions de répartition créées à partir des données provenant des distances générées et à partir d'une loi log-normale de paramètre $\mu = 8.0259$ et $\sigma^2 = 1.879$ (4.20). Dans le premier graphe, la borne inférieure vaut 20 et dans l'autre, la borne inférieure vaut 100.

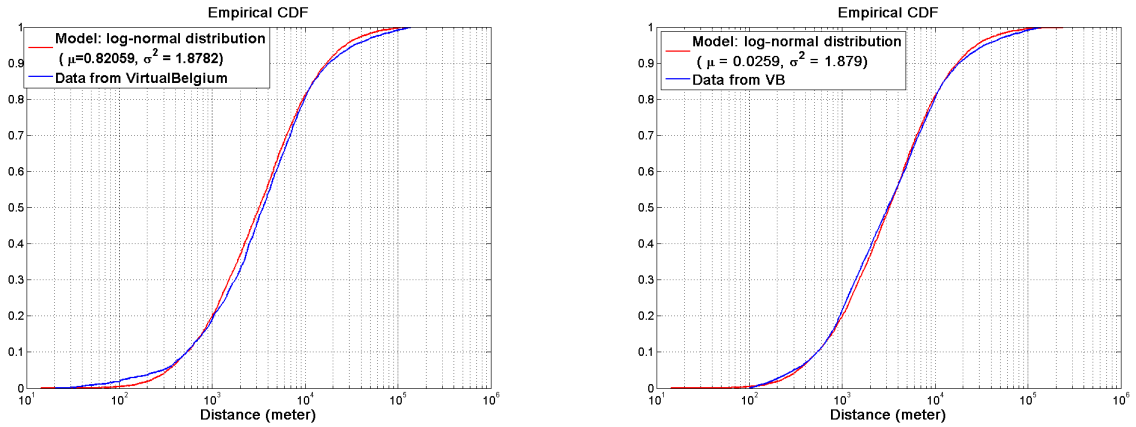


FIGURE 4.20 – Comparaison entre la fonction de répartition construite à partir des données provenant de VirtualBelgium et la fonction de répartition de la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$ pour la popualtion de Namur avec à gauche une borne inférieure égale à 20 et à droite égale à 100. Ces graphes sont construits avec le premier ensemble de modifications.

Tout d'abord, nous pouvons voir que la fonction de répartition des données de VirtualBelgium correspond plus à la fonction de répartition théorique en comparaison

avec les tests précédents. De plus, nous pouvons apercevoir une amélioration lorsque la borne inférieure est égale à 100. En effet, la p-valeur du test de Kolmogorov-Smirnov pour la comparaison entre les fonctions de répartitions pour la borne de 20 m avec un niveau égal à 0.01, est seulement de 10^{-6} comparé à 0.0202 si la borne inférieure est à 100 m. Nous pouvons conclure que le générateur de nombres pseudo-aléatoires est meilleur lorsque la borne inférieure est égale à 100 m et il est accepté par le test de Kolmogorov-Smirnov. Malgré le fait de prendre une borne inférieur à 100 m, en raison de la présence de la variable epsilon, une distance moindre à 100m pourrait être acceptée.

Nous vérifions maintenant si les distances Domicile-École suivent bien la loi de probabilité log-normale avec les paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$ où le générateur possède une borne inférieure égale à 100. Nous pouvons utiliser 100 m tout en ayant accès aux écoles dont la distance du domicile est inférieure à 100 grâce à la présence de la valeur epsilon dans le modèle.

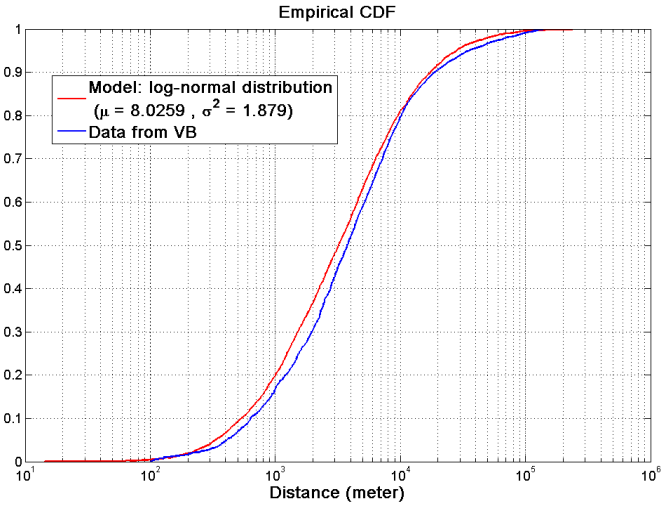


FIGURE 4.21 – Comparaison entre la fonction de répartition construite à partir des distances Domicile-École provenant de VirtualBelgium et la fonction de répartition de la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$ pour la popualtion de Namur

Bien que les courbes ont l'air d'être proches, le test de Kolmogorov-Smirnov n'est pas accepté car la p-valeur vaut 10^{-12}. Les distances Domicile-École ne suivent donc pas la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$ mais on peut constater qu'il y a une amélioration par rapport aux tests précédents

- Le deuxième ensemble de modifications ne tient pas compte aussi précisément de la valeur de la distance générée. Ces modifications sont visibles dans la table 4.31.

Soient : dist la distance générée par le générateur à l'étape 2 du modèle n°3.
dist2 une deuxième distance générée par le générateur à l'étape 2 du modèle n°3.
Epsilon la valeur de l'epsilon,
$n_{\text{noeud}_{\text{selectionné}}}$ le nombre de noeuds représentant une école, sélectionnés à l'étape 4 du modèle

Étape	Distance générée	Valeur Epsilon	Prochaine action
1	dist	250	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°2 sinon STOP
2	dist	dist $< 5000 \Rightarrow$ dist * 0.25 dist $\geq 5000 \Rightarrow$ 1000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°3 sinon STOP
3	dist	dist $< 5000 \Rightarrow$ dist * 0.55 dist $\geq 5000 \Rightarrow$ 1500	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°4 sinon STOP
4	dist	dist $< 5000 \Rightarrow$ dist * 0.75 dist $\geq 5000 \Rightarrow$ 2000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°5 sinon STOP
5	dist	dist $< 5000 \Rightarrow$ dist dist $\geq 5000 \Rightarrow$ 2500	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°6 sinon STOP
6	dist	2500	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°7 sinon STOP
7	dist	5000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°1 avec distance= dist2 sinon STOP

TABLE 4.31 – Modifications de la valeur d'epsilon pour le deuxième ensemble

Avec ce nouvel ensemble de modifications qui privilégie la rapidité plutôt que la précision, nous réalisons le même test. La fonction de répartition empirique créée avec les distances générées par le générateur est comparée à la fonction de répartition de la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$ dans la figure 4.22.

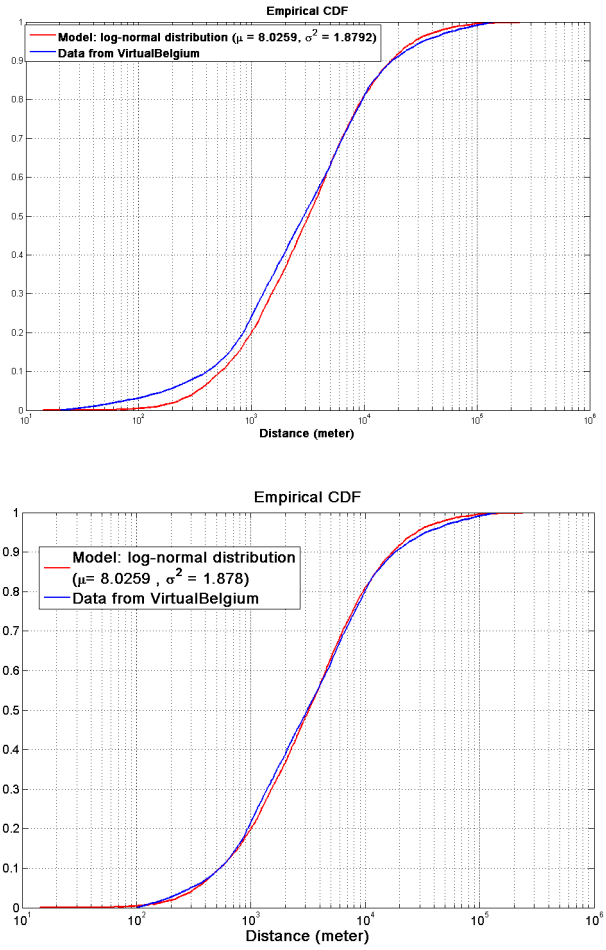


FIGURE 4.22 – Comparaison entre la fonction de répartition construite à partir des données provenant de VirtualBelgium et la fonction de répartition de la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$ pour la popualtion de Namur avec à gauche une borne inférieure égale à 20 et à droite égale à 100. Ces graphes sont construits avec le deuxième ensemble de modifications.

Nous pouvons remarquer que cet ensemble de modifications permet également à la fonction de répartition créée avec les distances générées (avec 20 ou 100 comme borne inférieure) de se rapprocher de celle de la loi log-normale demandée. Le test de Kolmogorov-Smirnov est donc effectué afin de pouvoir déterminer si nous pouvons dire que l'ensemble des distances suit statistiquement la loi de probabilité avec un niveau égale à 0.01. Pour la borne inférieure égale à 20m, nous obtenons une p-valeur égale à 1.14710^{-007}. Pour la borne inférieur à 100m, il y a une amélioration et la p-valeur est égale à 0.0199 . Nous pouvons donc accepter ce deuxième test.

Nous vérifions maintenant si les distances Domicile-École suivent bien la loi de probabilité log-normale avec les paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$. Cette vérification est visible grâce à la figure 4.23.

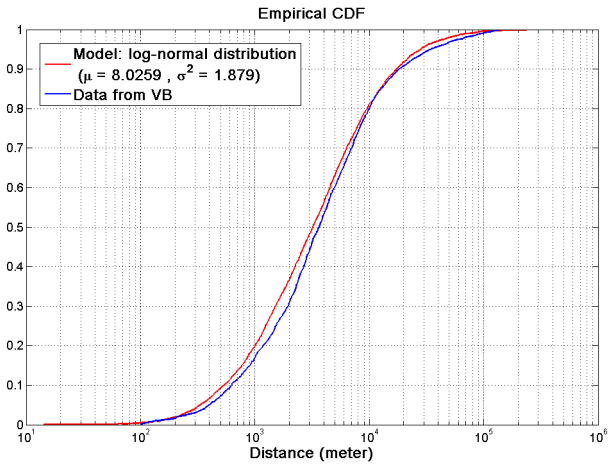


FIGURE 4.23 – Comparaison entre la fonction de répartition construite à partir des distances Domicile-École provenant de VirtualBelgium et la fonction de répartition de la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$ pour la popualtion de Namur. Ce graphe a été construit avec le deuxième ensemble de modificaitons.

Bien que les courbes aient l'air d'être proches, le test de Kolmogorov-Smirnov n'est pas accepté car la p-valeur vaut 10^{-12}. Les distances Domicile-École ne suivent donc pas la loi log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.879$.

Avec ces deux ensembles de modifications, nous avons pu obtenir des distances générées pour les étudiants de Namur suivent statistiquement la loi log-normale demandée. Malheureusement, nous avons pu constater que les distances domiciles-écoles ne suivent pas cette loi. Nous n'avons pas pu trouver un ensemble de modifications permettant une meilleure comparaison. Toutefois, nous avons une nette amélioration au niveau du générateur comparé aux graphes 4.17 et par ailleurs, ces ensembles de modifications, testés sur 1000 étudiants vérifient les tests Kolmogorov-Smitnov.

Afin de retenir le meilleur ensemble de modifications, nous allons faire une comparaison aux points de vue des distances, de la distribution des étudiants et du temps d'exécution en utilisant les paramètres du test 3, donnés par la table 4.16. Cependant, vu que le temps d'exécution est égal à 15h, nous essayons tout d'abord de le réduire dans le point suivant.

8.4.3.3. Changement de la valeur de l'epsilon aux cours des dernières synchronisations pour réduire le temps d'exécution

Comme expliqué ci-dessous, nous devons changer la valeur de l'epsilon au cours du modèle afin d'approcher le plus possible les distances Domicile-École de la loi log-normale de paramètre $\mu = 8.0259$ et $\sigma^2 = 1.879$. Cependant, nous avons expliqué dans le point 7.2. de cette section qu'il n'était pas possible de garder la même méthode dans les dernières synchronisations. En effet, pour ces dernières, les écoles sont de plus en plus complètes. Il est donc de plus en plus difficile d'en trouver une. Nous ne pouvons pas nous permettre de générer un nombre impressionnant de distances avec le générateur pour trouver les écoles où il reste encore des places libres. Cela entrainerait un boucle presque infinie.

Nous avons déjà effectué des changements pour les dernières itérations pour le test n°3 (voir table 4.16). Cependant, ces changements ne suffisent pas avec un temps d'exécution égal à plus ou moins 15h. Donc même si cela risque de modifier les distances parcourues pour les derniers étudiants, nous sommes obligé d'imposer une grande valeur d'epsilon pour les dernières synchronisations afin de respecter la contrainte liée au temps. Toutefois, nous pouvons ajuster les moments où on effectue ces changements.

Pour connaitre le moment où le temps d'exécution augmente considérablement à cause de l'application du modèle, nous avons construit deux graphes avec le test n°3. La première figure 4.24 représente le nombre de distances générées entre chaque synchronisation pour un processeur en particulier (P_0). Deux remarques sont mises en évidence. D'une part, le nombre de distances générées avant la première synchronisation est assez élevé car il y a beaucoup d'étudiants auxquels on assigne une école en raison du reste de la division. D'autre part, à partir de la 1075 $^{\text{ième}}$ synchronisation, nous pouvons observer une augmentation du nombre de distances générées. Au début, cette variation est faible mais devient assez importante vers la fin des synchronisations, c'est-à-dire au moment où les écoles sont presque complètes. L'assignation des écoles requiert donc beaucoup de temps entre ces dernières synchronisations.

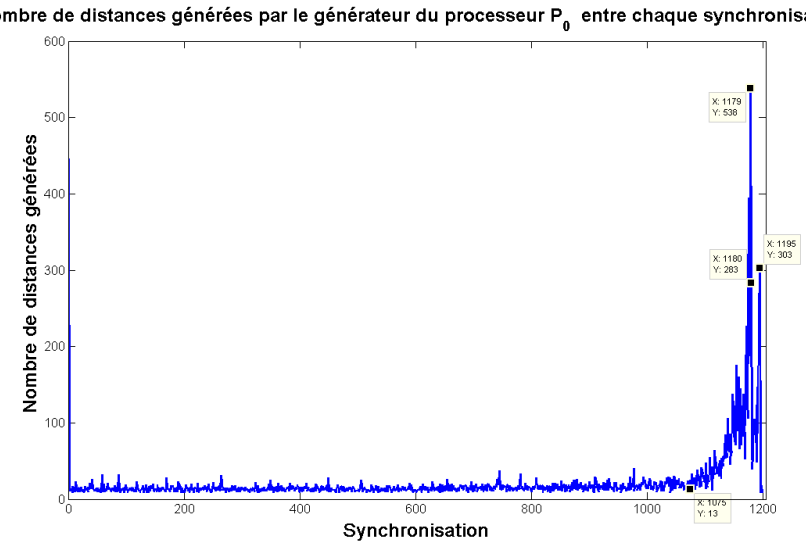


FIGURE 4.24 – Nombre de distances générées entre chaque synchronisation pour le test n°3.

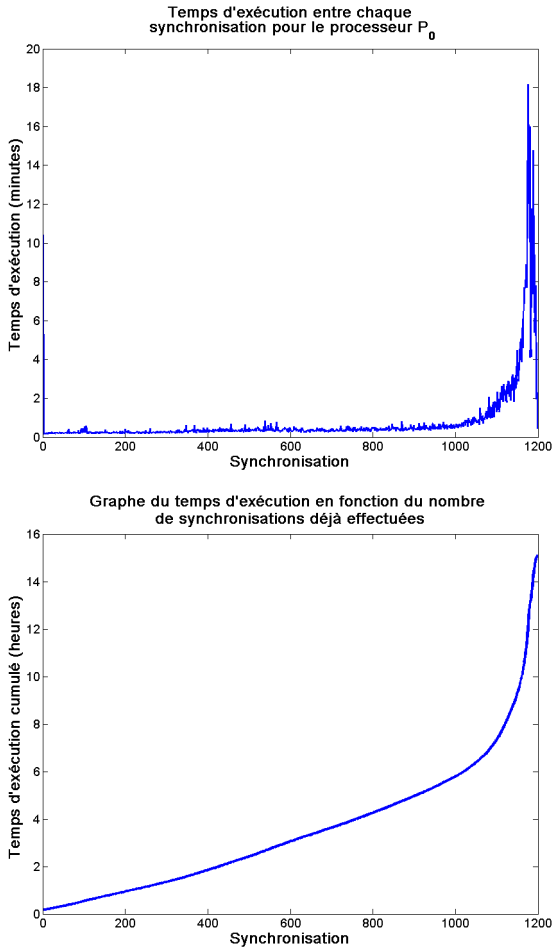


FIGURE 4.25 – Temps d'exécution lors de l'application du modèle pour le test n°3.

Nous pouvons également remarqué que le nombre de distances générées est assez important jusqu'à la 1180$^{\text{ième}}$ synchronisation et puis diminue car le test n°3, impose un changement de la valeur d'epsilon pour les 20 dernières synchronisations. Ce qui permet de trouver facilement une école et donc diminuer le nombre de distances générées. Nous devons donc imposer un changement de la valeur d'epsilon avant la synchronisation n°1075.

Pour compléter cette analyse, nous utilisons les graphes 4.25, qui nous permettent de déclarer que l'assignation d'une école aux étudiants du processeur P_0 entre chaque synchronisation prend une durée approximativement identique, excepté à partir de la $\pm$1000 $^{\text{ième}}$ synchronisation.

Afin de réduire le temps d'exécution, nous décidons donc de modifier la valeur de la variable epsilon pour toutes les assignations qui sont effectuées après la 1000 $^{\text{ième}}$ synchronisation. En ce qui concerne la nouvelle valeur de l'epsilon, nous avons réalisé deux tests avec des moments différents pour le changements des valeurs d'epsilon, comme exposé dans la table 4.32.

<table>
<tr><td>Paramètres</td><td>Test 12</td><td>Test 13</td></tr>
<tr><td>Modèle</td><td>Modèle 3</td><td>Modèle 3</td></tr>
<tr><td>Type de l'école</td><td>Primaire</td><td>Primaire</td></tr>
<tr><td>Nombre d'étudiants</td><td>1727134</td><td>1727134</td></tr>
<tr><td>Nombre de processeurs</td><td>100</td><td>100</td></tr>
<tr><td>Nombre de synchronisations</td><td>1200</td><td>1200</td></tr>
<tr><td>Ensemble des modifications d'epsilon pour l'étape 3 du modèle</td><td>Premier</td><td>Premier</td></tr>
<tr><td>Changement Epsilon pour les dernières synchronisations</td><td><table><tr><td>Numéro de la synchronisation</td><td>Epsilon</td></tr><tr><td>1000
1100
1150
1175
1190
1195</td><td>20 000
40 000
60 0000
80 000
100 000
250 000</td></tr></table></td><td><table><tr><td>Numéro de la synchronisation</td><td>Epsilon</td></tr><tr><td>1000
1100
1195</td><td>50 000
100 000
250 000</td></tr></table></td></tr>
<tr><td>Temps d'exécution</td><td></td><td></td></tr>
</table>

TABLE 4.32 – Tests pour trouver les meilleurs valeurs de la variable epsilon aux cours des 200 dernières itérations

Nous avons comparé les résultats mais le premier test ne convient pas. En effet, la valeur d'epsilon après la synchronisation n°1000 pour le premier test est trop faible pour permettre un réduction visible du temps d'exécution. Nous choisissons donc les itérations référentes correspondant au test n°13 pour un nombre de synchronisations égale à 1200 et un nombre de processeurs égal à 100.

En conclusion, pour un nombre de synchronisations, noté n, les changements de la valeur d'epsilon sont équivalents à :

Numéro de la synchronisation	Epsilon
$n * \frac{5}{6}$	50 000
$n * \frac{11}{12}$	100 000
$n - 5$	250 000

8.4.3.4. Comparaison de l'application des deux ensembles de modifications à l'étape 3 du modèle

Maintenant que les changements de la valeur d'epsilon pour les dernières synchronisations sont connus, nous pouvons déterminer quel ensemble de modifications de la valeur d'epsilon pour l'étape n°3 du modèle convient le mieux. C'est-à-dire sélectionner celui dont les distances domiciles-écoles sont les plus proches de la réalité même si elles ne suivront jamais la même loi de probabilité comme nous avons vu dans le point 8.4.3.2.

Les tests n°13 et le test n°14 sont à la base de cette comparaison. Les paramètres de ces tests sont stockés dans la table 4.33.

