Conception de modèles et d'outils de géosimulation

Robin Cura

21 octobre 2011

Rapport du stage de fin d'étude du Master Carthagéo,
effectué à l'UMR 8504 Géographie-cités,
sous la direction de Thomas Louail, Sébastien Rey-Coyrehourcq et Clara Schmitt,
encadré pédagogiquement par Joël Boulier.

JURY

Président du Jury
Francis DHÉE et Antonine RIBARDIÈRE

Commanditaire :
CNRS - UMR 8504 – Géographie-cités

Encadrement de stage :
Thomas LOUAIL
Sébastien REY-COYREHOURCQ
Clara SCHMITT

Responsable pédagogique :
Joël BOULIER

BORDEREAU D'INDEXATION
TITRE : Conception de modèles et d'outils de géosimulation.
AUTEUR : Robin CURA
CONFIDENTIALITE : Publique
MOTS CLES : Modélisation et simulation à base d'agents, NetLogo, programmation d'un plugin SIG, Quantum-GIS, cartographie de flux, outil de visualisation des trajectoires de villes.
RÉSUMÉ : Les travaux présentés dans ce manuscrit ont été effectués au cours d'un stage d'ingénieur géomaticien de 6 mois dans l'équipe PARIS du laboratoire Géographie-Cités. Nous présentons dans une première partie les réalisations en lien avec des modèles de simulation spatiale, *SimpopLocal* et *SimpopNet*. Ce travail inclut la conception et le développement de mécanismes pour enrichir ces modèles et permettre d'aborder de nouvelles questions, ainsi que la réalisation d'un outil de visualisation de trajectoires de villes, *TrajPop*. Dans la seconde partie, nous présentons les travaux réalisés pour améliorer la cartographie des flux sous SIG. Nous avons développé un *plugin* pour le SIG libre QGIS, *qFlowMapper*, en repartant d'un algorithme efficace de la littérature en visualisation.
SITUATION DU DOCUMENT : Rapport de stage de fin d'études du Master Carthagéo – Option thématique.
NOMBRE DE PAGES TOTAL : 58 pages.
SYSTEME HOTE : LaTeX

MODIFICATIONS			
EDITION	**REVISION**	**DATE**	**PAGES MODIFIEES**
1.0		21/10/2011	Création

Remerciements

Je tiens avant tout à assurer toute ma gratitude à l'équipe qui a accepté de m'encadrer durant ces cinq mois : Thomas Louail en premier lieu, qui n'a pu travailler directement avec moi passé la formation à NetLogo, mais a néanmoins porté un intérêt significatif et constant dans le déroulement du stage, et s'est montré un soutient sans faille, motivant et rassurant, dans l'épreuve de la rédaction du présent rapport.
À Clara Schmitt ensuite, pour la pédagogie dont elle a su faire preuve au quotidien pour me guider dans le monde de la modélisation informatique et des systèmes de villes.
À Sébastien Rey-Coyrehourcq enfin, pour son aide et sa curiosité technique, ainsi que pour l'intérêt partagé pour la veille technologique en géomatique.

Mes plus sincères remerciements sont aussi adressés à Joël Boulier, pour le suivi effectué au cours du stage tout autant que son aide dans l'organisation de ce rapport.

J'aimerais aussi témoigner de toute ma reconnaissance à Denise Pumain, pour son aide précieuse dans le développement de SimpopNet-Réseaux, de TrajPop, ainsi que pour les éclaircissements théoriques apportés tout au long du stage, à Arnaud Banos pour ses conseils sur le choix des algorithmes à utiliser dans SimpopNet-Réseaux, et à Hélène Mathian pour ses remarques sur qFlowMapper.

Je tiens enfin à saluer et remercier l'ensemble des membres du laboratoire, doctorants et vacataires en particulier, pour l'ambiance chaleureuse et le climat d'entraide apportés à ce cadre de travail si plaisant.

Table des matières

Introduction **5**

1 L'inscription du travail du stagiaire dans le cadre d'un laboratoire de recherche. **6**
1.1 Cadre institutionnel. 7
1.2 Pratique de la simulation informatique dans l'équipe P.A.R.I.S. 8
1.2.1 Rôle croissant de la simulation informatique comme méthode de recherche en géographie. 8
1.2.2 L'usage des SMA au laboratoire. 9
1.2.3 Analyse des résultats de simulations et besoins d'outils de visualisation de sorties de modèles spatiaux. 10
1.3 Inscription du travail dans les différentes phases d'un projet de simulation. . . 11
1.3.1 Travail du géographe modélisateur. 11
1.3.2 Prendre part à toutes les étapes de la modélisation. 12

2 Enrichissement et création de modèles. **15**
2.1 Modéliser à l'aide des SMA. 16
2.1.1 Logiques d'ensemble de la création d'un modèle 16
2.1.2 Présentation de NetLogo 16
2.1.3 Un exemple pratique et formateur 17
2.2 L'enrichissement d'un modèle : SimpopLocal. 18
2.2.1 Mécanismes de SimpopLocal. 18
2.2.2 Mise en place du module de catastrophes. 20
2.2.3 Résultat des catastrophes. 22
2.3 SimpopNet-Réseaux. 22
2.3.1 Introduction. 22
2.3.2 Mécanismes et hypothèses de départ. 24
2.3.3 Création de SimpopNet-Réseaux. 26
2.3.4 Résultats. 29