Paramètres	Test 13	Test 14
Modèle	Modèle 3	Modèle 3
Type de l'école	Primaire	Primaire
Nombre d'étudiants	1727134	1727134
Nombre de processeurs	100	100
Nombre de synchronisations	1200	1200
Ensemble des modifications d'epsilon pour l'étape 4 du modèle	Premier	Deuxième
Changement Epsilon pour les dernières synchronisations	Numéro de la synchronisation / Epsilon 1000 / 50 000 1100 / 100 000 1195 / 250 000	Numéro de la synchronisation / Epsilon 1000 / 50 000 1100 / 100 000 1195 / 250 000
Temps d'exécution	8h30	8h

TABLE 4.33 – Tests n°13 et n°14 pour trouver le meilleur ensemble de modifications de la variable epsilon à l'étape 3 du modèle

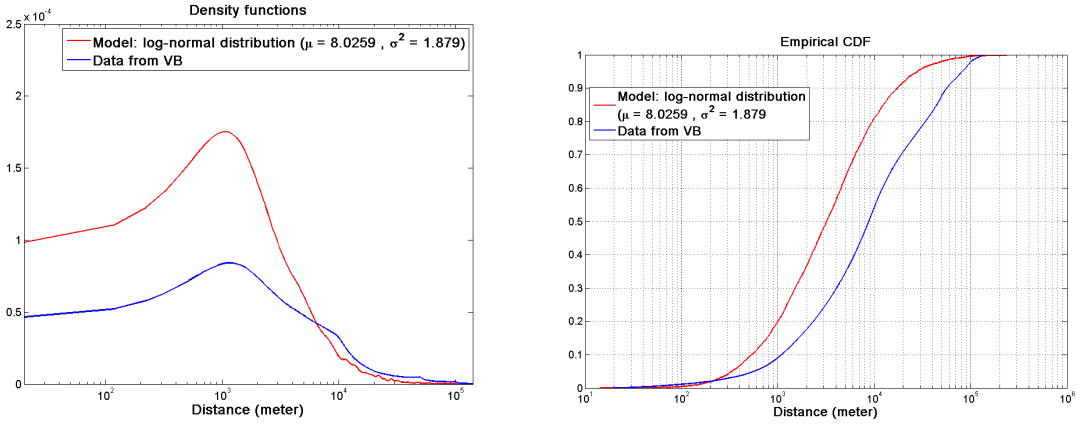
Dans l'ordre, nous allons examiner les distance Domicile-École, le temps d'exécution et la précision de chaque test.

- **Les distances :**

  Pour comparer ces distances, nous allons tout d'abord regarder la fonction de répartition créée avec les distances domiciles-écoles calculées avec le test n°15 réalisé sans aucune modification de la valeur epsilon, mis à part à la fin des synchronisations. Les paramètres de cet test sont stockés dans la table 4.34.

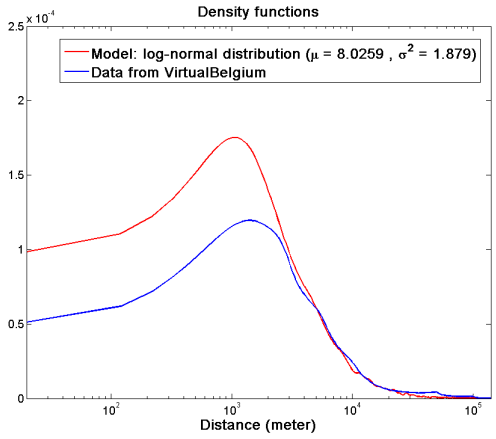
<table>
<tr><td>Paramètres</td><td>Test 15</td></tr>
<tr><td>Modèle</td><td>Modèle 3</td></tr>
<tr><td>Type de l'école</td><td>Primaire</td></tr>
<tr><td>Nombre d'étudiants</td><td>1727134</td></tr>
<tr><td>Nombre de processeurs</td><td>100</td></tr>
<tr><td>Nombre de synchronisations</td><td>1200</td></tr>
<tr><td>Ensemble des modifications d'epsilon</td><td>Auncun</td></tr>
<tr><td>Changement Epsilon</td><td>
<table>
<tr><td>Numéro de la synchronisation lors du changement</td><td>Epsilon</td></tr>
<tr><td>1000
1100
1195</td><td>50 000
100 000
250 000</td></tr>
</table>
</td></tr>
<tr><td>Temps d'exécution</td><td>5h 50</td></tr>
</table>

TABLE 4.34 – Tests n°15 du modèle 3

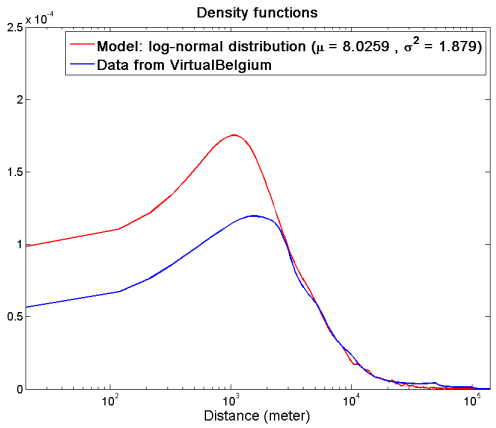


(a) Fonctions de densité (b) Fonction de réparition

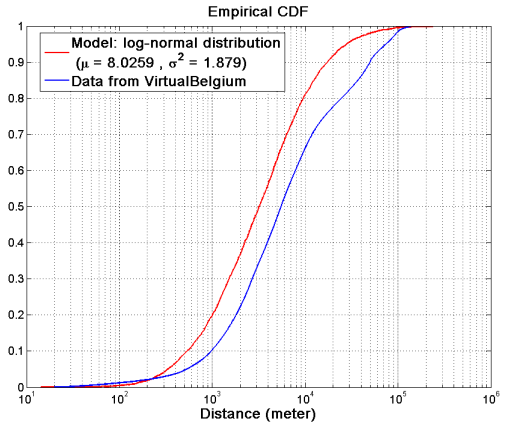
FIGURE 4.26 – Comparaison entre les distances Domicile-École pour le test n°15



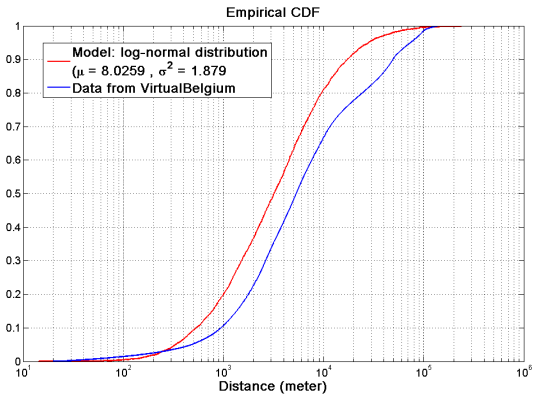
(a) Fonctions de densité : Premier ensemble de modifications



(b) Fonctions de densité : Deuxième ensemble de modifications



(c) Fonctions de répartition :Premier ensemble de modifications



(d) Fonctions de répartition : Deuxième ensemble de modifications

FIGURE 4.27 – Fonctions de répartition empiriques et théoriques des déplacements Domicile-École

Nous pouvons tout d'abord dire qu'il y a une amélioration de la distance Domicile-École entre le test sans aucune modification d'epsilon, excepté dans les dernières synchronisations et les deux autres tests avec modification. En ce qui concerne la comparaison entre les tests n°13 et n°14, nous pouvons voir dans un premier temps, qu'il n'y a presque pas de différences. Si nous regardons plus précisément, nous remarquons que la fonction de densité empirique du test n°14 est très légèrement plus proche de la fonction de densité de la loi log-normale demandée.

- Au niveau du temps d'exécution, le test n°14 est plus rapide.
- Nous pouvons trouver dans la table 4.35, les différents résultats des deux tests concernant la répartition des étudiants dans les écoles. Ces derniers sont quasiment identiques. Il n'y a pas de différence significative entre les deux tests.

Notions	Test 13	Test 14	Test 15
Pourcentage d'écoles dont le quota est atteint	83.1%	83.9 %	84.4%
Pourcentage d'écoles dont le quota a été dépassé	12.1%	11.95%	11.6%
Pourcentage d'écoles dont il reste au moins une place à attribuer	4.7%	4%	3.8%
Pourcentage d'écoles dont le quota a été dépassé de 5 étudiants	0%	0%	0%
Pourcentage d'écoles dont il reste au moins 5 places à attribuer	0.5%	0.6%	0.6%
Pourcentage d'étudiants qui s'inscrivent dans une école dont le quota déjà est atteint	0%	0 %	0%
Pourcentage de places dans les écoles qui ne sont pas occupées	0.04%	0.04%	0.04%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la moins remplie	93.16%	94.36 %	92%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la plus remplie	100.7%	101.78 %	101.3 %
Le nombre de places restantes en moyenne dans une école	0	0	0
L'écart-type du nombre de places restantes dans les écoles	1.49	1.49	1.51

TABLE 4.35 – Analyse des résultats des tests 13 à 15 du modèle 3

En conclusion, le test n°14 donne un meilleur résultat que le test n°13.

Nous avons essayé de modifier la valeur de la variable epsilon lors de l'étape 3 de ce modèle afin que les distances domiciles-écoles suivent la loi de probabilité log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.8792$. Malheureusement, les deux ensembles de modifications de la valeur d'epsilon n'ont pas donné les résultats escomptés malgré une légère amélioration nuancée par une augmentation du temps d'exécution en passant de 5h50 à 8h (8h30).

En conclusion, nous choisissons de ne pas modifier la valeur de l'epsilon de l'étape 3 car l'amélioration ne justifie pas les 2 heures supplémentaires. Un futur travail devra être consacré à cette problématique des distances afin de trouver une solution pour que les distances Domicile-École suivent la loi log-normale souhaitée.

8.5. Conclusion

Pour l'ensemble des étudiants allant dans une école primaire, si nous utilisons 100 processeurs avec 1200 synchronisations afin de pouvoir synchroniseur 8-20 étudiants par processeur, alors nous obtenons un bon modèle d'assignation même si toutefois, il ne respecte pas entièrement la contrainte 4.12 due à l'implémentation.

Nous affectons également ce modèle aux étudiants des autres types d'enseignement en effectuons les mêmes étapes dans le choix du nombre de synchronisations. Cependant, pour les étudiants de l'enseignement secondaire, il est assez difficile de réaliser des synchronisations car la répartition des étudiants de l'école secondaire à travers les processeurs n'est pas homogène. Un des processeurs possède 0 étudiant comparé à un autre qui en possède 402. Nous avons pu trouver une solution pour qu'il attende les autres processeurs qui sont en train d'assigner une école à leurs étudiants et ainsi tester le modèle avec 2 synchronisations.

Au final, 8-20 étudiants entre chaque synchronisation parait être un nombre idéal pour un grand nombre d'étudiants. Cependant si ce dernier est trop petit pour permettre autant d'étudiants par synchronisation, alors ce nombre devra être diminuer.

9. Quatrième modèle

9.1. Changement de la distribution de distance

L'hypothèse de domiciliation impose que la nouvelle distribution soit calculée seulement à partir des déplacements des individus entre leur domicile et l'école. Nous pouvons, améliorer la valeur des distances scolaires, en créant trois distributions à la place d'une unique. En effet, pour l'instant, la distribution a été créée avec tous les trajets dont la destination est un établissement scolaire, indépendamment du type de l'école. Tous les types d'écoles sont donc confondus dans cette distribution. Cependant, il nous parait plus juste et plus précis de réaliser trois distributions différentes en fonction du type de l'école. L'hypothèse suivante est donc émise :

Hypothèse 4.6. *La distance parcourue par les étudiants pour atteindre leur école dépend du type de cette dernière.*

Cette hypothèse a été relevée suite à une réflexion concernant les universités. En Belgique, il n'existe pas une université dans toutes les villes contrairement aux écoles

des enseignements primaire et secondaire. La plupart des étudiants universitaires doivent donc effectuer plus de kilomètres que si leur destination est une école secondaire. De plus, la même réflexion peut être envisagée pour les villageois qui effectuent leurs primaires à l'école du village mais qui se déplacent dans un endroit plus éloigné de leur domicile pour les études secondaires.

En conclusion, trois nouvelles distributions sont construites, une pour chaque type d'enseignement. Les différentes étapes effectuées pour réaliser ces dernières sont données dans le chapitre suivant. Cette hypothèse peut être confirmée par la figure ci-dessous :

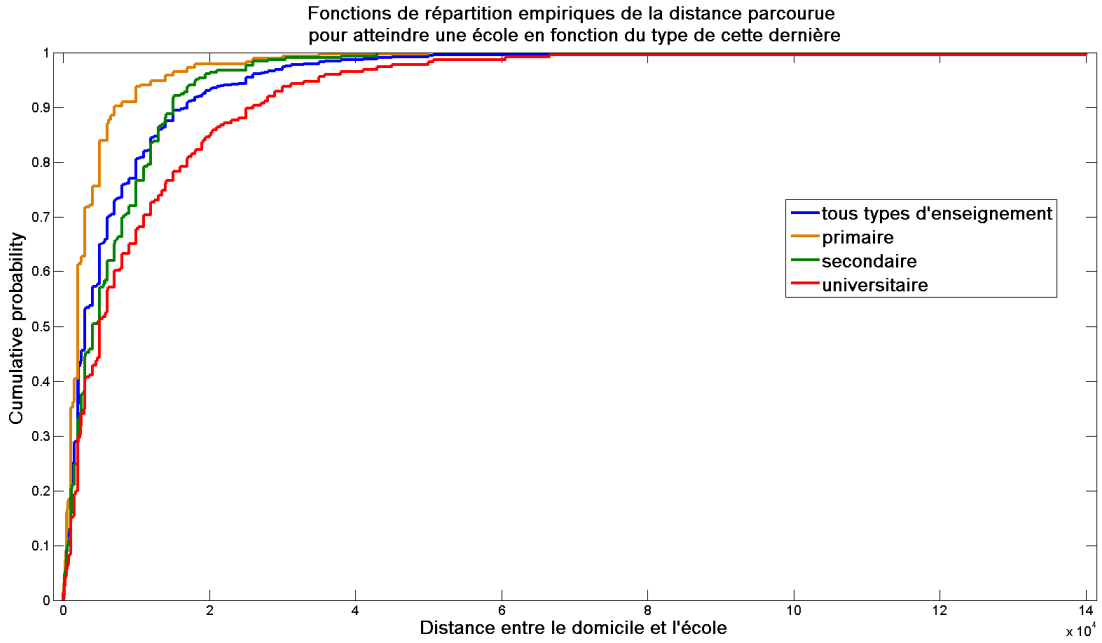


FIGURE 4.28 – Fonctions de répartition empiriques créées avec les données de MOBEL

Les informations relatives aux nouvelles distributions sont reprisent dans la table 4.36. La construction de ces dernières est détaillée dans la section 5.1.2 du chapitre 5.

Paramètres			
Distance	Domicile-Ecole	Domicile-Ecole	Domicile-Ecole
Type	Primaire	Secondaire	Supérieur
Loi de probabilité	Mixture de loi log-normale	Mixture de loi log-normale	Loi log-normale
μ	8.5301 et 6.7264	8.5301 et 6.7264	8.6715
σ^2	0.9385 et 2.6473	0.6385 et 2.6473	0.22123
p	0.80000039 et 0.199961	0.8 et 0.2	1
Maximum (mètres)	50 000	43 000	140 000
Minimum (mètres)	36	10	10

TABLE 4.36 – Information concernant la distribution Domicile-Ecole par type d'enseigement

Avec ces nouvelles distributions, nous devons revoir le point 2 des modèles, c'est-à-dire la détermination de la valeur de la distance. En effet, avant de générer une distance avec une distribution, nous devons choisir la bonne distribution. Chaque étudiant de VirtualBelgium est associé à un type d'enseignement en fonction de ses paramètres comme expliqué dans la section 4.3.2. Il en va donc de même avec les distributions : chaque étudiant est relié à une distribution. Pour connaitre la distance réalisée par l'étudiant pour atteindre son école, il suffit de suivre les différentes étapes citées ci-dessous :

1. Connaitre le type de l'école dans laquelle l'individu doit être assigné en suivant l'explication donnée dans la section 4.3.2
2. Choisir la distribution correspondante au type trouvé.
3. Générer une distance avec le générateur de nombres pseudo-aléatoires suivant cette distribution. Par exemple, si l'étudiant ne possède pas de diplôme, il va être dirigé vers une école primaire à une certaine distance générée par le générateur de nombres pseudo-aléatoires suivant une mixture de lois log-normales à deux composantes (μ = 8.5301 et 6.7264 , σ^2 = 0.9385 et 2.6473 ,p = 0.80000039 et 0.199961)

9.2 Comparaison

Dans cette partie, nous comparons l'impact de la distribution utilisée dans les tests précédents, c'est-à-dire celle calculée sans tenir compte du type d'enseignement et la nouvelle distribution sur le modèle n°3. Les différents paramètres utilisés pour cette comparaison sont repris dans la table 4.37. Ce test a été réalisé avec le premier ensemble de modifications de la valeur d'epsilon pour l'étape 3 du modèle. Effectivement, ce test a été

créé avant de prendre une décision quant à l'importance de la modification de la valeur d'epsilon. L'utilisation de cet ensemble de modifications n'influence pas la conclusion de ce point.

<table>
<tr><th>Paramètres</th><th colspan="2">Test 13</th><th colspan="2">Test 16</th></tr>
<tr><td>Modèle</td><td colspan="2">Modèle 3</td><td colspan="2">Modèle 3</td></tr>
<tr><td>Type de l'école</td><td colspan="2">Primaire</td><td colspan="2">Primaire</td></tr>
<tr><td>Distribution des distances</td><td colspan="2">Distance Domicile-École</td><td colspan="2">Distance Domicile-École -primaire</td></tr>
<tr><td>Nombre d'étudiants</td><td colspan="2">1727134</td><td colspan="2">1727134</td></tr>
<tr><td>Nombre de processeurs</td><td colspan="2">100</td><td colspan="2">100</td></tr>
<tr><td>Nombre de synchronisations</td><td colspan="2">1200</td><td colspan="2">1200</td></tr>
<tr><td>Ensemble des modifications d'epsilon à l'étape 3</td><td colspan="2">Premier</td><td colspan="2">Premier</td></tr>
<tr><td>Changement Epsilon</td><td>Numéro de la synchronisation</td><td>Epsilon (mètres)</td><td>Numéro de la synchronisation</td><td>Epsilon (mètres)</td></tr>
<tr><td></td><td>1000
1100
1195</td><td>50 000
100 000
250 000</td><td>1000
1100
1195</td><td>50 000
100 000
250 000</td></tr>
<tr><td>Temps d'exécution</td><td colspan="2">8h30</td><td colspan="2">16h</td></tr>
</table>

TABLE 4.37 – Paramètres pour les tests 13 et 16

Notions	Test 13	Test 16
Pourcentage d'écoles dont le quota est atteint	83.1%	82.8 %
Pourcentage d'écoles dont le quota a été dépassé	12.1%	12.9%
Pourcentage d'écoles dont il reste au moins une place à attribuer	4.7%	4.1%
Pourcentage d'écoles dont le quota a été dépassé de 5 étudiants	0%	0%
Pourcentage d'écoles dont il reste au moins 5 places à attribuer	0.5%	0.6%
Pourcentage d'étudiants qui s'inscrivent dans une école dont le quota déjà est atteint	0%	0 %
Pourcentage de places dans les écoles qui ne sont pas occupées	0.04%	0.04%
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la moins remplie	93.16%	92.89 %
Pourcentage d'étudiants par rapport au nombre souhaité dans l'école la plus remplie	100.7%	101.2 %
Le nombre de places restantes en moyenne dans une école	0	0
L'écart-type du nombre de places restantes dans les écoles	1.49	1.67

TABLE 4.38 – Analyse des résultats des tests 16 et 17 du modèle 3

En regardant la table 4.38, nous ne remarquons pas de différences significatives. On remarque une légère amélioration négligeable de 0.3 % pour le test n°13 en ce qui concerne le nombre d'écoles respectant la contrainte 4.12. En ce qui concerne le temps d'exécution, nous pouvons voir qu'il a presque doublé entre les deux tests. Ceci est dû à l'utilisation de la distribution des distances liée à enseignement primaire qui impose que ces dernières soient comprises entre 36 m et 50 000 m . Or nous avons déjà remarqué qu'il était difficile de trouver une école proche du domicile. Maintenant que nous réduisons la distance maximum de 140000m à 50000m, cela est encore plus difficile, ce qui entraine de nombreuses générations de distances et donc une augmentation non négligeable du temps d'exécution. Il en va de même pour l'enseignement secondaire avec une réduction de la borne supérieure de 140 000 m à 43 000 m.

Nous choisissons donc de ne pas utiliser les nouvelles distributions de distances. En effet, l'utilisation de ces dernières augmente considérablement le temps d'exécution tout

en n'améliorant pas le modèle. De plus, pour l'instant nous n'avons pas de données réelles nous permettant de comparer notre distribution des étudiants dans les écoles et donc aucune comparaison pour justifier l'emploi de ces nouvelles distributions bien qu'elles paraissent intuitivement utiles pour le modèle.