3 Création d'outils d'analyses de trajectoires spatio-temporelles. **31**
3.1 Permettre la visualisation - Première approche pour SimpopLocal. 32
3.2 TrajPop. 33
3.2.1 Un outil d'analyse automatique des trajectoires de villes. 33
3.2.2 Les rapports produits. 36

3.3 qFlowMapper. 40
3.3.1 État de l'art et besoins. 40
3.3.2 Outil de création et d'export de courbes de Bézier. 45
3.3.3 Regroupement de flux selon la méthode FDEB. 47
3.3.4 Continuer le développement. 51

4 Conclusion : limites et perspectives. 52
4.1 Travail effectué et objectifs initiaux. 53
4.1.1 Les modèles. 53
4.1.2 Autour des modèles - Pilotage et analyse de résultats. 53
4.2 Acquis et débouchés. 54
4.2.1 La découverte du monde de la recherche. 54
4.2.2 Une formation permanente. 55
4.2.3 Continuer les travaux. 56

Bibliographie 56

Introduction

Mon stage s'est déroulé à l'UMR Géographie-cités, sur une durée de 5 mois, de Mai à Octobre 2011. Le sujet initial, « Conception de modèles et d'outils de géographie computationnelle », prévoyait trois phases de travail partiellement successives. La première, occupant la totalité du premier mois, visait à la formation aux Systèmes Multi-Agents, par la lecture d'articles et d'ouvrages, ainsi que par la découverte de la logique de modélisation orientée agents et par l'apprentissage de la plate-forme de simulation NetLogo. Sur les quatre mois restant, trois étaient consacrés à la participation à la création du modèle SimpopNetainsi qu'à la production d'outils de pilotage et de visualisation des résultats pour les modèles SimpopLocal et SimpopNet, et le dernier mois devait permettre la réalisation d'une documentation sur les outils produits et la rédaction du rapport.

Les encadrants, Thomas Louail, post-doctorant au laboratoire, Sébastien Rey-Coyrehourcq et Clara Schmitt, tous deux doctorants dans l'UMR, recherchaient des compétences en analyse spatiale, en SIG et en développement. Le Master Carthagéo formant à ces disciplines, et en raison de mon intérêt pour les problématiques abordées, ma candidature a été retenue. Les deux principales phases du stage s'inscrivaient dans le cadre des thèses des deux doctorants, et dès lors, étaient dépendantes et conditions de leurs propres travaux de recherche, c'est-à-dire que si j'étais parfois tributaire d'eux pour avancer mes tâches, l'apport réalisé pendant le stage devait pouvoir leur servir pour leurs thèses.

Si le sujet du stage a évolué, de même que l'ordonnancement des tâches ou encore les tâches en elles-mêmes (voir figure 4 page 14), le domaine d'étude est resté constant, permettant, par la diversification des objets d'études affectés, de prendre part à l'ensemble des composantes d'un projet de modélisation informatique. Si le développement constitue une part importante de celles-ci et donc des travaux effectués au cours du stage, j'ai choisi de suivre une approche moins technique au sein de ce rapport, privilégiant un retour sur l'ensemble du stage avec un regard tourné vers l'ingénierie. Je ne reproduirais donc pas de code au sein du rapport, à l'exception des cas où celui-ci aide à la compréhension des démarches entreprises. Les programmes développés seront joints en version numérique et librement ré-utilisables, comme le veut la démarche du logiciel libre qui a été durant ce stage employé autant que possible, par nécessité (l'ordinateur mis à disposition ayant GNU/Linux comme système d'exploitation) et par choix de l'équipe encadrante.

Centré autour de la participation à toutes les étapes de conception d'un modèle, le rapport introduit d'abord l'environnement et la démarche du stage, avant de présenter dans les deux parties suivantes les phases de conception de modèle puis de constitution d'outils permettant l'exploitation de leurs résultats. On conclura ce retour sur les grandes lignes des tâches effectuées par une démarche critique et auto-réflective sur le déroulement du stage.

Première partie

L'inscription du travail du stagiaire dans le cadre d'un laboratoire de recherche.

1.1 Cadre institutionnel.

L'UMR Géographie-Cités. Le stage a été réalisé au sein de l'Unité Mixte de Recherche (UMR) 8504 Géographie-cités. Ce laboratoire de recherche a été l'un des fondateurs,dans les années 1970, de la géographie théorique et quantitative, laquelle est depuis restée l'un de ses axes d'études principaux. L'UMR est composé de trois équipes dont les domaines de recherches couvrent une partie importante du spectre de la géographie française : le C.R.I.A (Centre de recherches sur les Réseaux, l'Industrie et l'Aménagement), E.H.GO (Épistémologie et Histoire de la Géographie) et P.A.R.I.S (Pour l'Avancement des Recherches sur l'Interaction Spatiale). C'est dans cette dernière que s'est déroulé le stage, dans les locaux qui lui sont attribués 13 rue du Four. L'équipe regroupe des chercheurs autour de l'étude des interactions, en particulier en ce qu'elles définissent des systèmes. Les membres de l'équipe sont porteurs et héritiers des méthodes de la géographie théorique et quantitative, et réceptifs aux possibilités apportées par les statistiques spatiales, de géomatique et de modélisation pour aider à définir des cadres théoriques aux interactions spatiales. L'équipe compte plus d'une centaine de membres, mais ceux-ci sont répartis selon des réseaux d'intérêts dans différents locaux parisiens. On notera la forte et fréquente présence des doctorants dans les locaux du laboratoire, qui participent activement du climat d'échange qui caractérise l'équipe.