4.3.4.3 Perspectives

Au cours de ce travail, d'autres modèles ont été trouvés. Certains ne pouvaient se réaliser suite à un manque de données ou une demande trop importante de calculs des distances. D'autres n'ont pas pu être implémentés dans ce travail suite à un manque de temps. Dans cette section, nous allons énumérer tous les pistes auxquelles nous avons pensé.

1. Une deuxième enquête de mobilité a été effectuée par le Groupe de Recherche sur les Transports, appelée Beldam, effectué en 2011. Grâce à ces données, nous avons pu récupérer différentes informations qui pourraient être utiles pour la suite de ce travail.

   - Dans un premier temps, nous avons pu calculer le pourcentage de ménages dont les étudiants primaires qui le composent vont suivre des cours dans la même école. Ce dernier équivaut à 86.1%. En d'autres termes, une famille qui possède un étudiant âgé de 9 et un autre, âgé de 6 ans, a 86.1% de chance de placer son deuxième enfant dans la même école que son premier enfant. Ce pourcentage peut être intégré dans la plateforme afin de mieux comprendre le choix des écoles par les parents.

     Cette recherche a été motivée par le fait qu'en réalité, les parents mettent souvent leurs enfants dans la même école pour éviter de se déplacer vers différents écoles.

   - Les modèles testés dans ce travail ne tiennent compte seulement de la proportionnalité des écoles pour effectuer le choix de l'école. Une réflexion a donc été faite concernant les écoles de la commune où l'étudiant est domicilié afin de savoir s'il a plus de chance de suivre des cours dans sa propre commune comparée aux autres communes. Grâce aux données de l'enquête, nous pouvons connaitre la probabilité qu'un étudiant va dans une écoles de sa commune. Cette dernière est égale à 0.48

     Cette probabilité pourrait être incluse dans le modèle n°3, présenté dans la section 4.3.4.2 lors du calcul des probabilités de choisir les écoles sélectionnées dans l'étape 4 du modèle.

2. Dans notre modèle, les écoles sont déterminées en fonction du domicile de l'étudiant. Cependant, certains parents préfèrent mettre leur enfants dans des écoles proches de leur lieu de travail. Dans un futur, il serait intéressant de récolter l'information concernant ces ménages et inclure ce paramètre dans le modèle d'assignation.

3. Pour déterminer le choix de l'école à l'étape 5 du modèle n°3, nous avons pensé à utiliser une autre méthode pour remplacer la notion de proportionnalité et permettre de tenir compte de différents paramètres relatifs à l'étudiant. Cette méthode s'appelle le choix discret et permet de déterminer le choix d'une personne par rapport à un ensemble d'alternatives en calculant la probabilité de choisir chaque alternative. Pour plus d'information concernant cette méthode, veuillez consulter le site [36].

   Cette méthode n'est malheureusement pas utile dans notre travail car pour déterminer la probabilité, il faut tout d'abord des données réelles concernant les choix des écoles. De plus, même si les résultats de l'enquête de Beldam nous permet d'obtenir les données nécessaires, il nous faudrait un nombre précis d'alternatives, ce qui n'est pas possible vu que l'ensemble de noeuds sélectionnés à l'étape 4 est toujours différents .

4. Différents paramètres peuvent être intéressants à inclure dans le modèle d'assignation d'une école. Cependant le manque de données réelles ou même au sein de VirtualBelgium nous empêche d'améliorer le modèle. Par exemple, géocoder les écoles, nous permettrait d'inclure la positon de ces dernières dans les probabilités. Ainsi, si une école est plus proche de la gare ou d'un arrêt de bus, elle a peut être plus de chance d'être choisie en comparaison à une école non accessible en transport en commun.

   D'autres informations concernant les écoles devraient être récoltées telles que le coût financier et/ou la réputation, De plus dans VirtualBelgium, nous ne connaissons pas le niveau socio-économique du ménage, ni son budget. Ces paramètres pourraient également influencer le modèle d'assignation

5. Au début de ce travail, les étudiants de VirtualBelgium ne possédaient pas un âge précis mais étaient regroupés dans des classes d'âges. Nous avons donc classé les individus seulement par établissement scolaire.

   Depuis peu, les individus sont caractérisés par un âge précis. Il serait donc intéressant, dans un futur travail, de récolter le nombre d'étudiants dans les différentes années d'études par établissement et d'adapter le modèle à cette nouvelle informa-

tion. Dès lors, on pourra déterminer l'année d'étude des individus.

6. Une étude concernant l'assignation d'une école a déjà réalisée par Millingstom, Butler, Hammett[26]. Cependant, ces derniers possédaient plus de paramètres pour la création du modèle. En effet, ils tiennent compte, entres autres, du grade obtenu par l'étudiant, c'est-à-dire de son pourcentage de réussite. Cependant, nous ne pouvons avoir assez à cette donnée actuellement.

4.3.5 Conclusion

Dans ce travail, nous avons pu trouvé un modèle d'assignation qui permet de distribuer les étudiants dans les écoles virtuelles tout en tenant compte du quota d'étudiants maximum dans ces dernières. Malheureusement, en raison de son implémentation dans un programme en programmation parallèle, il ne peut totalement respecter le nombre souhaité d'étudiants dans les écoles. Différents tests ont été effectués afin de comprendre l'influence de l'implémentation dans les résultats du modèle.

Le modèle choisi est le modèle n°3, c'est-à-dire un modèle qui tient compte du nombre d'étudiants inscrits dans les écoles tout au long du processus en sachant qu'il respecte pas la distribution créée à partir des distances Domicile-École parcourues par les étudiants belges. Les paramètres de ce dernier sont stockés dans la table 4.39.

Paramètres			
Modèle	Modèle 3	Modèle 3	Modèle 3
Type de l'école	Primaire	Secondaire	Supérieur
Distribution des distances	Distance Domicile-École	Distance Domicile-École	Distance Domicile-École
Nombre d'étudiants	1727134	40353	435980
Nombre de processeurs	100	100	100
Nombre de synchronisations	1200	2	300
Ensemble des modifications d'epsilon à l'étape 3	Aucun	Aucun	Aucun
Changement Epsilon	Numéro de la synchronisation / Epsilon (mètres) 1000 / 50 000 1100 / 100 000 1195 / 250 000	Numéro de la synchronisation / Epsilon (mètres) 1 / 250 000	Numéro de la synchronisation / Epsilon (mètres) 250 / 50 000 275 / 100 000 295 / 250 000

TABLE 4.39 – Paramètres finaux pour le modèle 3

4.4 Conclusion

Dans ce chapitre, nous avons détaillé les deux modèles nécessaires à la modélisation du choix des écoles pour les étudiants dans VirtualBelgium. Le premier consistait à créer un réseau d'écoles en tenant compte des données récoltées dans le chapitre 3 et le second à assigner ces écoles aux étudiants virtuels. Différents tests ont été réalisés afin d'en retenir le meilleur. Cependant, nous disposons de peu de données pour réaliser ce choix. Il dépend donc pour l'instant que de la proportionnalité du nombre d'étudiants inscrits.

Le modèle ne respecte pas entièrement la contrainte 4.12 qui impose que le nombre d'étudiants inscrits dans chaque école doit être égale à son quota car il y a des marges d'erreur dû à l'implémentation du modèle en programmation parallèle.

Dans un futur travail, il serait intéressant de récolter plus de données concernant les écoles dont le nombre d'étudiants par années d'études mais également de permettra à la population synthétique de s'enrichir en incluant des données socio-économiques et budgétaires dans les ménages. Il serait également important de géocoder les écoles afin de pouvoir tenir compte de l'emplacement des écoles.

Chapitre 5

Etablissements scolaires dans VirtualBelguim : Partie pratique

Maintenant que les différents modèles créés sont expliqués ainsi que leurs résultats, les différentes méthodes d'implémentation sont détaillées pour les personnes désirant comprendre la construction des distances ou utiliser le programme VirtualBelgium.

Le programme écrit dans le logiciel MATLAB, conçu pour créer les différentes distributions de distances utiles à ce travail, est détaillé dans la suite, suivi des explications de l'implémentation du modèle de distribution des écoles. Pour conclure, l'implémentation finale du modèle d'assignation d'une école à un étudiant est décrite, ainsi que toutes les méthodes d'implémentation testées pour ce modèle mais qui n'ont pas données de résultats satisfaisants.

Toutefois, pour comprendre ce chapitre, le lecteur doit posséder certaines connaissances du logiciel MATLAB et du langage de programmation orienté objet C++. Les différents principes de la programmation orientée objet sont expliqués dans l'annexe E. Il est donc souhaitable que cette annexe soit parcourue avant de commencer la lecture de ce chapitre.

Le programme lié à la plateforme VirtualBelgium peut être demandé au Groupe de Recherche sur les Transports de l'Université de Namur.

5.1 Construction des distributions des distances

Dans la chapitre 4, nous avons modifié la distribution de la distance pour l'activité « École » due à l'hypothèse de domiciliation expliquée dans la section 4.3.3. Dans cette section, le programme utilisé pour la création de cette distribution à partir des trajets « Domicile-École » est détaillé. En complément, nous avons également testé des modèles

qui utilisaient des distributions de distances en fonction du type d'enseignement. La méthode pour les construire sera expliquée également.

5.1.1 Distance "Domicile-Ecole" indépendamment du type d'enseignement

Pour construire la distribution de la distance entre le domicile et l'école qui respecte l'hypothèse de domiciliation, nous utilisons les données récoltées lors de l'enquête Mobel [1]. Ces informations sont filtrées afin de permettre de garder uniquement les déplacements des belges entre leur domicile et leur école. Sur 1270 déplacements partant d'un domicile, provenant de l'enquête, nous obtenons 738 trajets dont la destination finale est l'école où l'individu suit ses cours.

Avec ces déplacements, nous construisons une fonction de densité et une fonction de répartition empiriques [2]. Pour créer une distribution valide pour le programme Virtual-Belgium, appelée « distribution Domicile-École », nous devons approximer ces fonctions par une loi de probabilité. Pour choisir la loi, nous nous inspirons de la distribution des distances scolaires réalisée par J. Barthelemy, Ph Toint et E. Cornelis, expliquée dans le deuxième chapitre. En effet, les déplacements utilisés pour la distribution Domicile-École sont inclus dans les données traitées par J. Barthelemy et Ph Toint.

Ces derniers ont approximé la fonction de répartition basée sur les déplacements scolaires indépendamment de lieu précédent, par une loi log-normale de paramètre $\mu = 7.9579$ et $\sigma = \sqrt{(1.926)}$. Nous pouvons donc supposer que la distribution Domicile-École va également suivre une loi log-normale mais avec des paramètres légèrement différents.

Pour confirmer cette hypothèse, nous approximons la courbe des échantillons avec la fonction « longfit » du logiciel MATLAB. Cette dernière ressort les paramètres μ et σ de la loi log-normale. Pour approuver cette modélisation, nous devons effectuer certains tests. Ces derniers consistent en :

1. Le test du χ^2 avec la fonction « chi2gof » qui permet de tester si l'échantillon suit une certaine loi de probabilités donnée grâce à un test d'hypothèse où
   H_0 la fonction de densité empirique suit la loi de probabilité donnée
   H_1 la fonction de densité empirique ne suit pas la loi de probabilité donnée

   Ce test organise les données par classe et vérifie la correspondance entre les effectifs empiriques donnés par l'échantillon et les effectifs théoriques donnés par le loi de

1. Cette enquête a déjà été introduite dans le premier chapitre.
2. La création de ces deux fonctions a déjà été expliquée dans la section 2.2.2

probabilité. Les classes doivent être définies telles que l'effectif théorique soit supérieur à 5 pour chaque classe. Pour utiliser ce test, il faut faire attention que la taille de l'échantillon soit supérieur à 30.

Ce test consiste à calculer la probabilité d'avoir pris une mauvaise décision en rejetant H_0 :

Soient :	n	nombre de classes
	χ^2_{n-1}	la loi du χ^2 à n-1 degrés de liberté
	$N_{empirique_i}$	Nombre d'éléments de l'échantillon se retrouvant dans la $i^{\text{ième}}$ classe.
	$N_{theorique_i}$	Nombre d'éléments suivant la loi donnée se retrouvant dans la $i^{\text{ième}}$ classe.

$$P_{H_0}(\chi^2_{n-1} >= \sum_{i=1}^{n} \frac{(N_{empirique_i} - N_{theorique_i})^2}{N_{theorique_i}})$$

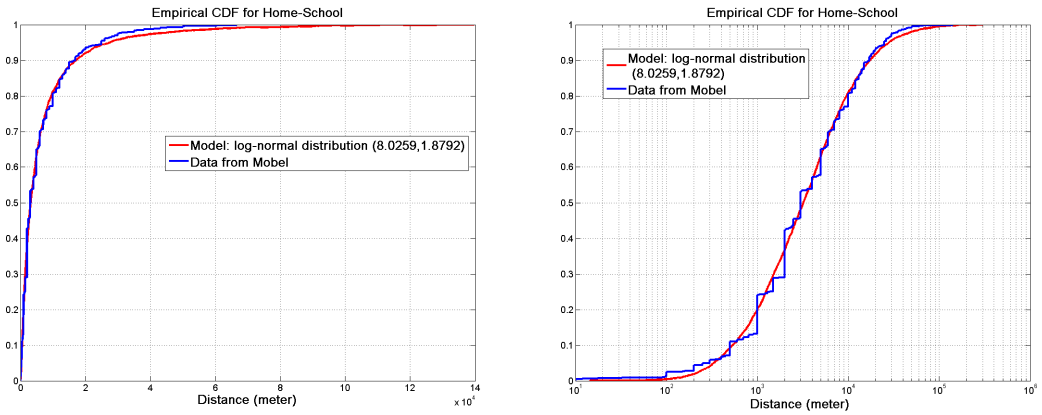
Si la probabilité est plus petite que 0.05 alors le test est rejeté et donc l'hypothèse nulle (H_0) ne peut être acceptée avec un seuil de confiance de 5%, c'est-à-dire que dans 95% des cas, le test n'est pas accepté. Sinon, l'échantillon suit la loi donnée.

Dans notre cas, cela consiste en un test d'hypothèse où H_0 signifie que les déplacements suivent une loi log-normale dont les paramètres sont identiques à ceux trouvés avec la fonction « lognfit ». Nous pouvons également utiliser ce test en prouvant que le logarithme népérien de chaque valeur de l'échantillon suit une loi normale [3] dont les paramètres sont identiques à ceux trouvés avec la fonction « lognfit ».

2. Le test de Kolmogorov avec la fonction « kstest » consiste également en un test d'hypothèse qui vérifie si un échantillon suit bien une loi connue donnée par la fonction de répartition. Ce test s'intéresse à l'écart entre la fonction de répartition empirique et celle théorique. Pour utiliser ce test, la loi donnée doit être continue et ne doit pas posséder des paramètres inconnus. Nous ne détaillons pas la statistique dans ce travail mais pour plus d'information, vous pouvez consulter la référence [8].

L'application de la fonction « lognfit » sur les déplacements Domicile-École donne les paramètres $\mu = 8.0259$ et $\sigma^2 = 1.8792$. Nous pouvons voir à la figure suivante que les fonctions de répartition empiriques et théoriques sont assez proches. Pour confirmer l'équivalence au point de vue statistique, nous effectuons les tests. Les résultats de ces derniers sont stockés dans la table 5.1.

3. Une variable aléatoire X suit une loi log-normale de paramètres μ, σ^2, si la variable aléatoire Y=Ln(X) suit une loi normale de paramètres μ et σ^2.



(a) Distance (b) Distance (Échelle logarithmique)

FIGURE 5.1 – Fonctions de répartition empiriques et théoriques des déplacements Domicile-École

Tests	Résultat	P-valeur
Chi carré	Accepté avec un niveau de test de 5 % avec un petit ensemble de déplacements	0.4632
KS test	Accepté avec un niveau de test de 5% avec un petit ensemble de déplacements	0.1513

TABLE 5.1 – Résultat de la modélisation de la distribution des déplacements Domicile-École avec les paramètres $\mu = 8.0259$ et $\sigma^2 = 1.8792$ d'une loi log-normale

Nous pouvons remarquer dans la table 5.1 que les tests effectués sont valides pour un petit ensemble de données, par exemple 150 déplacements mais pas pour l'ensemble des 738 déplacements. Nous ne pouvons donc pas confirmer que la modélisation est acceptée. Un autre test doit donc être réalisé.

Ce nouveau test est identique au test de Kolmogorov-Smirnov mais à la place d'analyser une fonction de répartition empirique avec une fonction de répartition théorique d'une loi de probabilité, il compare deux distributions empiriques. Dans le logiciel MATLAB, cela se résume à utiliser la fonction kstest2 à la place de kstest. En d'autres mots, pour effectuer ce test, nous devons créer une distribution empirique via une loi de probabilité log-normale de paramètres $\mu = 8.0259$ et $\sigma^2 = 1.8792$. Dans notre cas, cela signifie faire appel à la fonction « random » pour obtenir des nombres pseudo-aléatoires suivant cette loi.

Ensuite, on utilise la fonction « kstest2 » pour vérifier si les deux distributions empiriques, celle créée avec les données récoltées et celle créée avec la fonction « random »

sont statistiquement identiques avec un niveau de test équivalent à 1%. Le résultat de ce test est concluant avec une p-valeur égale à 0.0667.

Nous pouvons donc conclure que les déplacements Domicile-École peuvent être modélisés par une loi log-normale de paramètre $\mu = 8.0259$ et $\sigma^2 = 1.8792$.

5.1.2 Distance "Domicile-Ecole" dépendant du type d'enseignement

Comme nous avons pu le voir dans le chapitre précédant, un modèle a été créé en utilisant trois distributions différentes correspondant aux trois types d'enseignement : primaire, secondaire et supérieur. En effet, la distance moyenne parcourue peut varier en fonction du type. Pour prouver cette hypothèse, nous avons tracé les fonctions de répartitions empiriques des distances entre le domicile et l'école réciproquement pour les trois sortes d'établissements scolaires. Ces dernières ont été s analysés dans la section du chapitre précédent mais pour plus de facilité, elles sont reprises dans le graphe 5.2.

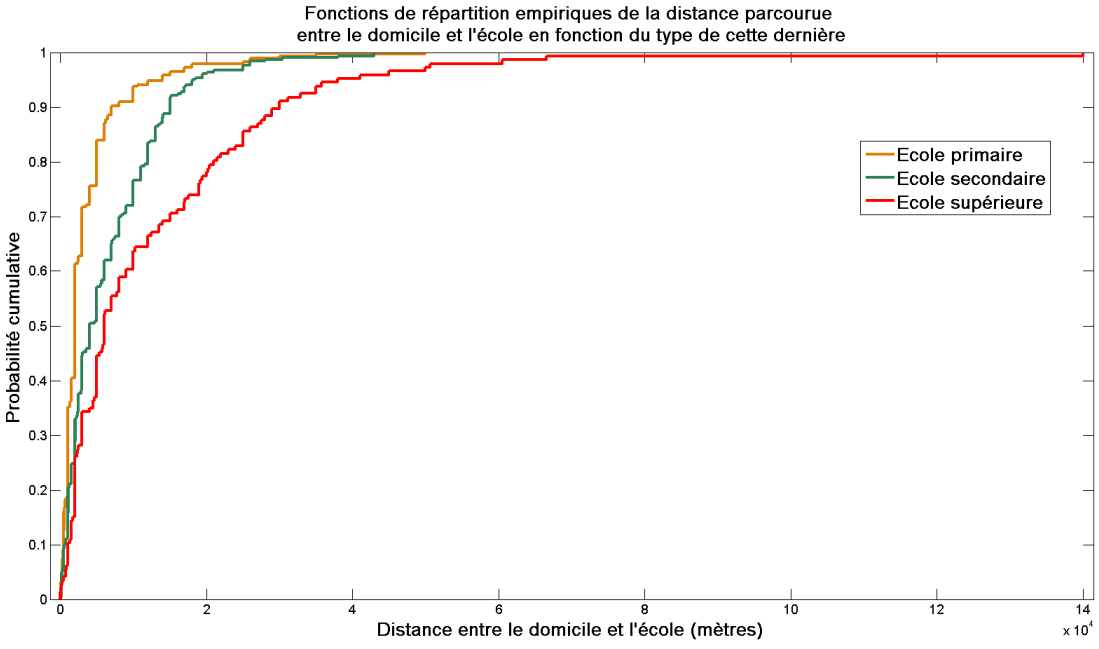


FIGURE 5.2 – Fonctions de répartition empiriques des distances parcourues entre le domicile et l'école en fonction du type de cette dernière

Ces fonctions ont été créées avec les données de l'enquête Mobel. La table 5.2 reprend toutes les informations concernant ces dernières.