L'ISC-PIF. Une partie du stage a aussi été effectuée dans les locaux de l'Institut des Systèmes Complexes - Paris Île-de-France, dont le directeur, Arnaud Banos, est chargé de recherche dans l'équipe P.A.R.I.S. L'ISC est un groupement d'intérêt scientifique pluri-disciplinaire rassemblant des chercheurs et doctorants de l'ensemble des domaines de recherche étudiant les systèmes complexes, essentiellement à l'aide de modélisations mathématiques et informatiques. Outre l'appartenance du directeur à Géographie-cités, l'ISC-PIF et l'UMR Géographie-cités ont un historique de collaboration active dans la mise en place de dispositifs d'étude de systèmes spatiaux complexes (mobilité individuelle dans la ville, coévolution et mise en réseau des villes). Bien qu'il existe une ancienne tradition à l'UMR de collaboration avec des chercheurs provenant des sciences physiques et de la nature [PSJS89], les projets en commun avec l'ISC-PIF ont néanmoins permis d'élargir les outils et méthodes d'appréhension de la complexité géographique.

Programmes de recherche et polarités. La mission initiale du stage consistait à enrichir deux projets logiciels, SimpopNet et SimProcess que l'on présentera peu après. Ces projets sont eux-même à l'interface entre plusieurs projets de recherche du laboratoire, parmi lesquels *TransMonDyn* (ANR),qui étudie les dynamiques d'installation de d'évolution des systèmes de peuplement sur le temps long, et *GeoDiverCity* (ERC), qui tend à étudier et catégoriser la forme des systèmes de villes en différents endroits du monde et à différentes époques. Si les objectifs du stage ont évolué au cours de celui-ci, l'ancrage dans *GeoDiverCity* a été en se renforçant.

1.2 Pratique de la simulation informatique dans l'équipe P.A.R.I.S.

1.2.1 Rôle croissant de la simulation informatique comme méthode de recherche en géographie.

Analyse spatiale et simulation informatique. La modélisation est avec l'expérimentation l'une des deux composantes principales de l'activité scientifique. Les mots "modèle", "modélisations", "modéliser", sont extrêmement généraux et renvoient à une très grande variété de raisonnements, pratiques, méthodes, ...Si une modélisation commence toujours par le choix d'un référentiel d'observation et d'un ensemble fini d'observables pour décrire le phénomène étudié, il y a ensuite, du dessin jusqu'au modèle formel à fort degré d'abstraction, un large panel d'approches de la modélisation qui répondent à des besoins différents. En analyse spatiale, les *modèles dynamiques de simulation spatiale*, qui décrivent et calculent comment les choses évoluent *dans le temps* et *dans l'espace*, sont mobilisés depuis le début des années 1970. Ces modèles permettent notamment de mettre à l'épreuve des raisonnements hypothético-déductifs en les calculant après les avoir formalisés. L'apport de l'informatique à la modélisation de systèmes dynamiques spatialisés a permis la création de modèles bien plus fins et de large envergure que ne le permettaient les séries d'équations mobilisées jusque là, qu'on devait garder suffisamment simples pour être résolvables analytiquement. L'élaboration d'un modèle de simulation peut poursuivre différentes pistes : compréhensive, explicatoire, prédictive, heuristique et pédagogique, etc. Dans le contexte de l'équipe P.A.R.I.S., les modèles dynamiques ont le plus souvent été conçus à la fois pour l'intérêt de l'exercice de modélisation en lui-même, qui oblige à un nécessaire filtrage entre l'essentiel et le spécifique, mais aussi comme une approche de test des théories explicatives de la mise en réseau, de la structuration hiérarchique, de la diversification fonctionnelle et de la diffusion spatiale des innovations dans les systèmes de villes.

Les Systèmes Multi-Agents (SMA). Les systèmes multi-agents constituent une approche individu-centrée de formalisation et de calcul de systèmes dynamiques distribués et spatialisés. Les composants ou acteurs du systèmes y sont représentés par des *agents*, dotés de capacité d'*interaction*. Les agents sont situés dans des *environnements* qui peuvent être multiples (spatial, social [groupes], etc.). Concevoir et écrire un modèle multi-agents (on dit aussi « à base d'agents ») consiste alors à formaliser les hypothèses que l'on fait au niveau des entités constituant le système ainsi que celles que l'on fait sur la façon dont les entités interagissent, et de voir à l'exécution du modèle les dynamiques collectives produites par ces agents en interaction lorsqu'ils s'exécutent en parallèle. Leur utilisation permet de mettre en évidence comment des règles simples, lorsqu'elles sont suivies par de très nombreuses entités évoluant dans un environnement commun et interagissant au niveau microscopique, peuvent donner naissance à des phénomènes collectifs, macroscopiques, complexes. Le vocabulaire très anthropomorphique qui est associé aux SMA résonne beaucoup avec celui des sciences humaines et sociales, et permet aux modélisateurs de ces communautés de ses les approprier rapidement. Pour le géographe, la notion d'agent est explicite et facilite donc la création de faits stylisés tout autant que la définition de ces agents. La notion d'agent pourra par exemple servir à représenter une entité atomique de la géographie (un individu, une route, un élément de bâti par exemple dans le cadre des études de mobilité urbaine), mais aussi un agrégat (une ville par exemple, définie par sa population, sa spécialisation ou encore son environnement géographique).

Les SMA en géographie. L'usage des SMA en géographie a été initié par le laboratoire Géographie-cités, à travers le développement du premier modèle Simpop [BGPM+96], qui visait à étudier l'émergence, la structuration hiérarchique et la spécialisation fonctionnelle d'un système de peuplements dans la phase de transition urbaine. Le succès de ce projet a entraînée la création d'un second modèle, au début des années 2000, plus mû re, performant, intégrant plus d'agents et adaptable à des systèmes de villes variés : Simpop2. Ce modèle intègre des mécanismes enrichis tels que les effets de concurrence entre villes, ou encore ceux de l'interaction spatiale sur leur croissance et spécialisation. Conçu comme un modèle générique de système de villes, Simpop2 a ensuite été adapté à certains cadres d'études spécifiques, tels que les États-Unis ou encore la prospective urbaine européenne (Eurosim) [1].

1.2.2 L'usage des SMA au laboratoire.

Les résultats obtenus dans Simpop2 ayant entraîné de nouveaux questionnements, et devant la lourdeur d'un modèle conçu comme générique mais finalement difficilement adaptatif pour des raisons techniques [2], une nouvelle génération de modèles a été conçu à partir de 2010. Ces modèles se veulent plus spécifiques, non pas à une étendue géographique, mais visent à étudier quelques phénomènes précis, et non à modéliser l'ensemble des dynamiques urbaines. Cette nouvelle série de modèles est organisée autour d'une plate-forme en construction qui doit permettre, avec des modèles fonctionnant sur le même logiciel, de faciliter le pilotage des expérimentations, notamment par le biais d'exécutions sur une grille de calcul, de génération semi-automatique de rapports de simulations, et grâce à l'intégration de données géographiques, permettant l'analyse et la cartographie des résultats dans un SIG, outil que les géographes connaissent et maîtrisent souvent. Nous présentons ici brièvement les modèles de cette nouvelle génération sur lesquels on avait à intervenir.

SimpopLocal. Premier modèle de la série, SimpopLocal se veut un outil permettant d'analyser la constitution d'un système de peuplement hiérarchisé et structuré sous l'influence de contraintes géographiques fortes. Elles sont stylisés par une attribution limitée en ressources aux foyers de peuplements, ressources qui conditionnent la croissance démographique. La structuration du système s'appuie sur une logique d'innovations (technologiques, sociales, politiques) qui permettent aux agents de voir leurs ressources augmenter quand ils voient apparaître une innovation ou quand ils copient une innovation d'un foyer de peuplement voisin. Il a été développé par Clara Schmitt et Sébastien Rey-Coyrehourcq dans le cadre de leurs thèses respectives.

SimpopNet. Ce modèle, encore en conception, doit permettre de simuler la création d'un réseau urbain dans le cadre d'une économie de marché. Tirant lui aussi parti des innovations, il devrait toutefois accorder un rôle prépondérant aux interactions spatiales entre villes reliées au sein d'un réseau routier ou ferré en plein développement dans la période de révolution industrielle que SimpopNet cherche à modéliser. SimpopNet constitue l'un des trois modèles que Clara Schmitt conçoit dans le cadre de sa thèse.

SimProcess. SimProcess n'est pas un modèle mais une plate-forme d'exploitation [3] de modèles qui s'inscrit aussi dans cette nouvelle génération d'outils, qui doit permettre de systématiser et d'automatiser l'analyse multi-scalaire des sorties de simulation. La conception et la mise au point de cette SimProcess constitue le cœur de la thèse de Sébastien Rey-Coyrehourcq.

1. On pourra trouver de plus amples détails sur Simpop2 ainsi que sur les modèles dérivés sur le site du projet : `http://www.simpop.parisgeo.cnrs.fr/`

2. L'obsolescence des bibliothèques logicielles employées rendaient difficiles sa reprise et son extension.

3. Définition de plan d'expérience, déploiement, pilotage, calibrage et production de rapports.

On peut citer d'autres modèles dans cette nouvelle vague, tels que SimpopNano (développé par Thomas Louail, cf. [Lou10]), qui étudie l'évolution intra-urbaine et doit être couplé à Simpop2 pour tester des hypothèses sur les processus d'interaction entre les différentes échelles, ou encore SimpopClim (dernier modèle devant être créé par Clara Schmitt) qui devrait modéliser les évolutions d'un système de villes dans le cadre de politiques urbaines entraînées par le changement climatique.