Paramètres	Primaire	Secondaire	Supérieur
Nombre d'effectif	287	303	146
Maximum (mètres)	50000	43000	140000
Minimum (mètres)	32	10	10
Moyenne	3625.6	6623.0657	12575.7
Ecart-type	5455	6875.3	16679.9

TABLE 5.2 – Données utilisées pour créer les trois distributions de distances dépendantes du type d'enseignement

Maintenant que nous avons toutes les informations nécessaires concernant les trois ensembles de données, nous essayons de modéliser ces distributions. Tout d'abord, nous avons également essayé la fonction « lognfit » pour obtenir les paramètres de la distribution théorique modélisant ces ensembles. Nous l'avons testé avec le test de chi-carré et le premier test de Kolmogorov. Cependant, aucun n'a pu être accepté. Nous pouvons donc supposer que les échantillons suivent une mixture de loi log-normale à plus d'un paramètre. Nous devons donc utiliser fonction « gmdistribution.fit ». En choisissant cette fonction, il faut savoir que le paramètre σ n'est pas l'écart-type mais une matrice de covariance entre les variables aléatoires de la mixture.

Les différents résultats de la modélisation de la fonction de densité par une loi mixture de log-normale sont représentés dans la table 5.3.

	Nombre de composantes	Paramètre μ	Paramètre σ	P	p-valeur
Primaire	2	8.5301 et 6.7264	0.9385 et 2.6473	0.80 et 0.2	0.0186
Secondaire	2	8.5301 et 6.7264	0.6385 et 2.6473	0.8 et 0.2	0.2078
Supérieur	1	8.6715	0.22123	1	0.3957

TABLE 5.3 – Résultat du test de Kolmogorov-Smirnov pour les trois nouvelles distributions

5.2 Implémentation du modèle de distribution des écoles

Avant d'appliquer le modèle d'assignation des écoles, nous devons créer des écoles dans VirtualBelgium grâce aux données récoltées avec le modèle de distribution. Ce dernier est expliqué dans la section 4.2.

Pour rappel, ce modèle impose deux conditions :

- Le nombre d'écoles virtuelles par commune correspond aux données récoltées contenues dans les trois fichiers, construits dans le chapitre 3.
- Chaque école virtuelle est associée à une école réelle décrite dans les données. Un quota d'inscriptions est attribué à chacune d'entre elles. Ce nombre représente le nombre d'étudiants que l'école souhaite accueillir.

5.2.1 Détails de l'implémentation

La première étape de la modification du programme VirtualBelgium pour exécuter le modèle de distribution consiste à l'ajout de propriétés dans le fichier bin/model.props. Ce dernier contient les différents paramètres destinés à la configuration du programme. Il dispose également de variables contenant les noms des fichiers nécessaires à l'introduction des données dans ce dernier.

Les propriétés additionnées sont trois variables dont la valeur respective est le nom du fichier contenant les informations sur les écoles d'un type d'enseignement. Les lignes de programmation représentant ces variables sont listées ci-dessous.

```
file.primary_school=../data/activity/primarySchool.txt
file.secondary_school=../data/activity/secondarySchool.txt
file.higher_school=../data/activity/higherSchool.txt
```

La deuxième étape consiste à implémenter le programme en utilisant la classe Data. En d'autres mots, cela revient à créer une méthode [4] dans la classe Data. Avant d'expliquer cette nouvelle méthode, nous devons comprendre le fonctionnement de cette classe.

La classe Data est une classe Fille [5] de la classe Singleton<Data>. La classe Singleton est en réalité implémentée en utilisant un Template T, une particularité de la programmation en C++. L'utilisation de ce dernier permet à la classe d'être implémentée sans connaitre le type T lors de la compilation. Il sera connu au moment de l'exécution. Dans notre cas, le type T est égal à la classe Data. La classe Singleton possède un constructeur avec le niveau de visibilité « protected » qui impose une forte condition sur ses instances. En effet, cela signifie que la classe ne peut être instanciée que par des instances de cette même classe ou par ses propres méthodes. La classe possède donc trois méthodes qui permettent de créer l'unique instance.

4. En programmation orientée objet, une fonction d'une classe est appelée une méthode
5. L'expression « classe Fille » fait appel au concept d'héritage. Il y a la classe Mère et la classe fille. Cette dernière hérite des variables et méthodes de la classe Mère. Pour plus d'information, l'annexe E est mise à votre disposition.

Singleton<T>
Variable : static T * _singleton = NULL
Méthode : static void makeInstance (repast : :Properties aProps) static T * getInstance () static void kill ()

Selon le principe d'héritage, allant de la classe Singleton< Data > vers la classe Data, cette dernière dispose de ses propres méthodes mais également des méthodes de la classe Singleton, adaptées au type Data. Cet héritage permet donc à la classe Data de ne s'instancier qu'une seule fois par processeur en faisant appel à la méthode « makeInstance ». Cette instance ne pourra être récupérée qu'avec la méthode « getInstance ».

En résumé, pour chaque processeur utilisé pour l'exécution du programme, une seule instance de la classe Data peut être créée. Cela permet d'éviter d'instancer plusieurs fois la classe Data et surtout de pouvoir récupérer la même instance à chaque moment sans passer par des solutions plus lentes.

Lors de l'instanciation de cette classe, son constructeur est directement exécuté. Ce dernier regroupe des instructions servant à initialiser certaines variables de la classe. Par exemple, la variable _network est initialisée suite à l'appel de la méthode read_network() lors de l'exécution du constructeur, en d'autres mots lors de l'instance de la classe Data. Cette variable est une instance de la classe Network et représente le réseau de VirtualBelgium avec des noeuds et des liens comme expliqué dans le premier chapitre.

Nous faisons donc de même dans l'implémentation du modèle de distribution. Nous créons une méthode dans la classe Data, nommée read_school. Cette dernière est appelée lors de l'exécution du constructeur. Ainsi dès la création de l'instance de la classe Data, le réseau est créé avec les écoles.

Cette méthode read_school() est divisée en trois grosses parties correspondant aux trois types d'enseignement. Ces parties sont relativement identiques. L'explication suivante porte seulement sur les écoles primaires car les trois parties sont relativement identiques.

Les étapes :

- **Récupération du nom du fichier contenant les informations sur les écoles primaires belges :**

```
string filenamePS = this->_props.getProperty("file.primary_school");
ifstream filePS(filenamePS.c_str(), ios::in);
```

- **Lecture du fichier ligne par ligne :**
  Chaque ligne du fichier représente une école primaire réelle. La variable « id », « ins », « fase » et « studentCount » désigne réciproquement, l'identifiant de l'école, le code INS de la commune dans laquelle elle se situe, son numero de fase et le nombre d'étudiants que l'école virtuelle qui la représentera pourra accueillir.

```
//Variables locales

string a_line;
vector<int> data;
int id,ins, fase,studentCount;


//Lecture du fichier ligne par ligne et récolte des informations telles
//que l'identifiant de l'école, son code INS, son numéro de fase et
//le nombre souhaité d'étudiants

while (getline(filePS, a_line))
       { data = split<int>(a_line, ";");
         id  = data[0];
         ins = data[1];
          fase = data[2];
        studentCount=data[3];
       .
       .
       .
      }
else {cerr << "Could not open file " << filePS ;}
```

- Pour chaque ligne (école), il faut sélectionner un noeud dans la commune déterminée grâce au code INS. Le noeud est sélectionné aléatoirement parmi tous ceux de la commune grâce à la méthode getOneNodeIdFromInsSameSeed(int ins) de la classe Data.

  Cette dernière méthode a dû être créée. En effet, la méthode getOneNodeIdFromIns(int ins), déjà existante, utilise un générateur aléatoire dont la graine (seed) est différente pour chaque processeur. Elle est déterminée en fonction du numéro du processeur. Or, si on utilise un générateur avec une graine différente pour chaque processeur, on va avoir un problème avec l'emplacement des écoles. En effet, si on utilise ce générateur, les écoles seront distribuées de manière aléatoire dans chaque processeur et donc à des emplacments différents. Or cela n'est pas pertinent dans notre modèle puisque la localisation de l'école est importante.

Une nouvelle classe, notée RandomGeneratorsSameSeed est donc créée, dont le constructeur initialise tous les générateurs avec une graine indépendamment du numéro du processeur . En d'autres termes, des générateurs qui ressortent la même suite de nombres pour tous les processeurs. La nouvelle méthode getOneNodeIdFromInsSameSeed est identique à la méthode getOneNodeIdFromIns, exceptée dans le choix du générateur.

```
while (getline(filePS, a_line))
  { data = split<int>(a_line, ";");
    id  = data[0];
    ins = data[1];
    fase = data[2];
    studentCount=data[3];



// l'id du noeud qui abrite l'école
long id_node;

// On sélectionne un noeud aléatoirement parmi la commune dont le
//code INS est "ins"
id_node=getOneNodeIdFromIns(ins);

// On reprend le réseau de VirtualBegium
Network net  = getNetwork();

//On prend l'ensemble des noeuds du réseau. Ils sont classés en
//fonction de leur id.
map<long,Node> nodes= net.getNodes();


// le noeud qui abrite l'école
Node *node;

// On récupère le noeud en fonction de son id
node=&nodes[id_node];

.
.
.
  }
```

- Maintenant que le noeud qui abritera l'école est connu, nous devons trouver un moyen de représenter cette école dans le programme VirtualBelgium.

  Pour réaliser cette étape, nous utilisons deux des principes de la programmation orientée objet : la délégation des tâches et l'utilisation de classes pour représenter des éléments ayant la même structure interne et les mêmes taches à réaliser. En d'autres mots, on ne va pas représenter les écoles avec une variable de la classe Data mais on va faire appel à une méthode afin que la classe Data ne fasse pas tout le travail. C'est dans cette optique qu'une classe « School » est créée ainsi qu'une méthode dans la classe Node qui construit une instance de la Classe « School ». Toutes les écoles de VirtualBelgium vont donc être représentées par des objets de

cette classe.

Dans la méthode read_school, l'appel de la méthode classe Node est :

```
(*node).becomePrimarySchool(id,fase, studentCount);
```

La création de la classe « School » va être expliquée ci-dessous suivie par celle de la méthode « becomePrimarySchool ».

a) **La Classe « School »**

Tout d'abord, afin d'être le plus général possible, nous créons une classe nommée Business qui représentera toutes les activités qu'un noeud peut abriter. Par exemple, un magasin ou une salle de gym peuvent être représentés par un objet de la classe Business. Ses variables et ses méthodes sont expliquées dans la table 5.4.

Business
Variable :
Méthode : virtual long getIdNode()=0 ;

TABLE 5.4 – La classe Business

Nous pouvons voir que la méthode de la classe Business est particulière. En effet, sa méthode est abstraite, ce qui signifie que cette classe ne peut être instanciée. Lorsqu'une classe hérite de cette classe alors elle est dans l'obligation d'implémenter la méthode. Nous utilisons ce principe de classe abstraite afin de généraliser les futures classes créées pour représenter les lieux des activités et permettre de les regrouper facilement en une classe. De plus, cela permet d'obliger aux instances de ces futures classes d'être créées seulement si les lieux sont associés à un noeud et de connaitre l'id de ce dernier afin d'implémenter la méthode getIdNode().

Comme expliqué dans le section 4.1, un noeud peut être un lieu où se déroule plusieurs activités. Donc il peut être, par exemple, à la fois un lieu de travail, un magasin, une école,.. Or si nous créons une classe « Lieu de travail », une classe « Magasin » et une classe « Ecole », toutes héritant de la classe Business, le noeud sera donc lié à trois objets différents mais liés à la classe « Business ». Nous pouvons donc rassembler toutes les instances des classes représentant les

activités qui se déroulent dans le noeud, en une variable même si ces dernières ne proviennent pas d'une même classe. En effet, comme elles ont toutes hérité de la classe Business, nous pouvons utiliser le polymorphisme[6] des objets et donc assigner les références de ces instances à des variables du type « Business ». La classe node peut donc posséder une variable, qui représentera une liste de pointeurs vers des objets des classes qui ont hérité de la classe « Business :

```
// vector contain the different services of the node.
std::vector<Businesss*> _business;
```

Actuellement, aucun lieu n'est défini dans VirtualBelgium, donc il n'y a pas encore de classe Fille de Business. Pour notre travail, nous définissons la classe « School » possédant l'héritage de la classe « Business » et implémentant la méthode getIdNode(). Ses variables et ses méthodes sont listées dans la table 5.5.

6. Soient une classe « ClasseS » et une classe « classI » telles que la « ClassI » hérite de la « ClasseS ». Grâce aux polymorphisme des objets, il est permis d'affecter à une variable de type « ClasseS », une instance de la « ClassI ».

Business
Variables :
Méthodes : virtual long getIdNode()=0 ;

↑

School		
Variables :		
int id	:	l'identifiant de l'école
long id_node	:	l'identifiant du noeud qui abrite l'école
int n_fase	:	le numéro Fase de l'école
int n_student	:	le nombre d'étudiants que l'école souhaite accueillir
int ins	:	le code INS de la commune de l'école
int n_individual	:	le nombre d'étudiants virtuels qui suivent des cours dans cette école
int business_id	:	la position de l'école dans le vecteur _business du noeud qui l'abrite
Méthodes :		
public School(long id_node)		
public School(int id, long id_node, int ins, int n_fase, int n_student)		
virtual School()		
public long getId()		
public long getIdNode()		
public int getIns()		
public int getNumberFase()		
public int getNumberStudent()		
public int getBusinessId()		
public int getIndividual()		
public void setId(int id)		
public void setIdNode(long idNode)		
public void setIns(int ins)		
public void setNumberFase(int fase)		
public void setNumberStudent(int n_student)		
public void setNIndividual(int n_individual)		
public void setBusinessId(int business_id)		

TABLE 5.5 – La classe School

La classe « School » va nous permette de représenter les écoles virtuelles. En effet, ces dernières sont des instances de la classe « School ». Cependant, nous savons qu'il existe trois types d'enseignement et il n'existe pas de variable dans la classe « School » permettant de préciser le type de l'école. Cette distinction

est réalisée en utilisant une deuxième fois les propriétés de l'héritage. Trois nouvelles classes sont construites et elles héritent de la classe « School » : « PrimarySchool », « SecondarySchool », « HigherSchool ». . Maintenant, une école peut être une instance de la classe « PrimarySchool » mais elle reste toujours une instance de la classe « School ».

Grâce aux propriétés de l'héritages, les trois classes citées ci- dessus possèdent les mêmes variables et les même méthodes que la classe « School » (voir table 5.6).

Business
Variables :
Méthodes : virtual long getIdNode()=0 ;

↑

School
Variables : (voir la table 5.5)
Méthodes : (voir la table 5.5)

↑

PrimarySchool
Variables :
Méthodes : public PrimarySchool(int id, int ins, long id_node, int n_fase, int n_student) ;

SecondarySchool
Variables :
Méthodes : public SecondarySchool(int id, int ins, long id_node, int n_fase, int n_student) ;

HigherSchool
Variables :
Méthodes : public HigherSchool(int id, int ins, long id_node, int n_fase, int n_student) ;

TABLE 5.6 – Les classes héritant de la classe « School »

Maintenant que nous avons créé les classes pour représenter les écoles virtuelles selon leur type d'enseignement, il faut construire trois méthodes dans la classe « Node » pour instancier, respectivement la classe « PrimarySchool », la classe « SecondarySchool » et la classe « HigherSchool » en fonction du type d'enseignement de l'école que l'on veut créer. Ces méthodes sont créées dans la classe « Node » afin que l'instance de l'école soit accessible par le noeud.

b) **Instanciation des Écoles**

Dans la classe Node, nous construisons la méthode becomePrimarySchool, utilisée pour instancier la classe « PrimarySchool », la méthode becomeSecondarySchool et le méthode becomeHigherSchool. Ces méthodes prennent en argument l'identifiant de l'école, le numéro de fase et le nombre souhaité d'étudiants dans l'école que le méthode va créer.

```
void Node::becomePrimarySchool(int id_school, int n_fase, int n_student)
void Node::becomeSecondarySchool(int id_school, int n_fase, int n_student)
void Node::becomeHigherSchool(int id_school, int n_fase, int n_student)
```

Les trois méthodes sont construites de la même manière donc seule la méthode becomePrimarySchool sera détaillée ci-dessous.

```
void Node::becomePrimarySchool(int id_school, int n_fase, int n_student) {


   PrimarySchool*  school =  new PrimarySchool(  id_school,
                   this->_ins,this->_id, n_fase, n_student);
   Business* business=school;


  this->_business.push_back(business);


  (*school).setBusinessId(_business.size()-1);



  }
```

Premièrement, la méthode instancie la classe « PrimarySchool » avec les mêmes arguments mais en y rajoutant l'identifiant du noeud et son code INS, qui sont des variables de la classe « Node ». En d'autres mots, ces dernières sont reprises grâces au pointeur this qui pointe vers l'objet représentant le noeud.

```
 PrimarySchool*  school =  new PrimarySchool(  id_school, this->_ins,
                           this->_id, n_fase, n_student);
```

Ensuite, elle rajoute le pointeur de l'instance de la classe « PrimarySchool » dans la variable _business créée précédemment en la faisant passer pour un pointeur de la Classe « Business ».

```
Business* business=school;
this->_business.push_back(business);
```

Pour finir, pour chaque école, on doit pouvoir déterminer son emplacement exact, c'est-à-dire le noeud qui l'abrite et sa position dans l'ensemble des activités qui se déroulent dans ce noeud. En d'autres mots, cela consiste à connaitre l'identifiant du noeud et la position de l'école dans le vecteur _business. Or pour l'instant, seule la variable id_school est initialisée lors de l'instanciation de la classe. Nous devons donc faire appel à la méthode setBusinessId de la classe « PrimarySchool » pour initialiser la variable business_id de cette dernière.

```
(*school).setBusinessId(_business.size()-1);
```

En conclusion, trois nouvelles méthodes ont été créées dans la classe « Node » ainsi que deux autres, permettant d'obtenir l'ensemble des activités liées au lieu (getBusiness()) ou uniquement une activité en particulier en fonction de sa position dans le vecteur _business (getBusinessAtaPosition (int i). La table 5.7 reprend toutes les variables et les méthodes de la classe « Node ». En particulier, celles qui sont récemment créées sont écrites en rouge.

Node		
Variables :		
long _id	:	l'identifiant du noeud
double _x	:	L'abscisse du noeud
double _y	:	L'ordonnée du noeud
int _ins	:	Le code INS de la commune
vector<long> _links_out_id	:	L'ensemble des liens partant de ce noeud
float _key	:	La clé du noeud
int _index	:	L'index du noeud
map <string,long> _indicators	:	Indicateur du noeud
vector <Business*> _business	:	Ensemble des activités se déroulant dans le noeud
Méthodes :		
public Node()		
public Node(long id, double x, double y, int ins)		
public long getId() const		
public void setId(long id)		
public int getIns() const		
public void setIns(int ins)		
public const std : :vector<long>& getLinksOutId() const		
public void setLinksOutId(const std : :vector<long>& linksOutId)		
public void addLinkOutId(long linkId)		
public float getKey() const		
public void setKey(float key)		
public double getX() const		
public void setX(double x)		
public double getY() const		
public void setY(double y)		
public int getIndex() const		
public void setIndex(int index)		
public const std : :map<std : :string, long>& getIndicators() const		
public void setIndicators(const std : :map<std : :string, long>& indicators)		
public void addIndicator(std : :string aIndicator, long aIndicatorValue)		
public void friend bool operator<(const Node &aNode1, const Node &aNode2)		
public friend bool operator>(const Node &aNode1, const Node &aNode2)		
public bool operator<(Node &aNode)		
public bool operator>(Node &aNode)		
public std : :vector<Business*> getBusiness()		
Business* getBusinessAtAPosition(int i)		
public void becomePrimarySchool(int id_school, int n_fase, int n_student)		
public void becomeSecondarySchool(int id_school, int n_fase, int n_student)		
public void becomeHigherSchool(int id_school, int n_fase, int n_student)		

TABLE 5.7 – La classe « Node »

Maintenant que nous avons conçu les nouvelles méthodes de la classe « Node », nous appelons la fonction becomePimarySchool dans la méthode read_school de la classe Data().

- La dernière étape à réaliser dans la fonction read_school est de compléter la variable _map_idPrimarySchool_id_Node qui contient les tuples constitués de l'identifiant de l'établissement scolaire primaire et l'emplacement ce ce dernier dans le réseau, c'est-à-dire l'identifiant du noeud et sa position dans le vecteur _business de ce dernier.

```
while (getline(filePS, a_line))
  { data = split<int>(a_line, ";");
    id  = data[0];
    ins = data[1];
    fase = data[2];
    studentCount=data[3];


// l'id du noeud qui abrite l'école
long id_node;

// On sélectionne un noeud aléatoirement parmi la commune dont le
//code INS est "ins"
id_node=getOneNodeIdFromIns(ins);

// On reprend le réseau de VirualBegium
Network net  = getNetwork();

//On prend l'ensemble des noeuds du réseau. Ils sont classés en
//fonction de leur id.
map<long,Node> nodes= net.getNodes();


// le noeud qui abrite l'école
Node *node;

// On récupère le noeud en fonction de son id
node=&nodes[id_node];



   //On crée l'école
   (*node).becomePrimarySchool(id,fase, studentCount );


// On stocke la position de l'école dans un vecteur
schoolPosition.clear();
schoolPosition.push_back(id_node);
schoolPosition.push_back((*node).getBusiness().size()-1);

//On remplit la variable _map_idPrimarySchool_idNode avec la nouvelle
//école créée.
this->_map_idPrimarySchool_idNode.insert(make_pair(id, schoolPosition));
}
```

5.2.2 Conclusion

En résumé, pour chaque ligne du fichier, nous créons une instance de la classe « PrimarySchool » ou « SecondarySchool » ou « HigherSchool » en fonction du type d'enseignement. Cette dernière contient le nombre d'étudiants que l'école souhaite accueillir. De plus, on l'assigne aléatoirement à un noeud. Cette démarche correspond donc bien au modèle de distribution expliqué dans la section 4.