1.2.3 Analyse des résultats de simulations et besoins d'outils de visualisation de sorties de modèles spatiaux.

Méthodes. SimProcess doit permettre la production de rapports semi-automatisés sur les simulations, ce qui constitue un objectif important en raison de l'inexistence actuelle d'outils de ce type. Pour la création d'indicateurs statistiques, cela servira à accélérer et à uniformiser le processus, chaque modélisateur utilisant actuellement des outils et méthodes propres de traitement sur les tableaux de données générées par les simulations. Dans le cas de la production cartographique, l'enjeu est plus important, en ce qu'elle se cantonnait, dans Simpop2, à la visualisation des cartes dans un *Explorateur Simpop*, sans aucune possibilité de traitement géographique sur les données représentées (cf. Figure 1 page 10).

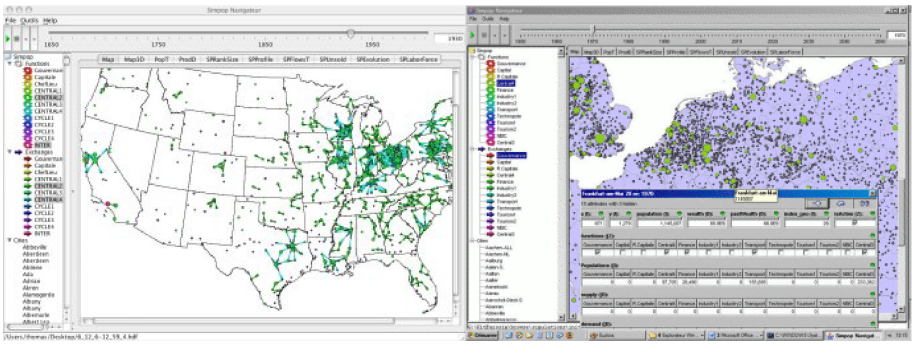

FIGURE 1 – L'explorateur Simpop

Le rendu des flux en SIG. Si l'ajout d'une production de flux géométriques vectoriels dans Simpop2 aurait pu se faire sans grande complexité, leur utilisation pose pourtant problème. La majorité des SIG n'est en effet pas du tout conçue pour analyser et cartographier les flux. Si cela est tout de même possible en considérant les flux comme de simples figurés linéaires, l'esthétique des cartes produites sera alors de bien faible niveau, et avec elle, la lisibilité de la carte le sera tout autant. Si la cartographie de flux est fréquente pour tout cartographe, on peut s'étonner qu'elle ne soit pas mieux intégrée dans des outils censés faciliter la création de cartes. On ne peut l'expliquer que par l'important cloisonnement existant entre les domaines du SIG et de la cartographie. En effet, le plus souvent, les utilisateurs de SIG publient leurs cartes telles qu'elles sont produites par leur logiciel SIG, sans chercher à les retoucher pour y appliquer des modifications esthétiques qui pourraient pourtant rendre leurs cartes bien plus fonctionnelles. A l'inverse, dans le monde de la cartographie, le SIG n'est considéré que comme un outil de conversion de données, qu'elles soient vectorielles, pour utilisation comme fond de carte dans un logiciel de DAO, ou numérique, par exemple pour discrétiser les entités géographiques selon un attribut et mener ensuite la cartographie à la main en DAO. De par ce cloisonnement, peu d'efforts sont faits dans un sens ou dans l'autre pour adapter les outils existants au traitement des flux, traitement nécessairement plus complexe que celui d'entités

zonales ou ponctuelles standards en raison de la forte adaptation nécessaire des méthodes de représentation aux données analysées.

1.3 Inscription du travail dans les différentes phases d'un projet de simulation.

1.3.1 Travail du géographe modélisateur.

La conception d'un modèle suit des étapes séquentielles en apparence, mais exigeant dans les faits des allers- retours constants entre les phases d'analyse, de conception, de développement et de test, comme l'illustre la figure 2 page 11.

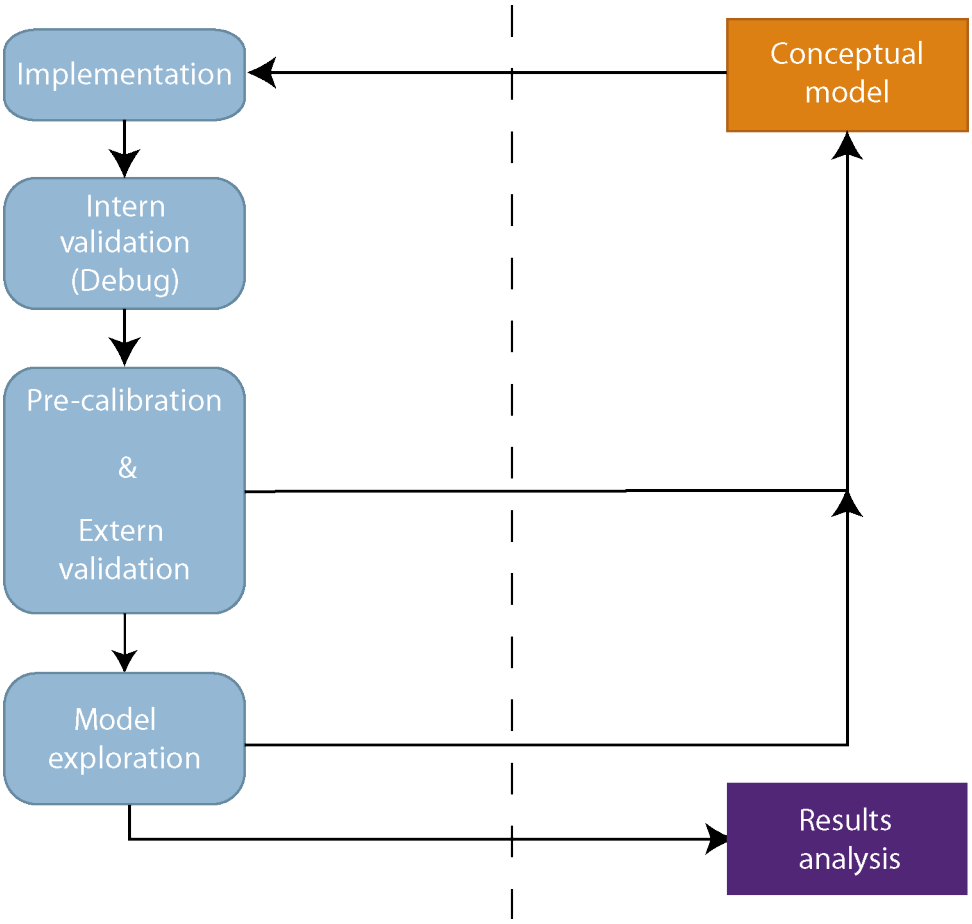

FIGURE 2 – La construction d'un modèle, schéma de S. Rey-Coyrehourcq.

1. *Conception théorique.* Cette étape consiste à définir, en premier lieu, le rôle du modèle. On y décide des faits stylisés à modéliser pour obtenir les résultats souhaités, ou au contraire, on fait une construction abstraite de faits stylisés correspondants à des observations pour comprendre les processus qui résultent de leurs interactions. Elle s'appuie nécessairement sur une importante phase de recherche bibliographique permettant de justifier le choix des faits stylisés ou des dynamiques à représenter.
2. *Développement.* Débutant avec le choix d'une plate-forme de modélisation, le développement consiste en la création du code du modèle, comprenant les règles de comportement des agents ainsi que la mise en place d'indicateurs internes permettant le suivi du déroulement des simulations et de leurs résultats.

3. *Validation interne.* Elle consiste à vérifier que les mécanismes informatiques implémentés correspondent bien aux processus souhaités. C'est une étape nécessaire dès lors qu'on cherche à transcrire des mécanismes parfois complexes qui peuvent mener à des mauvaises interprétations dans l'implémentation. Il existe une branche de l'informatique (la vérification) qui propose des moyens des méthodes formelles pour s'assurer qu'un programme a un comportement consistant avec ses spécifications. Ces méthodes sont lourdes et sont inadaptées à des projets de simulation sociale interdisciplinaires. On y teste aussi la robustesse du modèle et sa constance comportementale en testant des jeux de paramètres éloignés et en en observant les résultats. Dans notre cas, la validation interne s'apparente à du *debugging* effectué par le développeur lui-même.
4. *Exploration des données produites et validation externe.* Sargent définit la validation interne comme « Build the model right », et la validation externe à l'aide de la phrase « Build the right model ». Par la création de rapports (qui peuvent être automatisés), on vérifie que le modèle produit les résultats attendus par la théorie ou conformes aux données observées.
5. *Exploitation du modèle (contrôle et calibrage).* On s'assure dans cette phase de la robustesse du modèle en analysant ses trajectoires de réponses pour de larges gammes de paramétrages. L'exploitation du modèle permet aussi de calibrer précisément celui-ci.

1.3.2 Prendre part à toutes les étapes de la modélisation.

Le stage était organisé (voir figure 4 page 14) de manière à ce que le travail effectué aborde chacune des cinq étapes décrites. Si l'important était que cet axe transversal soit respecté, il n'était ni possible ni souhaitable de concevoir un modèle complet dans le temps du stage. On a ainsi traité les différentes étapes au sein de plusieurs projets, apportant un enrichissement et une expertise dans chacun de ceux-ci. Si l'on ne peut donc pas considérer le stage comme la gestion d'un projet collaboratif complet, il s'inscrit toutefois dans une démarche d'ingénierie caractérisée par l'implication et la prise de décisions à toutes les étapes de la conception d'un projet. Si cela n'est pas nécessaire au modélisateur qui peut choisir de se cantonner à l'une des phases de la réalisation du modèle, le suivi régulier de toutes les phases du modèle assure une vision plus large et permet des réponses plus appropriées dans chaque champs.
Le découpage du mémoire ne suit pas pour autant ces différentes étapes de la réalisation d'un modèle en ce que chacune des quatre principales tâches effectuées, que nous détaillons ici, peut avoir fait appel à plusieurs de ces étapes, ainsi que le résume la figure 3 page suivante.