En plus d'implémenter le modèle, la méthode initialise les variables suivantes :

```
// map de l'identifiant de l'école primaire (key) x
//l'emplacement de l'école (value)
map<int, vector<long> > _map_idPrimarySchool_idNode;

 // map de l'identifiant de l'école secondaire (key) x
 //l'emplacement de l'école (value)
map<int,vector<long> > _map_idSecondarySchool_idNode;

// map de l'identifiant de l'école supérieure(key) x
//l'emplacement de l'école (value)
map<int, vector<long> >  _map_idHigherSchool_idNode;
```

Elles seront utiles dans l'implémentation du modèle d'assignation des écoles expliquées dans la section suivante.

5.2.3 Remarques

- Nous avons choisi de créer une classe « Business » à la place de construire une seule classe « School ». Le programme fonctionnerait quand même en utilisant que la classe « School » mais il serait moins modulaire. Une personne qui veut, dans un certain futur, déterminer un noeud comme un lieu de travail, va créer une nouvelle classe représentant les différents lieux de travail. Or, dans ce cas, nous obtenons deux classes complètement indépendantes et nous ne pouvons utiliser le polymorphisme des objets pour créer une unique variable dans la classe « Node ».

- Pour un programme plus structuré, il aurait été plus adéquat de créer trois méthodes dans la classe Data, une pour chaque type d'enseignement. Cependant, il y a un inconvénient à opter pour cette structure. En effet, le réseau ne peut être modifié. Si on effectue un changement dans les noeuds de ce dernier, nous sommes obligés de redéfinir tous les noeuds du réseau avec les fonctions suivantes.

  ```
  net.setNodes(nodes);
  setNetwork(net);
  ```

  Dès lors, si on réalise trois méthodes différentes, il faudra réactualiser trois fois les noeuds du réseau.

5.3 Implémentation du modèle d'assignation d'une école

Cette section met en évidence les différentes modifications réalisées dans le programme Virtualbegium afin d'y implémenter le modèle final d'assignation d'une école expliqué dans la section 4.3. Dans cette section, nous avons vu que pour mieux distribuer les étudiants dans les écoles, il faut réaliser plusieurs synchronisations. Ces implémentations ne se sont pas faites sans difficulté, d'autant qu'il faut toujours respecter la contrainte liée au temps d'exécution.

5.3.1 Détails de l'implémentation

Nous avons expliqué dans la section précédente que les écoles de VirtualBelgium étaient représentées par des instances de la classe School. Cependant, la librairie MPI n'inclut pas de fonction qui permet de mettre en commun des objets. Nous ne pouvons donc pas prendre l'objet représentant la même école dans chaque processeur et le mettre en commun.

Différentes tentatives ont échouées dont l'utilisation des propriétés du framework Repast pour synchroniser les écoles. Les écoles sont devenues des agents et nous pouvions les synchroniser. Cependant cette implémentation prenait 2 fois plus de temps que celle actuelle pour 20 processeurs. De plus, plus il y avait de processeurs, plus le temps augmentait de manière exponentielle. Au final, nous avons dû revenir à une solution assez simple qui consiste à stocker les nombres d'étudiants inscrits dans les écoles, dans un tableau qui sera mis en commun à travers tous les processeurs.

Dans cette partie, nous séparons les étudiants dans trois groupes en fonction de leur diplôme. Nous effectuons cette distinction afin de créer trois tableaux reprenant, respectivement, les écoles primaires, secondaires et supérieures et ainsi éviter d'effectuer des synchronisations avec l'ensemble des écoles stockées dans un seul tableau. En effet, synchroniser un grand tableau prend plus de temps que la synchronisation de trois petits tableaux consécutivement.

Nous devons également mettre en évidence que tous les étudiants sont repris dans ces trois groupes et pas uniquement les individus qui possèdent l'activité « École » dans sa chaine d'activités.

- **Modifications dans le fichier Model.hpp**

```
 //Shared context containing the Primary student individual
 //agents of the simulation
repast::SharedContext<Individual> agentsStudentPS;

  // Shared context containing the Secondary student individual
  //agents of the simulation
 repast::SharedContext<Individual> agentsStudentSS;

 //!< Shared context containing the Higher student individual
 //agents of the simulation
 repast::SharedContext<Individual> agentsStudentHS;
```

- **Modifications dans le constructeur de la classe « Model » du fichier Model.cpp**

```
// Cette partie est incluse dans la boucle d'initialisation des individus
// et ménages,
//c'est -à-dire dans le constructeur de la classe Model.
// "a_dip" est  le diplôme obtenu par l'étudiant ("O"= aucun, "P"= primaire,
//"S"= secondiare, "u"= supérieur)
// "a_spstatus" est la catégorie professionnelle où "E"= Etudiant



//Pour les écoles primaires:
if((a_dip=='O') && (a_spstatus=='E'))
  {  agentsStudentPS.addAgent(new Individual(ind_temp));
  }


//Pour les écoles secondaires:
if((a_dip=='P') && (a_spstatus=='E'))
  { agentsStudentSS.addAgent(new Individual(ind_temp));
  }

//Pour les écoles supérieures:

if(((a_dip=='S') && (a_spstatus=='E')) ||
((a_dip=='U') && (a_spstatus=='E')))
 {
  agentsStudentHS.addAgent(new Individual(ind_temp)); }
```

Dans la suite du travail, seules les modifications effectuées pour les étudiants de l'enseignement primaire sont expliquées. Les changements pour les deux autres types d'enseignement suivent la même logique.

1. Premièrement, nous devons initialiser les tableaux qui vont contenir les informations à synchroniser, c'est-à-dire le nombre de places restantes dans les écoles primaires. Chaque ligne représente une école. Cette initialisation est effectuée dans le constructeur de la classe « Model ».

En réalité, deux tableaux vont être nécessaires à la synchronisation.

- Le premier tableau n'est pas mis en commun à travers tous les processeurs. Il sert juste à contenir le nombre de places qui ont été attribuées entre la précédente synchronisation et la prochaine. Ce tableau sera donc différent pour chaque processeur. Il est nommé NbTemporaryLastPlacePS et est initialisé à un tableau nul pour tous les processeurs, excepté, le processeur 0, dont le tableau contiendra le nombre d'étudiants inscrits dans les écoles. Le processeur 0 sera celui de référence dans la synchronisation.
- Le second tableau sera mis en commun à travers les processeurs. Il contiendra les nombres totaux de places restantes dans les écoles lors de la synchronisation. Il est nommé NbLastPlacePS.

Ces tableaux doivent être accessibles dans toutes les méthodes de la classe « Model ». Ils vont donc devenir des variables de la classe « Model ».

- **Changement dans le fichier Model.hpp**

```
int *NbTemporaryLastPlacePS;
int *NbLastPlacePS
```

- **Changement dans le fichier Model.cpp**

```
Network net  = Data::getInstance()-> getNetwork();
map<long,Node> nodes= net.getNodes();
Node *node;

totalPS   = Data::getInstance()->getNumberPrimarySchool() ;

NbTemporaryLastPlacePS=(new int[totalPS]);
NbLastPlacePS=(new int[totalPS]);

Business* business;
School* school;
for (int i = 0 ; i < totalPS; i++)
{  vector<long> schoolPosition;
   schoolPosition =Data::getInstance()->getIdNodePrimarySchool(i+1);
   node=&nodes[schoolPosition[0]];
   business=(*node).getBusinessAtAPosition(schoolPosition[1]);

   school=&static_cast<School&>(*business) ;

   if (_proc==0)
    {NbTemporaryLastPlacePS[i]=(*school).getNumberStudent() ;
     }
   else
     {
      NbTemporaryLastPlacePS[i]=0;
     }
      NbLastPlacePS[i]=(*school).getNumberStudent() ;

}
```

(a) On récupère le nombre d'écoles virtuelles primaires grâce à une méthode de la classe "Data"

```
totalPS   = Data::getInstance()->getNumberPrimarySchool() ;
```

(b) On initialise les deux tableaux :

```
// Initialisation des variables NbTemporaryLastPlacePS
//et NbLastPlacePS
NbTemporaryLastPlacePS =(new int[totalPS]);
NbLastPlacePS=(new int[totalPS]);
```

(c) On va boucler sur les écoles. Dans la boucle, on va récupérer un pointeur vers l'objet représentant l'école n°i, grâce à la position de cette dernière, donnée par la méthode "getIdNodePrimarySchool" de la classe Data. Grâce à ce pointeur, on peut connaitre le nombre d'étudiants que l'école peut accueillir. En d'autres mots, cela représente le nombre de places restantes dans l'école.

```
Business* business;
School* school;

for (int i = 0 ; i < totalPS; i++)
       {vector<long> schoolPosition;
     schoolPosition =Data::getInstance()->getIdNodePrimarySchool(i+1);
        node=&nodes[schoolPosition[0]];
        business=(*node).getBusinessAtAPosition(schoolPosition[1]);

  //On récupère un pointeur vers l'objet qui représente l'école n°i
        school=&static_cast<School&>(*business) ;

      .
      .
      .


  }
```

(d) Maintenant qu'on a le nombre souhaité d'étudiants dans chaque école, on va initialiser les deux tableaux à travers la boucle.

```
Business* business;
School* school;


for (int i = 0 ; i < totalPS; i++)
     {  vector<long> schoolPosition;
        schoolPosition =Data::getInstance()->getIdNodePrimarySchool(i+1);
        node=&nodes[schoolPosition[0]];
        business=(*node).getBusinessAtAPosition(schoolPosition[1]);

  //On récupère un pointeur vers l'objet qui représente l'école n°i
        school=&static_cast<School&>(*business) ;




   //La variable "_proc" donne le numéro du processeur
       if (_proc==0)
     {NbTemporaryLastPlacePS[i]=(*school).getNumberStudent() ;
     }
       else
     {
       NbTemporaryLastPlacePS[i]=0;
     }

      NbLastPlacePS[i]=(*school).getNumberStudent() ;




  }
```

2. Maintenant que les deux tableaux sont remplis et que nous avons notre ensemble d'étudiants primaires grâce à l'appel du constructeur de la classe « Model », nous créons deux nouvelles méthodes dans cette dernière. La méthode « allocationSchool » et « loopSchool » sont communes à tous les types d'enseignement.

Changement dans le fichier Model.hpp

```
void allocationSchool(Individual *agent, float epsilon);
void loopSchool();
```

Pour réaliser le modèle d'assignation, on fait appel à la méthode loopSchool() dans la méthode initSchedule(). Elle est appelée dans cette dernière afin que le modèle soit appliqué avant de commencer à faire évoluer les personnes en utilisant la méthode step(). La méthode loopSchool() est en réalité une série de trois boucles sur les étudiants, respectivement de l'enseignement primaire, l'enseignement secondaire, l'enseignement supérieur. À chaque étape fait appel à la méthode allocationSchool pour assigner une école à l'étudiant courant de la boucle.

Nous allons expliquer les deux méthodes par l'ordre d'appel.

(a) Initialisation des différentes variables utiles à la synchronisation des tableaux.

```
void Model::loopPrimarySchool()
{
// On récupère le premier et le dernier iterator de la liste
// des étudiants.
repast::SharedContext<Individual>::const_local_iterator it_beg =
                    agentsStudentPS.localBegin();


repast::SharedContext<Individual>::const_local_iterator it_end =
                        agentsStudentPS.localEnd();



 //On calcule le nombre d'étudiants de l'enseignement
 // primaire dans le processeur
  int sizeAgent=agentsStudentPS.size();

//Le nombre de synchronisations est fixé
  int numberSynchonization=1200;

  //On réalise la somme entière comme expliqué dans le
  // point 7.1 de la section 4.3.4.2
  int numberAgentPerSynchro=sizeAgent/numberSynchonization;

  //On calcule le reste de cette division.
  double difference=sizeAgent%numberSynchonization;


  // Variables  permettant de compter le nombre d'étudiants qui ont
  //déjà été assignés à une école
  // et le nombre de synchronisation déjà effectuées.
  int numberOfIndInLoop(0);
  int currentSynchro(0);
```

(b) On boucle sur les itérators jusqu'à ce qu'il n'y en ait plus. À chaque itération, un appel à la méthode allocationSchool est réalisé en ayant comme argument, le pointeur de l'objet qui représente l'étudiant courant et la valeur de l'epsilon comme arguments.

```
allocationSchool(&**it_beg,epsilon);
```

Cette méthode est divisible en trois parties en fonction du diplôme de l'étudiant. L'explication suivante ne porte que sur la première partie qui concerne les étudiants sans diplôme :

i. Déclaration des variables utilisées et séparation en fonction du type d'enseignement. Ceci est la première étape du modèle d'assignation expliquée au point 8.1 de la section 4.3.4.2. Ensuite, on génère une distance avec le générateur de loi log-normale.

```
// Le réseau
Network net = Data::getInstance()->getNetwork();
map<long, Node> nodes = net.getNodes();
Node node;

double count(0);
//////////////////
// primary school// ///////////////////

if((*agent).getEducation()=='0')
    { //vecteur contenant les identifiants des écoles
      //sélectionnées
       vector<int>  setSelectPrimarySchool;

       //vecteur contenant les places restantes des écoles
       //sélectionnées
   vector<int> vector_totalLeftoverPlaces;

//paramètres pour le générateur mixture log-normale
     float _distance;
     vector<float> mu,sigma,p;
             float _distance;
     mu.push_back(8.0259);
     sigma.push_back(sqrt(1.8792));
     p.push_back(1);



    // On génère une distance avec le générateur de mixture de loi
    // log-normale avec une borne inférieure
   //égale à 20 et la borne supérieure égale à 140 000
   _distance = RandomGenerators::getInstance()->
mixt_lognorm_dev.dev(mu, sigma,p,20,140000);
          .
        .
    .
  }
```

```
if((*agent).getEducation()=='P')
{ ....}

if(((*agent).getEducation()=='U') || ((*agent).getEducation()=='S'))
    {...}
```

ii. Avec cette distance, nous avons déjà implémenté la deuxième étape du modèle. La troisième étape consiste à trouver tous les noeuds qui sont à la distance "_distance" du noeud représentant le domicile de l'étudiant, à 250 m près. Ensuite on va seulement garder les noeuds qui représentent une école, c'est-à-dire si la variable _business contient au moins un pointeur vers une instance de la classe PrimarySchool. Si aucun des noeuds selectionnés ne possèdent une école primaire, alors une nouvelle distance est générée

```
if((*agent).getEducation()=='0')
    {
    vector<int>  setSelectPrimarySchool;
    vector<int> vector_totalLeftoverPlaces;

    float _distance;
    vector<float> mu,sigma,p;
    float _distance;
    mu.push_back(8.0259);
    sigma.push_back(sqrt(1.8792));
    p.push_back(1);


   _distance = RandomGenerators::getInstance()->mixt_lognorm_dev.dev(mu,
                                  sigma,p,20,140000);
   setSelectPrimarySchool.clear();
   vector_totalLeftoverPlaces.clear();

   //On selectionne l'ensemble de noeuds qui se trouve à la distance
   // _distance du domicile à epsilon près
   setOfIdNode = net.getSetOfDestFromSource((*agent).getHouse(),
                                                 _distance,epsilon);

   //On regarde quels noeuds possèdent une école primaire
   for(unsigned int i=0; i< setOfIdNode.size() ; i++)
   {vector<Business*> vector_business =
                              (nodes[setOfIdNode[i]]).getBusiness();

       for(unsigned int j=0; j< vector_business.size() ;j++)
         {
           if(typeid(*vector_business[j]).name()
                        ==typeid(PrimarySchool).name())
          {
             PrimarySchool *school;
             school=&static_cast<PrimarySchool&>(*vector_business[j]) ;
             int totalLeftoverPlaces;
            if(_proc!=0)
             {
                 totalLeftoverPlaces= NbLastPlacePS[[id_school-1]
                              + NbTemporaryLastPlacePS[[id_school-1];
              }
             else {
                  totalLeftoverPlaces= NbLastPlacePS[[id_school-1];
                  }

             if(totalLeftoverPlaces <0)
               {totalLeftoverPlaces=0;

               }

            // on stocke l'école trouvée
            count=count+totalLeftoverPlaces;
            setSelectPrimarySchool.push_back(id_school);
            vector_totalLeftoverPlaces.push_back(totalLeftoverPlaces);


          }

         }
       }



 }
 //on itère jusqu'à trouver au moins un noeud avec une école primaire
  while ((setSelectPrimarySchool.size()==0) || (count==0));
```

Si l'ensemble est non vide, alors on garde en mémoire dans le vecteur setSelectPrimarySchool , l'identifiant des écoles trouvées. De plus le vecteur vector_totalLeftoverPlaces contient le nombre de places restantes dans ces écoles.

iii. On utilise la méthode prédéfinie « draw_discrete(vector<float> number) ». Cette méthode réalise la méthode de la fonction de répartition utilisée dans l'étape 5 du modèle. Elle ressort un nombre entier qui indiquera la position (en commençant par 1) de l'école choisie dans le vecteur setSelectPrimarySchool.

```
int choix;
choix=draw_discrete(proportion);
```

iv. Ensuite, il faut mettre à jour le tableau NbTemporaryLastPlacePS pour les processeurs autres que 0. Pour le processeur n°0, les deux tableaux sont mis-à-jour car le tableau NbTemporaryLastPlacePS du processeur 0 est le tableau de référence lors de la synchronisation.

```
    if(_proc==0)
{
  NbTemporaryLastPlacePS[ setSelectPrimarySchool[choix]-1]
                      =NbLastPlacePS[setSelectPrimarySchool[choix]-1]-1;
  NbLastPlacePS[ setSelectPrimarySchool[choix]-1]
                       =NbLastPlacePS[setSelectPrimarySchool[choix]-1]-1;


}
      else
{
 NbTemporaryLastPlacePS[setSelectPrimarySchool[choix]-1]--;
}
```

Maintenant qu'on a assigné une école à l'étudiant courant de la boucle, on revient dans la fonction loopSchool(). Après il y a deux possibilités :

- soit on passe à l'étudiant suivante
- soit on synchronise les tableaux. Comme nous avons expliqué dans le chapitre précédent, la synchronisation ne peut se faire à chaque assignation. La synchronisation sera réalisée si la position de l'étudiant dans l'ensemble agentsStudentPS qui est donnée par la variable « multiple » est un multiple du nombre d'étudiants par synchronisation. Le code ci-dessous permet donc une synchronisation des places restantes dans les écoles.

```
while (it_beg != it_end) {

 //on va assigne une école à l'étudiant
 allocationSchool(&**it_beg,epsilon);


     numberOfIndInLoop++;

// on effectue la synchronisation
  if ((numberAgentPerSynchro !=0 ) && (numberOfIndInLoop> difference))
    { int multiple=numberOfIndInLoop-difference;
      if(((multiple)%numberAgentPerSynchro)==0)
     { boost::mpi::all_reduce(*(RepastProcess::instance()->getCommunicator()),
       NbTemporaryLastPlacePS,totalPS,NbLastPlacePS,std::plus<int>());

         // Mise à jour des tableaux
          if(_proc!=0)
           {
             for(int i=0; i<totalPS; i++)
              {
              NbTemporaryLastPlacePS[i]=0;
               }
           }
          else{ for(int i=0; i<totalPS; i++)
           {
             NbTemporaryLastPlacePS[i]=NbLastPlacePS[i];
           }
    }


   currentSynchro ++;


   }




  it_beg++;

}
```

La fonction all_reduce de la norme MPI additionne les tableaux NbTemporary-LastPlacePS de la taille totalPS de tous les processeurs et stockent le résultats dans les tableaux NbLastPlacePS.

5.3.2 Conclusion

Nous pouvons conclure que toutes ces modifications permettent d'appliquer le modèle d'assignation, défini au chapitre précédent.

Conclusion

Grâce à ce travail, la plateforme VirtualBelgium a évolué. En effet, les étudiants de la population synthétique ne vont plus suivre leurs cours dans n'importe quel noeud du réseau virtuel belge mais uniquement dans un ensemble de noeuds précis. Effectivement, il existe maintenant un réseau d'écoles de l'enseignement primaire, secondaire et supérieur. Ce dernier a été conçu en tenant compte de la réalité, c'est-à-dire du nombre d'écoles qui composent la Belgique et le nombre d'étudiants accueillis dans ces dernières pour l'année 2011-2012 (2010-2011 pour l'enseignement supérieur). Cependant, nous avons pu observer une différence considérable entre le nombre d'étudiants de la population synthétique, créée à partir de l'enquête Mobel effectuée en 2001 et le nombre de places dans les écoles belges. Après plusieurs recherches, nous avons constaté que ce problème a pour origine les corrélations réalisées lors de la création de la population synthétique. La modification de cette population ne peut être réalisée dans ce travail par manque de temps. Une solution intermédiaire a été trouvée et consiste à modifier les valeurs des données récoltées sur les écoles belges tout en gardant la même proportionnalité. Les écoles virtuelles ont donc été créées à partir de ces données ajustées. Suite à l'application du modèle de distribution, ces écoles ont été placées aléatoirement dans le réseau de VirtualBelgium.

Après que le réseau d'écoles ait été créé, nous avons essayé de modéliser la distribution des étudiants dans ces dernières. Ce modèle devant respecter principalement la contrainte suivante : le nombre d'étudiants inscrits dans chaque école doit correspondre au nombre souhaité d'étudiants. Différents modèles ont été considérés, appartenant à la classe de modèle « Trips Distribution », mais malheureusement beaucoup d'entre eux ne peuvent s'appliquer à VirtualBelgium en raison du nombre important de données à traiter et donc, d'une trop grande demande de calculs de distance. Seuls les modèles utilisant une distance prédéfinie entre le domicile et l'école sont acceptables et peuvent être implémentés dans le programme. Différents tests et comparaisons ont été effectués et le modèle tenant compte du nombre de places restantes a été retenu. En d'autres mots, ce modèle utilise le principe de proportionnalité avec les nombres de places restantes dans les écoles. Plus il y a de places libres, plus on a de la chance d'aller dans cette école.

Cependant, ce modèle ne respecte pas entièrement la contrainte. En effet, l'implémen-

tation de ce modèle demande des synchronisations entre les processeurs lors de l'exécution du programme en programmation parallèle. Cette méthode peut donner à une école un nombre d'étudiants inscrits différent du nombre attendu d'étudiants. Comme la différence est assez petite pour certaines d'entre elles et en prenant compte une marge d'erreur de cinq individus, nous pouvons dire que nous avons obtenu un bon modèle. De plus, une deuxième contrainte liée à la programmation, a joué un rôle dans la distribution des étudiants : le changement de la valeur maximum autorisée entre la distance générée par un générateur de nombres pseudo-aléatoires et la distance que l'étudiant virtuel parcourt pour atteindre son école dans les dernières synchronisations afin de minimiser le temps d'exécution. Cette modification entraine également une marge d'erreurs. Et enfin, il y a un troisième défaut à ce modèle : les distances entre le domicile et l'école des étudiants virtuels ne suivent la loi de probabilité demandée lorsque l'ensemble de ces dernières dépasse la longueur de 1000 distances. Nous avons tenté différentes solutions afin de pouvoir rapprocher au maximum cet ensemble de distance de la loi de probabilité voulue. Au final, nous ne pouvons pas déclarer que ce modèle respecte la contrainte mais il s'en approche.

Nous ne pouvons comparer cette distribution des étudiants à travers les écoles, à la réalité par manque de données. De plus, les données initiales récoltées ont été modifiées pour correspondre aux individus de la population synthétique. Toutefois, l'analyse effectuée dans ce travail permet d'améliorer la plateforme VirtualBelgium et d'y apporter des éléments supplémentaires. De plus, une fois que la population synthétique aura été modifiée, il suffira de garder les données initiales récoltées.

Avec cette répartition des étudiants dans les écoles, nous ne pouvons pas trouver de conclusions pour la mobilité ou le choix d'école des étudiants belges. En effet, cette plateforme est plutôt un outil de description et des données n'ont pas été prises en compte malgré leur importance dans le choix de l'école (données socio-économiques, l'emplacement de l'école, emplacement du travail des parents,...) en raison d'un manque d'information et un augmentation considérable du temps d'exécution. De plus, actuellement les écoles sont placées aléatoirement dans le réseau VirtualBelgium. Or cela sera intéressant de les géocoder et ainsi permettre de mieux comprendre la popularité d'une école. Par exemple, une école peut-elle être plus populaire si elle est proche d'une gare.

En conclusion, nous avons trouvé un modèle d'assignation qui s'approche de la réalité avec une certaine marge d'erreur. Cependant, dans un travail futur, il serait intéressant de géocoder les écoles et d'y ajouter différentes variables pour mieux étudier le choix de l'école et perfectionner le modèle en fonction des distances pour qu'elles suivent la loi log-normale demandée.

Bibliographie

[1] Activity-based modeling of travel demand, Bhat C., Koppelman F., http://orfe.princeton.edu/~alaink/NJ_aTaxiOrf467F12/Papers/lit%20review/TSHANDBK.pdf, consulté le 25/10/2013

[2] Agents et systèmes mutliagents, http://www.damas.ift.ulaval.ca/~coursMAS/ComplementsH10/Agents-SMA.pdf, consulté le 5/11/2013

[3] AgODI-Agentschap voor Onderwijsdiensten : Baisonderwijs en CLB, Secundair onderwijs en DKO, http://www.ond.vlaanderen.be/wegwijs/agodi/default.htm, consulté le 10/05/2013

[4] Akwawua S., Pooler J. , *The development of an intervening opportunities model with spatial dominance effects*, Departement of Geography, University of Saskatchewan, Saskatoon, Saskatchewan, Canada S7N 5A5, Journal of Geographical Systems, Springer-Verlag 2001

[5] Barthélemy J., Toint PH., *A simple and flexible activity-based simulation for Belgium : a simulation platform for the Belgian population and an application to mobility*,Unamur :NaXYS, Namur ; Unamur.NAXYS,2013, Presses universitaires de Namur

[6] Barthélemy J., Toint Ph (promoteur), Cornelis E. (promoteur), *A parallelized microsimulation platform for population and mobility behaviour : application to Belgium*, Unamur : Faculté des sciences, Namur, 2014, Presses universitaires de Namur

[7] Barthélemy J., Toint Ph., *Synthetic population generation in presence of data inconsistencies*, Report NAXYS-12_2010, Namur, 23 décembre 2010, Presses universitaires de Namur

[8] Bibm@th.net, La bibliographique des mathématiques, http://www.bibmath.net, consulté le 9/05/2013

[9] Boigelot B., Programmation orientée-objet : notes de cours, Université de Liège,2013.

[10] Cirillo C., Cornelis E. Toint Ph. , *A model of weekly labor participation for a Belgian synthetic population*, Report NAXYS-19_2011, 15 mars 2011, Presses universitaires de Namur

[11] Consortium des Équipements de Calcul Intensif, http://www.ceci-hpc.be/, consulée le 17/04/2014

[12] Conseil des recteurs des universités francophones de Belgique, http://www.cref.be, consulté le 12/04/2013

[13] Deutschsprachige gemeinschaft belgien, http://www.bildungsserver.be/desktopdefault.aspx, consulté le 9/11/2013

[14] Dico Info, http://dictionnaire.phpmyvisites.net/, consulté le 5/11/2013

[15] Économie,Statistic Belgium, http://statbel.fgov.be/fr/statistiques/chiffres/,Belgian Federal Governement,2013, consulté le 23 avril 2014.

[16] Entreprise publique des technologies nouvelles de l'information et de la communication, http://www.etnic.be, consulté le 11/04/2013

[17] Fédération Wallonie-Bruxelles, httpf://www.inscription.cfwb.be/index.php?id=383., consulté le 11/04/2013

[18] Futura-sciences : High-tech http://www.futura-sciences.com/magazines/high-tech/, 1995,consulté le 17/04/2014

[19] Hardy A., Probabilités : slides du cours, chapitre 1 : slide 15, Université de Namur, 2012.

[20] Hoger onderwijs : Voor hogerscholen en universiteiten, studenten en personeel,..., http://www.ond.vlaanderen.be/hogeronderwijs/default.htm, consulté le 10/05/2013

[21] Hubert J-P, Toint Ph, *La mobilité quotidienne des Belges*, Presses universitaires de Namur,2002

[22] HYPERGEO, http://www.hypergeo.eu, consulté le 5/05/2013

[23] Lima & Associates, *Lincoln MPO Travel Demand Model,Draft Model Documentation* , January 2006, http://www.princeton.edu/~alaink/Orf467F12/LincolnTravelDemandModel.pdf, consultée le 10 avril 2014

[24] Main Page : Welcome to OpenStreetMap, http://wiki.openstreetmap.org/wiki/Main_Page, consulté le 10/11/2013

[25] Mathématiques : outils pour la biologie, http://spiral.univ-lyon1.fr/mathsv/, consulté le 24/10/2013

[26] Millington J., Butler T., Hammer C, *Aspiration, Attainment and Success : An Agent-Based model of Distance-Based School Allocation*, Journal of Articificial Societies and Social Simulation 17 (1) 10, 31 janvier 2014, JASS

[27] Neutens T. ,*Space, time and accessibility : Analyzing human activities and travel possibilities from a time-geographic perspective*, 2010, Universiteit Gent, University press.

[28] NOTRE BELGIQUE.BE, La Belgique, ses communes et ses dépendances, mise à jour le 7/11/12, 2008-2012, http://www.notrebelgique.be/fr/index.php?nv=1&PHPSESSID=b8b0b83f9415a649fb4cfe4858a35fad, consulté le 25 mai 2014

[29] Onderwijsaanbod : Waar kan it wat studeren, http://www.ond.vlaanderen.be/onderwijsaanbod/, consulté le 10/05/2013

[30] Onderwijsstatisteiken : Statistische data, indicatoren, PISA,... , http://www.ond.vlaanderen.be/onderwijsstatistieken/default.htm, consulté le 11/04/2013

[31] Onderwijs.vlaanderen.be : Site van het Vlaams Ministerie van Onderwijs en Vorming, www.ond.vlaanderen.be, consulté le 11/04/2013

[32] OpenStreetMap, la carte coopérative libre, http://www.openstreetmap.org/,2013, consulté le 13/04/2013

[33] Ortùzar J. de D, Willlumsen L. G., *Modelling transport*, Second Edition, John Wiley & Sons, 1995

[34] Simmonds D., Bates J.J., *Activity-based modelling : Research directions and possibilities*, Institut für Verkehrsplanung, Transporttecknik, Strassenund Eisenbajnbau,Draft-Paper , Octobre 2000

[35] Toint Ph, *École et mobilité quotidienne*, Groupe de Recherche sur les Transports, FUNDP, Namur, http://perso.fundp.ac.be/~phtoint/talks/2003_mob_scolaire.pdf, consulté le 23 mai 2014

[36] Train K.,*Discrete Choice Methods with Simualtion*, University of California, Berkeley and NERA, Second Edition, 2009 Cambridge University Press

[37] Transportation Engineerin, online Lab Manuel, Oregon State University, Portland State University, University of Idaho, 2000-2003, http://www.webpages.uidaho.edu/niatt_labmanual/index.htm, consulté le 10 avril 2014

[38] Trip Distribution, Format Word, http://www.ctre.iastate.edu/educweb/ce451/LECTURES/Trip\%20Distribution/distribution.doc, consulté le 10 avril 2014

[39] Union Francophones des Associations de Parents de l'enseignement Catholique ascbl, http://www.ufapec.be/, consulté le 11/04/2013

[40] Veenstra S.A., Thomas T., Tutert S.I.A. , *Trip distribution for limited destinations : A case study for grocery shopping trips in the Netherlands*, 15 Novembre 2009 , http://www.utwente.nl/ctw/vvr/pdf/2010_Veenstra_Thomas_Tutert_TRB_Trip_distribution.pdf

[41] Wikipédia : l'encyclopédie libre, http://fr.wikipedia.org/wiki/Wikip%C3%A9dia:Accueil_principal, consulté le 5/11/2013

Table des figures

Liste des tableaux

Appendices

Annexe A

Installation de VirtualBelgium

A.1 Installation

La section suivante, écrite en collaboration avec J. Barthelemy, décrit le processus d'installation du programme VirtualBelgium sur un ordinateur personnel. Plus précisément, ce document explique comment utiliser et installer VirtualBelgium sur une architecture GNU/linux 64 bits. Pour commencer, nous détaillerons les fichiers et librairies à télécharger. Ensuite, nous décrirons les commandes d'installation à entrer et enfin celles pour lancer le programme en lui-même.

A.2 Téléchargements

Avant toute chose, il est évident que nous devons commencer par télécharger une distribution Linux. Toutes les commandes décrites dans les lignes suivantes ont été testées sur la distribution Linux Mint 15.

Les fichiers du code en lui-même sont hébergés à l'adresse suivante :

`http://sourceforge.net/projects/virtualbelgium/files/`

et sont organisés de la manière suivante :

./	dossier racine, contient les scripts d'exécution ainsi que le "Makefile"
./bin	fichiers d'exécution et de configuration
./data	données d'entrées
./doc	documentation
./include	fichiers d'en-tête
./licenses	licenses de Repast HPC, tinyxml2 et VirtualBelgium
./logs	fichiers log des simulations
./outputs	sorties générées par les simulations
./scripts	scripts pour le traitement des sorties
./src	fichiers sources et le "Makefile"

Listons à présent les différents outils obligatoires à la compilation et/ou exécution du programme.

Compilateur C++

Le projet étant codé en C++, un compilateur de ce langage est nécessaire. Le compilateur le plus connu est g++. Il est disponible dans le gestionnaire de dépôts grâce à la commande

apt-get install g++

Outil Make

Cet outil permet de simplifier les commandes de compilation lorsque le programme comporte beaucoup de fichiers. Il s'occupe des dépendances et automatise certains processus.

apt-get install g++

La commande Make exécute les règles définies par le fichier Makefile.

L'environnement MPI

Le Message Passing Interface (MPI) est un standard permettant à différents processus d'une même exécution du programme de communiquer ensemble. Un utilitaire bien connu est l'exécutable OpenMPI, disponible à nouveau dans le gestionnaire de dépôt par la commande

apt-get install libopenmpi-dev openmpi-common

La librairie Repast HPC

La librairie principale utilisée par notre programme est le framework Repast HPC 1.0.1. Elle permet de modéliser au mieux un système basé sur le c++ et utilisant un grand ensemble d'unités de calcul. À l'origine, elle a été créée par le Laboratoire National d'Argonne. Le code compilable est disponible à l'adresse suivante : `http://repast.sourceforge.net`. Cette librairie en nécessite d'autres avant d'être compilée :

- Curl ;
- Boost 1.48 (ou supérieure) : En plus de cette librairie générale, nous avons besoin des spécifiques suivantes :boost-mpi, boost-system, boost-serialization and boost-filesystem ;
- NetCDF 4.2.1 ;
- NetCDF C++ 4.2.

Les commandes de téléchargement sont simplement :

Pour Curl :

apt-get install libcurl-dev

Pour Boost :

apt-get install libboost-dev libboost-mpi-dev libboost-serialization-dev
libboost-system-dev libboost-filesystem-dev

Pour NetCDF :

apt-get install libnetcdf-dev

A.3 Repast HPC

Compilation de Repast HPC

Lorsque tout est téléchargé et extrait de l'archive, la compilation s'effectue grâce aux trois commandes suivantes : (attention à utiliser les privilèges super-utilisateur).

1. ./configure
2. ./make
3. ./make install

Il se peut que la commande "make" génère une erreur. Pour y remédier, il faut remplacer toutes les occurrences de

```
getItems(...)
```

par

```
this->getItems(...)
```

dans les documents

./src/repast_hpc/DirectedVertex.h et ./src/repast_hpc/UndirectedVertex.h

Si une erreur se présente à nouveau lors de l'exécution du "make", le Makefile doit être édité de telle façon à supprimer toutes les références vers les modèles Zombie et Rumor.

A.4 Compilation et exécution de VirtualBelgium

Ordinateur personnel

Pour compiler simplement le programme sur un ordinateur personnel, il suffit d'exécuter la commande

make

Après la compilation, il faut lancer l'exécutable. Un script a été créé spécialement, il s'agit de la commande :

./run.sh NP

où NP est le nombre de processeurs désirés.

Exécution de calculs intensifs

Pour des simulations mettant en scène un nombre important d'agents, il est recommandé d'effectuer les opérations sur un système de plusieurs ordinateurs, "cluster". Suivant les appareils utilisés, il est possible que le fichier "Makefile" doive être modifié. Si le cluster utilisé est Lemaitre du Consortium des Equipements de Calcul Intensif(`http://www.ceci-hpc.be/`), la compilation se fait simplement avec la commande

make ucl

Ensuite, l'exécution se lance avec le script run_lemaitre2.sh

sbatch run_lemaitre2.sh

Ce script permet aussi de personnaliser l'exécution en proposant à l'utilisateur les options suivantes :

- mail-user : une adresse e-mail pour les notifications ;
- time : le temps d'exécution demandé ;
- ntask : le nombre de processeurs voulu ;
- mem-per-cpu :la mémoire réquisitionnée par processeur.

Debuggage

Si le code est modifié pour un ajout de fonctionnalité ou pour une optimisation, il est préférable de pouvoir le débugger si une erreur survient lors de la compilation ou l'exécution. À nouveau la commande "make" va nous être utile en lui ajoutant l'option "debug"

make debug

Cette commande fait appel au débugger GNU gdb dont la documentation peut être trouvée à `http://www.sourceware.org/gdb/documentation/`.

A.5 Configuration de VirtualBelgium

VirtualBelgium est un programme de simulation par agent contenant plusieurs modèles possibles. Le choix du modèle se fait dans le fichier

\bin\model.props

Ce fichier reprend aussi le choix, entre autres, du réseau, de la population synthétique ou encore de la partie à simuler[1].

A.6 Mise à jour du code

Les dernières versions du code de VirtualBelgium sont disponibles sur un répertoire Subversion (SVN) hébergé à l'Université de Namur. Lorsque le logiciel Subversion est installé sur notre machine, les données sont récupérables en trois étapes :

1. Demander un compte à virtualbelgium@math.unamur.be ;
2. Créer un tunnel SSH vers Gauss :

   ssh -N -f -l 5555 :localhost :3690 user@gauss.math.fundp.ac.be

3. Se connecter à

   svn co svn ://user@localhost :5555/virtualbelgium/

   pour recevoir la dernière version du programme

Les différentes commandes de Subversion pour éditer un projet sont listées à `http://svnbook.red-bean.com/index.en.html` (disponible en français).

1. Chaines d'activités, naissances, morts ou âges

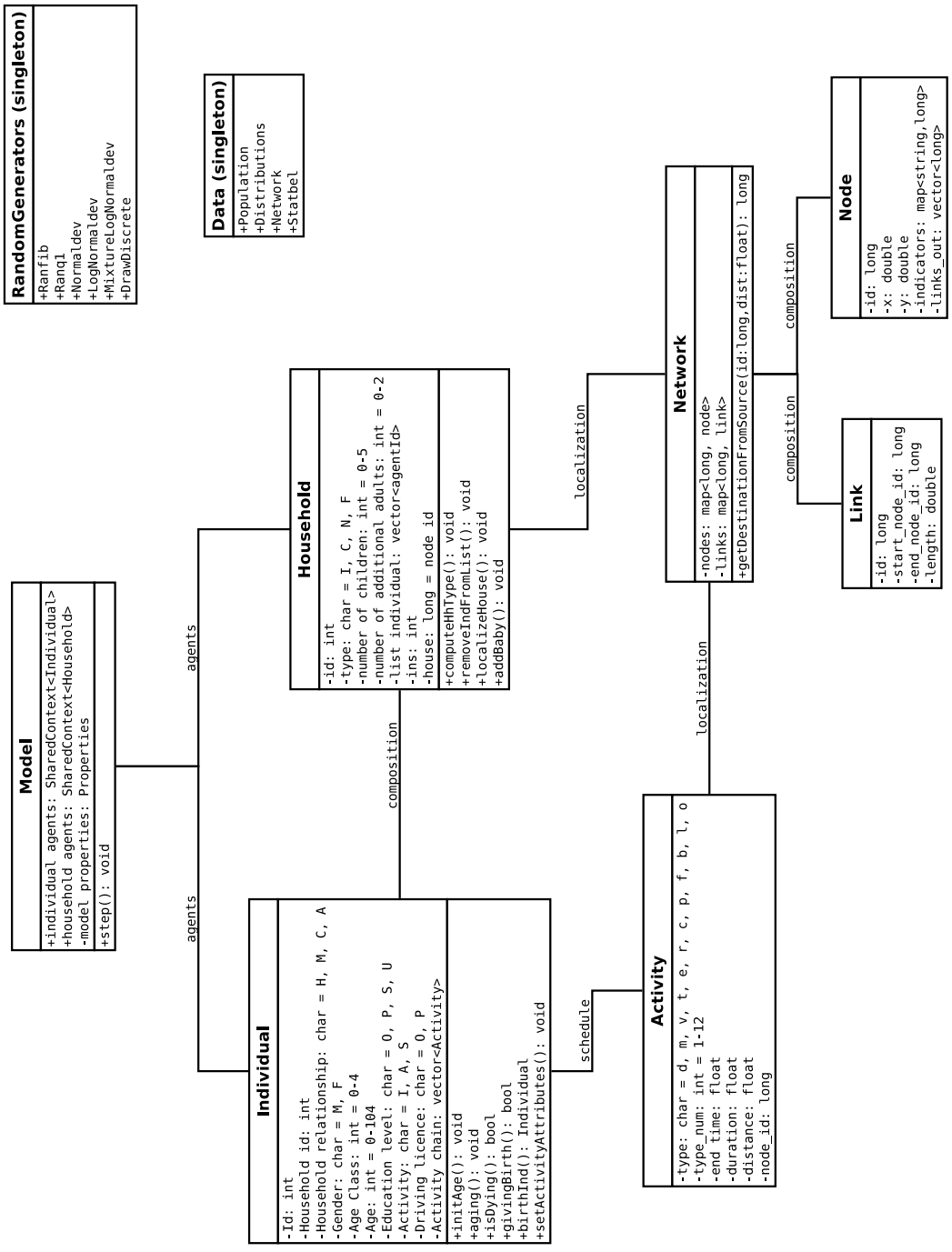


FIGURE A.1 – Diagramme des classes pour VirtualBelgium

Annexe B

Annexes concernant les activités

B.1 Variables de la classe Activity

Une activité est créée en instanciant la classe Activity du programme VirtualBelgium.