1. *Enrichissement du modèle SimpopLocal - Ajout d'un module de catastrophes.* Ici, la majorité du travail effectué a concerné les validations internes et externes du module de catastrophe, afin que son impact s'inscrive au sein des mécanismes de SimpopLocal et ne rendent pas ceux-ci inefficients devant des catastrophes qui auraient des effets trop importants. Cette tâche a aussi été l'occasion d'aborder (sans toutefois trop s'y attarder) la phase d'explorations de données.
2. *Participation à la création de SimpopNet.* Si le développement a ici été la plus grande tâche, on a du se baser, après différents essais, et face à l'inadaptation des mécanismes initialement mis en place, sur des mécanismes re-définis, et donc, s'engager dans la phase de conception théorique du modèle, en particulier dans le choix des modes d'évolution du réseau, aboutissant à la création du prototype SimpopNet-Réseaux.
3. *Création d'un outil de production de rapports de simulations - TrajPop.* Cette tâche s'inscrit dans l'exploration des données, par la construction d'un outil de production de rapport de simulations semi-automatisé, conçu autour de SimpopLocal et en particulier

Participation des tâches aux différentes étapes de création d'un modèle

	Conception théorique	Développement	Validation interne	Validation externe	Exploration
SimpopLocal Module catastrophes	✗	✗	✓	✓	✓
SimpopNet Prototype Réseau	✓	✓	✓	✗	✗
TrajPop	✗	✗	✗	✗	✓
Plugin QGIS	✗	✗	✗	✗	✓

FIGURE 3 – Participation des tâches effectuées à chacune des étapes de création d'un modèle.

de l'impact du module de catastrophes mais aussi, dès le départ, pensé pour pouvoir représenter des données observées ou venant d'autres simulations. La réalisation de TrajPop a permis de mieux comprendre et définir les mécanismes de croissance de population dans les simulations, ce qui a été autant utile à la compréhension de SimpopLocal qu'à la mise en place de ces mécanismes dans SimpopNet-Réseaux. Au sein de ce projet logiciel, on retrouve en fait l'ensemble des étapes d'ingénierie décrites ci-dessus pour un modèle.

4. *Création d'un outil de cartographie des résultats de simulations* - *qFlowMapper*. Ici, on s'est concentré sur l'exploration de données, mais à nouveau, le développement d'un plugin QGIS de traitement et de visualisation des flux, en tant que projet logiciel complet, a fait appel aux différentes phase d'ingénierie nécessaires à la conception et à la réalisation d'un outil adapté autant à nos données que ré-employable dans d'autres situations par des utilisateurs qui ne seraient pas nécessairement des modélisateurs.

La structure du mémoire tend donc à regrouper ces quatre tâches sur un aspect plus thématique que chronologique, c'est à dire en considérant d'abord les modèles enrichis et conçus, et ensuite les outils d'analyses créés pour leur exploitation.

N°	Nom de la tâche	Durée
1	**Stage**	**161,88 jours**
2	**Formation**	**15,18 jours**
3	Découverte des SMA - Lecture d'ouvrages et articles	4 jours
4	Découverte de NetLogo - Création de mini-modèles et enrichissement des modèles de réseau	5 jours
5	Création d'un modèle de réseaux - La constitution des réseaux de recherche européens en NBIC - Pour Marie-Noëlle Comin	5 jours
6	Découverte de l'API QGIS via l'utilisation de la console Python	2 jours
7	**Simulations**	**69,82 jours**
8	**SLocal**	**10 jours**
9	Découverte et analyse du fonctionnement de SimpopLocal	2 jours
10	Création d'un module de catastrophes naturelles pour SimpopLocal	5 jours
11	Analyse du comportement des catastrophes dans SimpopLocal - Bouleversements dans la hiérarchie urbaine	3 jours
12	**SNet**	**59,82 jours**
13	Définition des mécanismes de base - Création de l'architecture	4 jours
14	Recherche sur l'implémentation d'un algorithme de plus court chemin - Compréhension et implémentation de l'algorithme de Dijkstra	5 jours
15	Optimisation de la recherche de plus court chemin - Compréhension et implémentation de l'algorithme A*	9 jours
16	Vérification du bon fonctionnement des mécanismes - Analyse des corrélations entre la population des villes et leur nombre de liens	3 jours
17	Redéfinition des mécanismes de SimpopNet_Réseaux - Refonte du modèle	5 jours
18	**Outils**	**71,06 jours**
19	**PluginFlux**	**67,29 jours**
20	Apprentissage de Python	3 jours
21	Apprentissage de l'architecture d'un plugin QGIS - Création du squelette du plugin	2 jours
22	Compréhension du format SVG - Création de fichiers SVG depuis des entitées géographiques dans QGIS	3 jours
23	Travail Python sur la manipulation, l'extraction et la création d'entités géographiques - Création des Points de Contrôles	2 jours
24	Travail sur les courbes de Bézier - Création de courbes de Bezier à partir des Points de Contrôle, affichage dans QGIS, export en SVG	3 jours
25	Création d'un outil de type MapTool dans QGIS - Manipulation manuelle des entités géographiques	4 jours
26	Refactorisation du plugin pour intégration dans un MapTool - Intégration des fonctions en classes	8,5 jours
27	Création de l'outil BezierTool - Manipulation interactive de segments et création/ajustement de courbes de Bezier	3 jours
28	Mise en place d'un outil de regroupement des flux - Intégration de l'algorithme de Force-Directed Edge-Bundling dans QGIS	8 jours
29	Intégration des composantes du Plugin - Finition et Livraison du Plugin - Annonce à la communautée QGIS	2 jours
30	**TrajPop**	**51,65 jours**
31	Découverte, analyse et utilisation de l'outil RankClock du laboratoire CASA	4 jours
32	Création de la méthodologie de traitement de TrajPop - Script R permettant l'analyse des trajectoires de villes	3 jours
33	Simplification et modification du script pour le rendre plus "user-friendly"	4 jours
34	Réglage des sorties - Sorties des graphiques en pdf - Sorties des tableaux enrichis en csv	3 jours
35	**Mémoire**	**19,76 jours**
36	Plannification - Création du plan - Adaptation du planning	2 jours
37	Redaction du mémoire	5 jours
38	Reprise - Corrections et adaptations	3 jours
39	Congés	11 jours