Caractéristiques	**Variables**	**Valeurs**
Le but de l'activité	_type	Les valeurs de ces deux variables sont reprises dans la table B.2.
Le numéro	_type_num	
L'heure de fin	_end_time	valeur réelle calculée en seconde
La durée	_duration ;	valeur réelle calculée en seconde
La distance parcourue	_distance	valeur réelle calculée en mètres
La durée du trajet	_dur_trip	valeur réelle calculée en seconde
L'id du nœud où se trouve l'activité	_nodeId	valeur entière

TABLE B.1 – Variables d'une activité

Le motif d'une activité est déterminé par un numéro ou une lettre suivant le tableau suivant :

_type	_type_num	Signification
d	1	déposer / reprendre quelqu'un
m	2	Activités à la maison
v	3	Se déplacer pour le travail
t	4	Travail
e	5	École
r	6	Manger à l'extérieur
c	7	Faire les courses
p	8	Activités personnelles (banque, docteur)
f	9	Rendre visite à la famille / aux amis
b	10	Promenade
l	11	Loisirs
o	12	autre
X	13	aucune

TABLE B.2 – Les Activités dans VirtualBelgium avec les valeurs aux variabes _type et _type_num

B.2 Fonctions de répartition pour la distance parcourue et celles pour la durée

B.2.1 Distance parcourue

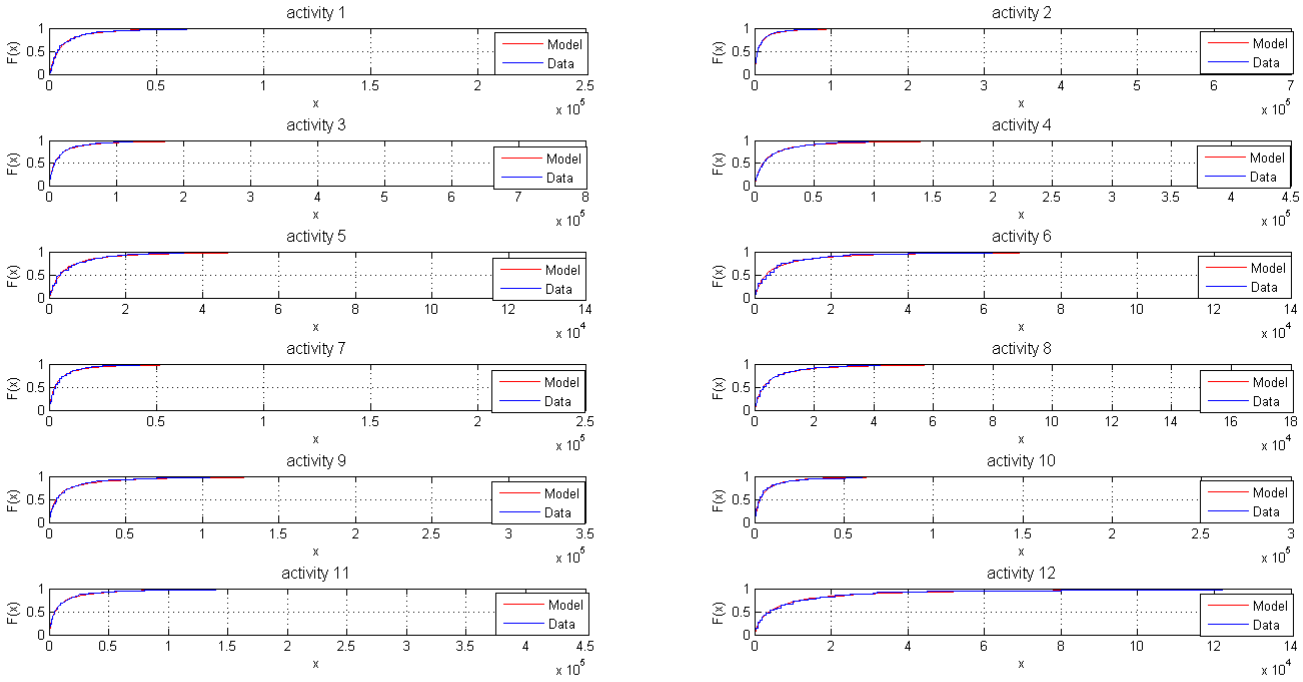


FIGURE B.1 – Comparaison entre la fonction de répartition des données empiriques concernant la distance parcourue et la fonction de répartition de la loi log-normale pour chaque type d'activités. Cette figure est réalisée à partir des codes MATLAB [1] écrits par J. Barthelemy.

1. MATLAB est un langage avec une interface graphique pour le calcul numérique, la visualisation,...

B.2.2 Durée

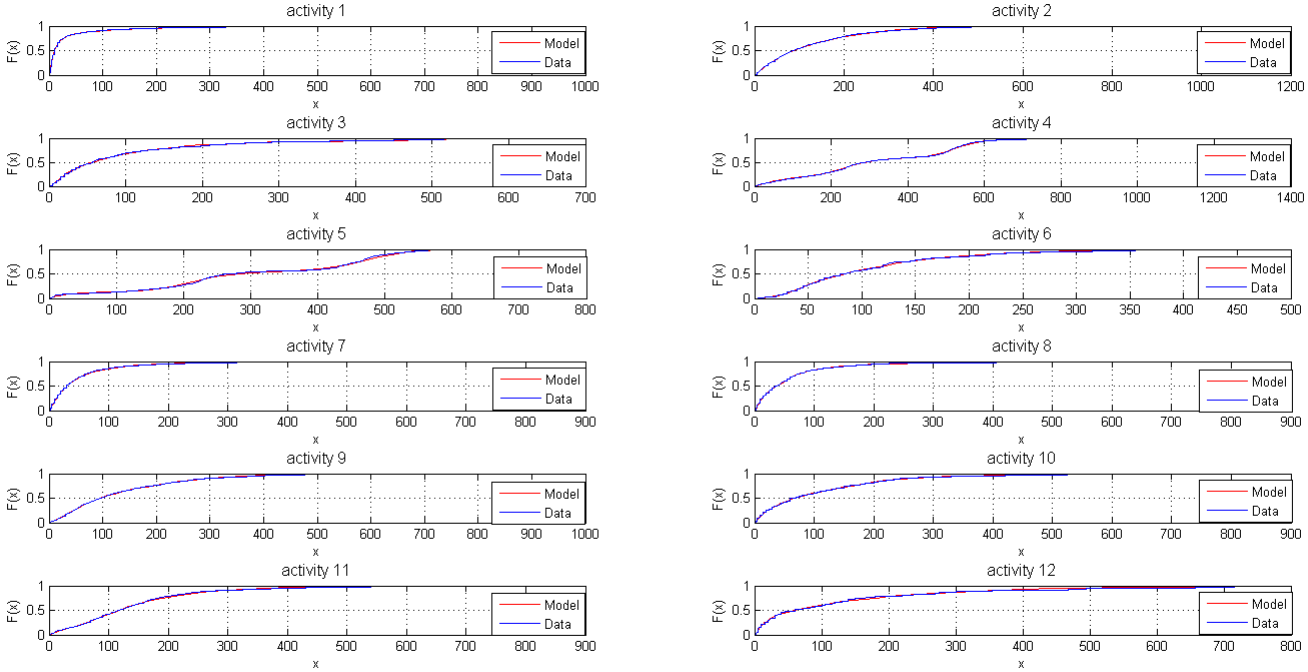


FIGURE B.2 – Comparaison entre la fonction de répartition des données empiriques concernant la durée et la fonction de répartition d'une mixture de loi log-normale pour chaque type d'activités. Cette figure est réalisée à partir des codes MATLAB écrits par J. Barthelemy.

Annexe C

Réseau de VirtualBelgium

Le réseau de VirtualBelgium est créé en instanciant la classe Network qui contient l'ensemble des objets de la classe Node et l'ensemble des objets de la classe Link. En d'autres termes, l'objet de la classe Network possède l'ensemble des nœuds et l'ensemble des chemins qui forment le réseau belge. La table C.1 contient les variables associées à chaque objet de la classe Network.

Variables	**Caractéristiques**
_Nodes	Ensemble de nœuds (objets de la classe Node)
_Links	Ensemble de chemins (objets de la classe Links)

TABLE C.1 – Variables d'un objet de la classe Network

Après avoir instancié la classe Network, il faut créer des nœuds (objets de la classe Node) et des chemins (objets de la classe Link) correspondant à ceux décrits par le fichier xml, extrait de OpenStreetMap et modifié par le logiciel MATSIM.

C.1 Classe Node

Tout objet de la classe Node possède chaque variable décrite par le tableau C.3. Certaines peuvent ne pas être initialisées lors de l'instanciation.

Vu que les nœuds sont créés à partir des informations contenues dans le fichier xml, l'identifiant et les coordonnées des nœud sont directement extraits du fichier. Actuellement, les variables _links_out_id, _key, _index ne sont pas initialisées lors de la création d'un nœud.

Variables	**Valeurs**	**Caractéristique**
_id	entier long	L'identifiant unique du nœud
_x	réel en double précision	La coordonnée x du nœud
_y	réel en double précision	La coordonnée y du nœud
_ins	entier	Le code INS de la commune à laquelle le nœud appartient.
_links_out_id	un vecteur d'entier long	L'ensemble des liens partant de ce nœud.
_key	réel en simple précision	La clé du nœud (distance entre le nœud et une référence source)
_index	entier	L'index du nœud
_indicateur ;	vecteur	Ensemble d'indicateurs relatifs à la commune donnés au départ sous forme tableau excel. Pour l'instant, il n'existe qu'un seul indicateur

TABLE C.2 – Variables d'un nœud

C.2 Classe Link

Tout objet de la classe Link possède chaque variable décrite par le tableau C.3. Certaines de ces variables peuvent ne pas être initialisées lors de l'instanciation.

Variables	**Valeurs**	**Caractéristique**
_id	entier long	L'identifiant unique du chemin
_start_node_id	entier long	L'identifiant du nœud où commence le chemin.
_end_node_id	entier long	L'identifiant du nœud où commence le chemin.
_length	réel.	La longueur en mètre du chemin

TABLE C.3 – Variables d'un chemin

Vu que les chemins sont créés à partir des informations contenues dans le fichier xml, l'identifiant du chemin, le nœud de départ et celui d'arrivée sont directement extraits du fichier.

Annexe D

Données récoltées

D.1 Wallonie et Bruxelles

D.1.1 Enseignement obligatoire

Vous trouverez dans cette section, les tables D.1, D.2, D.3 contenant les premières lignes des listes des établissements scolaires.

FASE école	FASE implantation	Nom école	Adresse	Code postal	Localité
6	15	Ecole fondamentale Clair Soleil (P1-M8-M16)	Rue du Potaerdenberg, 170	1070	BRUXELLES
7	16	Ecole fondamentale communale Les Marronniers	Rue de Douvres 80	1070	BRUXELLES
9	20	Ecole fondamentale Les Tourterelles M5-P8-M4	Rue Odon 22	1070	BRUXELLES
10	21	Ecole fondamentale P9/10 Carrefour	Rue Eloy 114	1070	BRUXELLES

TABLE D.1 – Les premières lignes de la liste concernant les établissements primaires en Wallonie et à Bruxelles

FASE établissement	Nom de l'établissement	Adresse de l'établissement (rue, code postal, localité)			Fase implantation	Adresse de l'implantation (rue, code postal, localité)		
123	ATHENEE ADOLPHE MAX	BOULEVARD CLOVIS 40	1000	BRUXELLES	190	BOULEVARD CLOVIS 40	1000	BRUXELLES
126	ATHENEE ROBERT CATTEAU	RUE ERNEST ALLARD 49	1000	BRUXELLES	193	RUE ERNEST ALLARD 49	1000	BRUXELLES
127	ATHENEE LEON LEPAGE	RUE DES RICHES	1000	BRUXELLES	194	RUE DES RICHES CLAIRES 30	1000	BRUXELLES
130	LYCEE HENRIETTE DACHSBECK	RUE DE LA PAILLE 24	1000	BRUXELLES	198	RUE DE LA PAILLE 24	1000	BRUXELLES

TABLE D.2 – Les premières lignes de la liste concernant les établissements secondaires en Wallonie et à Bruxelles

n°FASE Secondaire	Nom de l'établissement secondaire	Code postal	Localité implantation	n°Fase fondamentale	Nom de l'établissement fondamental	Code postal	Localité
132	INSTITUT SAINT-LOUIS	1000	BRUXELLES-VILLE	114	Ecole fondamentale libre Saint-Louis 2 (Cardinal)	1000	BRUXELLES
132	INSTITUT SAINT-LOUIS	1000	BRUXELLES-VILLE	3169	Ecole fondamentale libre Saint-Louis 1 (Marais)	1000	BRUXELLES
133	C.S. MARIA ASSUMPTA LYCEE MARIA ASSUMPTA	1020	LAEKEN	111	Ecole primaire Maria Assumpta	1020	BRUXELLES

TABLE D.3 – Les premières lignes de la liste concernant les établissements secondaires reliés à un établissement primaire en Wallonie et à Bruxelles

D.1.2 Enseignement non obligatoire

La table D.4 contient les données récoltées sur les universités en Wallonie et à Bruxelles.

		ULg	UCL	ULB	UMONS	FUNDP	FUSL	FUCaM	TOTAL
Total	Belges	15 676	21 143	16 423	4 577	5 166	2 190	1 156	66 331
	Étranger sec belge	373	606	1 318	159	89	168	34	2 747
	Étranger sec étr.	3 155	4 241	6 227	613	391	225	47	14 899
	Total	19 204	25 990	23 968	5 349	5 646	2 583	1 237	83 977

TABLE D.4 – Nombre d'étudiants dans les universités francophones pour l'année 2010-2011

Pour lire ce tableau, il faut prendre en considération quelques remarques. Ce tableau utilise le nombre d'étudiants qui est différent du nombre d'inscriptions. Par exemple, un étudiant qui est en réussite partielle dans une année, est inscrit dans deux années alors qu'il est compté qu'une seule fois comme étudiant. Le nombre d'étudiants participant à l'échange Erasmus n'est pas pris en compte dans ce tableau.

D.2 Flandre

D.2.1 Enseignement primaire

Dans le tableau D.5, nous retrouvons les premières lignes de la liste des établissements primaires en Flandre. Cette liste est triée selon le code postal de l'établissement. Dans les tables D.6, D.7, nous observons les premières lignes de la liste des établissements primaires de Flandre mélangés avec les établissements préscolaires. Cette dernière contient le nombre d'élèves suivant des cours dans chaque école. Dans la table D.8, vous trouverez les informations relatives aux écoles spécialisées de la commune d'Alost.

schoolnummer	naam	hoofdzetel	straat	postcode	gemeente
3673	Gemeentelijke Lagere School	hoofdzetel	Groendreef 16	1000	BRUSSEL-STAD
3889	Vrije Basisschool	hoofdzetel	John Waterloo Wilsonstraat 21	1000	BRUSSEL-STAD
18	GO! basisschool De Kleurdoos Brussel	hoofdzetel	Moutstraat 24	1000	BRUSSEL-STAD
3699	Vrije Basisschool -Sint-Joris	vestiging	Nieuwland 194	1000	BRUSSEL-STAD
106112	Vrije Lagere School -Sint-Jan Berchmanscollege	hoofdzetel	Nieuwland 75	1000	BRUSSEL-STAD

schoolnummer	telefoon	fax	e-mail	url
3673	02-274.09.25	02-274.09.28	nadia.berhmoun@brunette.brucity.be	www.klavertjevierschool.be
3889	02-230.75.28	02-230.48.44	abeullens@tennude.be	
18	02-512.49.79	02-511.39.50	bs.brussel@gemeenschapsonderwijs.be	
369	0471-76.32.15			
106112	02-512.32.81	02-512.32.81	luc.de.coninck@vgc.be	http ://lagere.school.sint-jan-brussel.

TABLE D.5 – Premières lignes de la listes des établissements primaires en Flandre

instellings-nummer	naam school	adres	post-nummer	fusiegemeente
18	GO! basisschool De Kleurdoos Brussel	Moutstraat 24	1000	Brussel
3657	Vrije Kleuterschool	Rogier van der Weydenstraat 28	1000	Brussel
3673	Gemeentelijke Lagere School	Groendreef 16	1000	Brussel
3681	Vrije Basisschool -Maria Boodschap	Vlaamsesteenweg 155	1000	Brussel
3699	Vrije Basisschool -Sint-Joris	Cellebroersstraat 16	1000	Brussel
3889	Vrije Basisschool	John Waterloo Wilsonstraat 21	1000	Brussel
45922	Gemeentelijke Kleuterschool	Groendreef 16	1000	Brussel
106112	Vrije Lagere School -Sint-Jan Berchmanscollege	Nieuwland 75	1000	Brussel

TABLE D.6 – Nombre d'élèves par établissement primaire en Flandre (première partie)

instellings-nummer	teldatum	"Kleuter aantal leerlingen"	"Kleuter omkadering"	"Kleuter lestijden schalen"	"Kleuter SES-lestijden"	"Kleuter additionele lestijden"
18	01-févr-12	165	275	229	46	0
3657	01-févr-12	90	154	133	21	0
3673	01-févr-12	0				0
3681	01-févr-12	90	151	133	18	0
3699	01-oct-12	141	251	199	52	0
3889	01-févr-12	63	122	98	24	0
45922	01-févr-12	194	360	266	94	0
106112	01-févr-12	0				0

"Kleuter ratio leerling/ leerkracht"	"Kleuter lestijden sociale maatr."	"Lager aantal leerlingen"	"Lager omkadering"	"Lager lestijden schalen"	"Lager SES-lestijden"
14,40	0	153	280	214	66
14,03	0	0			
	0	301	554	405	149
14,30	0	141	218	199	19
13,48	0	128	237	183	54
12,39	0	84	164	125	39
12,93	0	0			
	0	300	468	402	66

"Lager additionele lestijden"	"Lager ratio leerling/ leerkracht"	"Lager lestijden sociale maatr."
0	13,11	0
0		0
0	13,04	0
0	15,52	0
0	12,96	0
0	12,29	0
0		0
0	15,38	0

TABLE D.7 – Nombre d'élèves par établissement primaire en Flandre (deuxième partie)

Aantal leerlingen	Jongens	Meisjes	Totaal
Buitengewoon Kleuteronderwijs	25	12	37
Buitengewoon Lager Onderwijs	225	121	346
Buitengewoon Secundair Onderwijs	399	260	659
Totaal	649	393	1 042

TABLE D.8 – Nombre d'étudiants dans l'ensemble des écoles spécialisées de la commune d'Alost

D.2.2 Enseignement secondaire

Dans le tableau D.9, nous retrouvons les premières lignes de la liste des établissements secondaires en Flandre. Cette liste est triée selon le code postal de l'établissement. Dans le table D.10, nous observons les premières lignes de la liste des établissements secondaires de Flandre avec le nombre d'élèves associés.

schoolnummer	naam	hoofdzetel	straat	postcode	gemeente
32284	Instituut Anneessens - Funck	vestiging	Fonteinstraat 2	1000	BRUSSEL-STAD
32284	Instituut Anneessens - Funck	hoofdzetel	Groot Eiland 39	1000	BRUSSEL-STAD
33258	Imelda-Instituut	hoofdzetel	Moutstraat 19	1000	BRUSSEL-STAD
32151	Maria-Boodschaplyceum	hoofdzetel	Moutstraat 22	1000	BRUSSEL-STAD
41954	GO! atheneum Sint-Pieters-Woluwe	vestiging	Moutstraat 24	1000	BRUSSEL-STAD
31963	GO! atheneum Anderlecht	vestiging	Moutstraat 24	1000	BRUSSEL-STAD

schoolnummer	telefoon	fax	e-mail	url
32284	02-279.58.62	02-279.58.73		
32284	02-510.07.50	02-510.07.98	chris.pijpen@brunette.brucity.be	http ://www.anneessens-funck.be
33258	02-511.06.53	02-513.15.71	info@imelda-instituut.be	http ://www.imelda-instituut.be
32151	02-506.89.20	02-513.84.80	mabo@skynet.be	http ://www.mabobrussel.be
41954	02-511.07.42			
31963	02-511.07.42			

TABLE D.9 – Premières lignes de la liste des établissements secondaires en Flandre

Provincie	Postnr	Gemeente	Instelling	Naam instelling	Straat	Huisnr	Postnr	Gemeente
Antwerpen	2000	Antwerpen	28721	Sted. Inst. vr Sierkunsten en Ambachten	Cadixstraat	2	2000	Antwerpen
			28845	Sint-Norbertusinstituut	Amerikalei	47	2000	Antwerpen
			28852	Onze-Lieve-Vrouwecollege	Frankrijklei	91	2000	Antwerpen

Provincie	Teldatum	Aantal lln	"Indicator ""opleiding moeder"" "	Indicator "schooltoelage"	Indicator "thuistaal"
Antwerpen	1/02/2011	1116	441	375	136
Antwerpen	1/02/2011	1138,5	617	603	496,5
Antwerpen	1/02/2011	640	61	76	112

Indicator "buurt"
632
887
343

TABLE D.10 – Nombre d'élèves par établissement secondaire en Flandre

D.3 Demande rédigée pour l'enseignement obligatoire

Bonjour,

Je m'appelle Stéphanie Marchal. Je suis actuellement étudiante en première année de master en sciences mathématiques aux facultés universitaires Notre-Dame de la Paix. Durant mes deux années de master, je réalise un mémoire concernant la modélisation des écoles dans une population virtuelle de la Belgique. Ce mémoire est en collaboration avec un projet ''VirtualBelgium: une plateforme de simulation de la population belge ''dans lequel travaillent L. Hollaert et J. Barthelemy. Mon promoteur est Philippe Toint (professeur aux facultés universitaires Notre-Dame de la Paix).

Je suis à la recherche de données concernant le nombre total d'élèves admis par établissement scolaire en Belgique (ou sinon le nombre actuel d'élèves par établissement).

S'il y a moyen, j'aimerais recevoir ces données concernant les écoles maternelles, primaires, secondaires de la Belgique ainsi que d'autres variables qui me seraient nécessaires pour réaliser ce travail:

Noms des écoles
Niveau : maternel, primaire, secondaire, tous réseaux confondus
n° fase
code postal et localité
Nombre d'élèves par établissement
Nombre d'élèves par année (si possible)
Nombre d'enseignants, personnel auxiliaire d'éducation, direction, personnel administratif (si possible)

Ces données me serviront pour créer un modèle qui me permettra d'attribuer une école à un enfant.

J'espère qu'il y aura moyen de recevoir ces données. Je suis prête à signer un formulaire de confidentialité si nécessaire.

Je vous remercie d'avance pour votre précieuse aide et vous présente mes meilleures salutations.

Stéphanie Marchal
%Rue de la Closière, 3
%5590 CINEY
%083/21.51.71
%0476/49.89.97
%stmarcha@student.fundp.ac.be

D.4 Demande adressée à Madame Ladavid

D.4.1 Échange de courriels

----- Message transféré de christelle.ladavid@cfwb.be -----
Date : Mon, 25 Feb 2013 14:53:13 +0100

De : "christelle.ladavid" <christelle.ladavid@cfwb.be>
Objet : RE: Demande de données pour la réalisation d'un mémoire
À : stmarcha@student.fundp.ac.be

Re-bonjour,

Je me disais également qu'il serait plus opportun que votre demande de données par école soit introduite (càd signée) par votre promoteur.

En effet, conformément au décret, de telles demandes de données doivent être introduites par des personnes de droit public ou par des chercheurs qualifiés ou autres personnes et organismes privés agréés par le ministre compétent et dont les objectifs auront été approuvés par la direction du Service des statistiques.

Bien à vous,

----- Message transféré de christelle.ladavid@cfwb.be -----

Date : Mon, 25 Feb 2013 14:25:36 +0100
De : "christelle.ladavid" <christelle.ladavid@cfwb.be>
Objet : RE: Demande de données pour la réalisation d'un mémoire
À : stmarcha@student.fundp.ac.be

Bonjour,

Comme convenu, je vous invite à trouver, pour information et ajout éventuel, la demande de données par école que je vais adresser à la Ministre concernant votre mémoire.

Si vous aviez davantage de renseignements concernant le projet « VirtualBelgium », quels en sont les objectifs, le contexte ?? Cela pourrait être intéressant car j?avoue que, pour ma part en tout cas, je n?en comprends pas bien la finalité.

Par ailleurs, je tenais également à vous préciser que les données concernant les enseignants, ne relèvent pas de nos compétences. À ce sujet, il convient de vous adresser à l'Administration générale des Personnels de l?Enseignement : www.enseignement.be/index.php?page=25570.

Bien cordialement,

Christelle Ladavid

-----Message d'origine-----
De : stmarcha@student.fundp.ac.be [mailto:stmarcha@student.fundp.ac.be]
Envoyé : vendredi 22 février 2013 18:03
À : christelle.ladavid
Objet : RE: Demande de données pour la réalisation d'un mémoire

Merci pour cette proposition.

Si vous avez besoin de l'appui de mon promoteur, Philippe Toint, vice-Recteur à la recherche et professeur à l'université de Namur, je pourrais lui demander lundi matin.

Je pourrais juste préciser que ce projet fait partie des projet du centre Namurois des Systèmes Complexes et du Groupe de recherche sur les transports et qu'aucune donnée reçue ne sera publiée.

Concernant les numéros FASE, ils ne sont pas nécessaires.

Merci d'avance.

Bon weekend

Bien à vous.
Stéphanie Marchal

-----"christelle.ladavid" <christelle.ladavid@cfwb.be> a écrit :

Concernant le fait de soumettre votre demande à Madame Simonet, je peux bien sûr m'en charger. Je m'en occuperai début de la semaine prochaine sur base des éléments que vous m'avez communiqués. Si vous vous souhaitez m'adresser
plus de renseignements dans cette perspective, n'hésitez pas.

Bonne fin de semaine,

Christelle Ladavid

> -----Message d'origine-----

De : stmarcha@student.fundp.ac.be [mailto:stmarcha@student.fundp.ac.be]
Envoyé : vendredi 22 février 2013 17:08
À : christelle.ladavid
Objet : RE: Demande de données pour la réalisation d'un mémoire

Bonjour,
merci pour cette information.

Vu que mon mémoire se porte sur la simulation des écoles, j'ai besoin d'informations précises sur ces dernières.

Je vais donc essayer de soumettre ma demande à Madame Marie-Dominique Simonet et si ce n'est pas possible, j'utiliserais des données à une échelle plus grande.

Merci.
Bien à vous.

Stéphanie Marchal

------"christelle.ladavid" <christelle.ladavid@cfwb.be> a écrit :

Bonjour,

Je fais suite à votre demande de données par école du 18 février dernier.

À ce sujet, je vous invite à prendre connaissance des éléments de réponse suivants:

- Conformément à l'article 8, paragraphe 4 du « Décret portant diverses mesures en matière de culture, de santé, d'enseignement et de budget » du 27 décembre 1993, nous ne pouvons malheureusement communiquer aucune donnée par établissement scolaire en dehors des services du Gouvernement ou du Ministre compétent :

§ 4. Toutes les données sont absolument anonymes et aucune donnée par école n'est communiquée en dehors des services du Gouvernement et des ministres responsables de l'enseignement sauf :

1° lorsque la communication de telles données est nécessaire à l'exécution d'un engagement international ;
2° Si, à la suite d'une demande expressément motivée sur les objectifs poursuivis par le traitement des données et introduite par des personnes de droit public ou des personnes et organismes visés au § 3, 3e tiret, le Ministre compétent autorise la communication de ces données.
Les ministres, à la demande des organisations représentatives des pouvoirs organisateurs leur fournissent ces données pour les écoles qu'elles fédèrent.

Toutefois, comme vous pouvez le constatez dans le paragraphe 4 repris ci-dessus, il est possible de soumettre votre demande de données par école à Madame Marie-Dominique Simonet, Ministre en charge de l'enseignement obligatoire. Cette procédure prendra plusieurs semaines.

Il vous est également possible de revoir la granularité des données souhaitées (en ne sollicitant pas de données par école).

- Concernant les numéros FASE, il s'agit d'identifiants réservés à l'administration et nous ne pouvons les communiquer en externe.
- Par ailleurs, vous pouvez avoir accès à des listes des établissements scolaires ventilées par niveau d'enseignement via le lien suivant : www.enseignement.be/index.php?page=25949.

Et comme vous le savez peut-être, des données chiffrées concernant l'enseignement obligatoire sont également disponibles sur www.etnic.be/index.php?id=325 et www.enseignement.be/index.php?page=26723&navi=3352.

Je reste à votre disposition si nécessaire à ce sujet.

Bien cordialement,

Christelle Ladavid - Attachée

Affaires générales et intergouvernementales

DG Enseignement obligatoire
Rue A. Lavallée, 1 - 1080 BRUXELLES
Tél: +32 (0)2 690 83 59
Fax: +32 (0)2 690 85 83

D.4.2 Demande adressée à Madame la Ministre Simonet

Bonjour, Madame, Monsieur

Je m'appelle Stéphanie Marchal et suis actuellement étudiante en première année du master en sciences mathématiques aux facultés universitaires Notre-Dame de la Paix à Namur. Durant mes deux années de master, je réalise un mémoire concernant la modélisation des écoles dans une population virtuelle de la Belgique. Ce mémoire est en collaboration avec un projet "VirtualBelgium : une plateforme de simulation de la population belge" pour lequel travaillent entre autres Mademoiselle L. Hollaert et Monsieur J. Barthelemy (membres du département de mathématique). Ce travail fait partie des projets du Centre Namurois des Systèmes Complexes et du Groupe de recherche sur les transports. Mon promoteur est Monsieur Philippe Toint, professeur, Directeur du groupe de recherche sur les transports, Vice-Recteur à la recherche aux facultés universitaires Notre-Dame de la Paix.

Je suis à la recherche de données concernant le nombre d'élèves admis par établissement scolaire en Belgique et ce, tous réseaux confondus. Les données concernant l'année 2010-2011 ou l'année 2011-2012 sont également pertinentes. En complément, les informations relatives à la description de l'établissement sont également nécessaires pour la réalisation du travail :

- Nom de l'établissement
- Niveau : primaire, secondaire,
- Code postal et localité
- Nombre d'élèves par établissement
- Nombre d'élèves par année (si possible)

Avec les données reçues, notre objectif sera de réaliser un modèle d'attribution d'écoles aux enfants d'un ménage de la population synthétique.

Dans l'espoir de recevoir une réponse favorable à cette demande, je vous confirme d'ores et déjà que la confidentialité sera d'application et si nécessaire, je suis prête à signer un formulaire.

Je vous remercie d'avance pour votre précieuse aide et vous présente mes meilleures salutations.

Stéphanie Marchal
Etudiante en Première année du Master Math

FUNDP
Année académique 2012-2013

D.4.3 Décret

Dans cette annexe, nous citons l'article 8, du paragraphe 4 du « Décret portant diverses mesures en matière de culture, de santé, d'enseignement et de budget du 27 décembre 1993 ».

§ 3. Les données recueillies sont traitées par les agents du Service des statistiques et des directions générales d'enseignement concernées, qui les regroupent en vue du calcul de l'encadrement et du financement ainsi que de l'élaboration de données statistiques destinées :

- à la publication d'informations sur l'état de l'enseignement en Communauté française ;
- à la documentation des services nationaux, étrangers et internationaux officiellement reconnus ;
- et à celle des chercheurs qualifiés ou autres personnes et organismes privés agréés par le ministre compétent et dont les objectifs auront été approuvés par la direction du Service des statistiques.

§ 4. Toutes les données sont absolument anonymes et aucune donnée par école n'est communiquée en dehors des services du Gouvernement et des ministres responsables de l'enseignement sauf :

1. lorsque la communication de telles données est nécessaire à l'exécution d'un engagement international ;

2. Si, à la suite d'une demande expressément motivée sur les objectifs poursuivis par le traitement des données et introduite par des personnes de droit public ou des personnes et organismes visés au § 3, 3e tiret, le Ministre compétent autorise la communication de ces données. Les ministres, à la demande des organisations représentatives des pouvoirs organisateurs leur fournissent ces données pour les écoles qu'elles fédèrent.

D.5 Recolte de données concernant les Hautes Ecoles à Bruxelles et en Wallonie

D.5.1 Démarches

Pour la recolte de données concernant les Hautes Ecoles à Bruxelles et en Wallonie, un courriel a été envoyé à madame Chantal Kaufmann, la directrice générale de l'enseignement non-obligatoire. Cependant, une réponse négative provenant de madame Brigette Morue travaillant à l'Observatoire de l'enseignement supérieur, a été reçue en nous indiquant le même décret que celui cité dans la partie relative à l'enseignement obligatoire. Elle explique que « L'Observatoire de l'enseignement supérieur dispose d'une base de données reprenant par année académique les inscriptions et les diplômes des étudiants inscrits dans l'enseignement supérieur non universitaire (hautes écoles, écoles supérieures des arts) » mais elle ne peut pas les communiquer. La seule alternative possible est de revoir le niveau d'agrégation. En effet, elle peut transmettre les données concernant les étudiants par arrondissement pour l'année scolaire 2011-2012 ou d'autres années si nécessaire.

Dès lors, la dernière démarche entreprise pour obtenir éventuellement ces informations est une demande écrite au ministre responsable, monsieur Jean-Claude MARCOURT. Monsieur Toint propose d'être l'intermédiaire pour transmettre la demande et ainsi pouvoir l'appuyer. Une copie de la demande adressée à Monsieur le Ministre se trouve dans la section suivante.

D.5.2 Demande rédigée pour Monsieur Marcourt

Bonjour, Madame, Monsieur

Je m'appelle Stéphanie Marchal et suis actuellement étudiante en première année du master en sciences mathématiques aux facultés universitaires Notre-Dame de la Paix à Namur. Durant mes deux années de master, je réalise un mémoire concernant la modélisation des écoles dans une population virtuelle de la Belgique. Ce mémoire est en collaboration avec un projet "VirtualBelgium : une plateforme de simulation de la population belge" pour lequel travaillent entre autres Mademoiselle L. Hollaert et Monsieur J. Barthelemy (membres du département de mathématique). Ce travail fait partie des projets du Centre Namurois des Systèmes Complexes et du Groupe de recherche sur les transports. Mon promoteur est Monsieur Philippe Toint, professeur, Directeur du groupe de recherche sur les transports, Vice-Recteur à la recherche aux facultés universitaires Notre-Dame de la Paix.

Je suis à la recherche de données concernant le nombre d'étudiants admis par établissement de l'enseignement supérieur en Belgique. En complément, les informations relatives à la description de l'établissement sont également nécessaires pour la réalisation du travail :

- Nom de l'établissement
- Code postal et localité
- Nombre d'étudiants par établissement

Avec les données reçues, notre objectif sera de réaliser un modèle d'attribution d'une Haute Ecole aux étudiants de la population synthétique.

Dans l'espoir de recevoir une réponse favorable à cette demande, je vous confirme d'ores et déjà que la confidentialité sera d'application et si nécessaire, je suis prête à signer un formulaire.

Je vous remercie d'avance pour votre précieuse aide et vous présente mes meilleures salutations.

Stéphanie Marchal
Etudiante en Première année du Master Math
FUNDP
Année académique 2012-2013
0476/49.89.97
stmarcha@student.fundp.ac.be

Annexe E

Concepts de la programmation orientée objet

Cette annexe est principalement inspirée du cours programmation orientée-objet de Monsieur Bernard Boigelot[9], donné à l'université de Liège. La référence [41] a également été consultée.

La programmation orientée objet est une technique de programmation qui consiste à développer une partie d'un programme qui interagit avec les autres parties sans les connaitre en détails. Plusieurs personnes peuvent donc travailler sur le même programme sans qu'elles connaissent en détails la partie de leurs collègues.

Cette technique de programmation possède son propre langage. Vous trouverez dans la table E.1, les définitions des différents concepts qui sont cités dans ce travail.

Concept	Définition
Classe	Ensemble de variables et de méthodes. Une classe est instanciée pour obtenir un objet. Elle représente la structure d'un objet.
Classe principale	Classe qui possède une méthode Main Cette méthode permet d'exécuter le programme.
Instance d'une classe	un objet
Méthode	Opération réalisable par un objet. Elle peut être caractérisée de fonction d'une classe.
Objet	Structure de données présentes dans la mémoire de l'ordinateur au cours de l'exécution d'un programme
Programme Orienté-objet	Ensemble de classes dont la classe principale
Héritage	Création d'une classe qui sont dérivées d'autres classes et qui a donc accès aux variables et méthodes de ces dernières
Single Singleton	Un concept en programmation qui permet une instance unique d'une classe

TABLE E.1 – Définitions des concepts de la programmation orientée objet

Annexe F

Modèle d'assignation d'une école à un étudiant

F.1 Ensemble de modifications de la valeurs d'epsilon

Dans cette section, vous trouverez les modifications du premier ensemble, appliquez à la valeur d'epsilon comme expliqué dans le point 8.4.3.2. de la section 4.3.4.2.

- **Si dist $<$ 1000**

Soient : dist — la distance générée par le générateur à l'étape 2 du modèle n°3.
dist2 — une deuxième distance générée par le générateur à l'étape 2 du modèle n°3.
Epsilon — la valeur de l'epsilon,
$n_{\text{noeud}_{\text{selectionné}}}$ — le nombre de noeuds représentant une école, sélectionnés à l'étape 4 du modèle

Étape	Distance générée	Valeur Epsilon	Prochaine action
1	dist	250	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°2 sinon STOP
2	dist	dist * 0.25	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°3 sinon STOP
3	dist	dist * 0.50	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°4 sinon STOP
4	dist	dist * 0.75	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°5 sinon STOP
5	dist	dist	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°6 sinon STOP
6	dist	dist * 1.25	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°7 sinon STOP
7	dist	dist * 1.50	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°8 sinon STOP
8	dist	dist * 1.75	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°9 sinon STOP
9	dist	dist * 2	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°10 sinon STOP
10	dist	dist * 2.25	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°11 sinon STOP
11	dist	dist * 2.50	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°12 sinon STOP
12	dist	2500	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°13 sinon STOP
13	dist	5000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°1 avec distance = dist2 sinon STOP

- **Si dist $\geq$ 1000 et dist $<$ 15000**

Étape	Distance générée	Valeur Epsilon	Prochaine action
1	dist	250	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°2 sinon STOP
2	dist	dist * 0.25	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°3 sinon STOP
3	dist	dist * 0.50	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°4 sinon STOP
4	dist	dist * 0.75	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°5 sinon STOP
5	dist	dist	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°6 sinon STOP
6	dist	dist * 1.25	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°7 sinon STOP
7	dist	dist * 1.50	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°8 sinon STOP
8	dist	dist * 1.75	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°9 sinon STOP
9	dist	2500	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°10 sinon STOP
10	dist	5000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°1 avec distance = dist2 sinon STOP

- **Si dist $\geq$ 1500 et dist $<$ 2000**

Étape	Distance générée	Valeur Epsilon	Prochaine action
1	dist	250	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°2 sinon STOP
2	dist	dist * 0.25	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°3 sinon STOP
3	dist	dist * 0.50	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°4 sinon STOP
4	dist	dist * 0.75	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°5 sinon STOP
5	dist	dist	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°6 sinon STOP
6	dist	2500	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°7 sinon STOP
7	dist	5000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°1 avec distance = dist2 sinon STOP

- **Si dist $\geq$ 2000 et dist $<$ 5000**

Étape	Distance générée	Valeur Epsilon	Prochaine action
1	dist	250	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°2 sinon STOP
2	dist	dist * 0.25	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°3 sinon STOP
3	dist	dist * 0.50	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°4 sinon STOP
4	dist	dist * 0.75	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°5 sinon STOP
5	dist	2500	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°6 sinon STOP
6	dist	5000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°1 avec distance = dist2 sinon STOP

- **Si dist >= 5000**

Étape	Distance générée	Valeur Epsilon	Prochaine action
1	dist	250	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°2 sinon STOP
2	dist	1000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°3 sinon STOP
3	dist	1500	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°4 sinon STOP
4	dist	2000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°5 sinon STOP
5	dist	2500	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°6 sinon STOP
6	dist	4000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°7 sinon STOP
7	dist	5000	Si $n_{\text{noeud}_{\text{selectionné}}} = 0$ alors Etape n°1 avec distance = dist2 sinon STOP