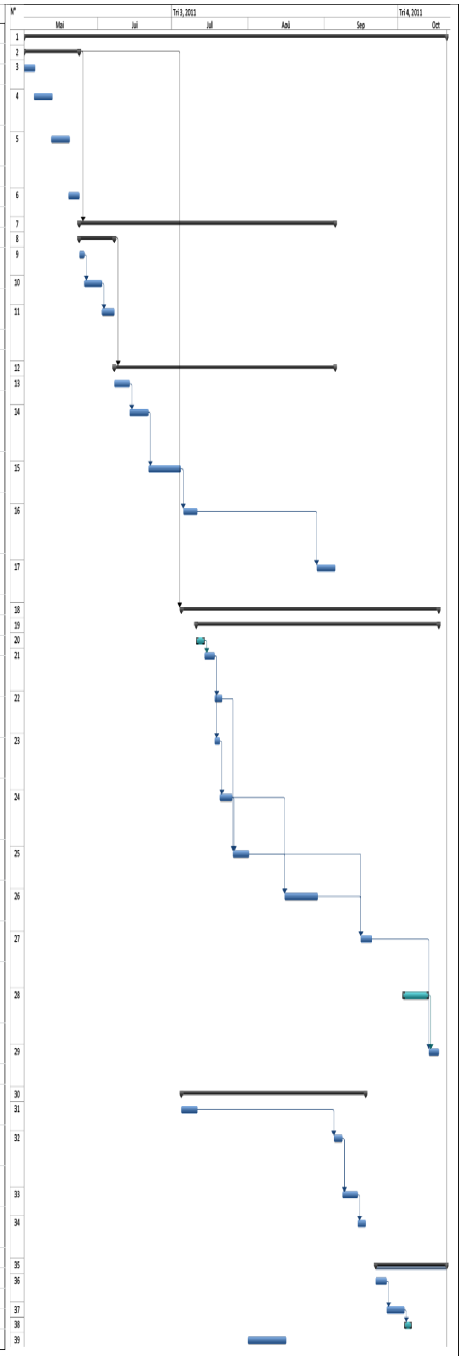

FIGURE 4 – Diagramme de GANTT de l'organisation du stage.

Deuxième partie

Enrichissement et création de modèles.

2.1 Modéliser à l'aide des SMA.

2.1.1 Logiques d'ensemble de la création d'un modèle

L'adoption d'une méthode de développement agile. On l'a vu peu avant (1.3.1 page 11), la création d'un modèle n'est possible que par de fréquents allers-retours entre l'analyse des processus à intégrer, la conception des mécanismes associés, le développement en lui-même et les phases de vérification des sorties du modèle. Dans un contexte de mise au point collégiale, la fréquence et la rapidité de ces itérations devient une composante nécessaire de l'aboutissement du modèle. Dès lors, le travail collectif doit privilégier un échange régulier et facilité par le biais de l'adoption d'une méthode agile, de type *extreme programming*, basée sur la modularité de la conception et du développement et sur des cycles de production courts et fréquents.

Une approche parcimonieuse. Tout au long du stage, on a ainsi privilégié une approche modulaire, choisissant de travailler mécanisme par mécanisme, chacun étant tour à tour pensé, conçu, implémenté, vérifié indépendamment des autres et enfin intégré au modèle. En plus de permettre un travail commun et une compréhension facilitée de chaque mécanisme puisque indépendant, cette méthode a aussi permis de conserver des bases saines à chaque remaniement des mécanismes, les modifications apportées à chaque module n'impliquant pas de remaniement nécessaire du reste du modèle.

La mise en place de cette méthode de gestion de projet a été facilitée par l'utilisation d'un environnement de modélisation maîtrisé par l'ensemble de l'équipe, au langage suffisamment expressif pour permettre une compréhension rapide des mécanismes implémentés, et dispensant dans la plupart des cas de l'écriture en pseudo-code des algorithmes correspondant : NetLogo[TW04] [4].

2.1.2 Présentation de NetLogo

Un outil de modélisation complet et pédagogique. Outil actuellement largement utilisé en simulation à base d'agents dans les sciences humaines et sociales, NetLogo a l'avantage de proposer un environnement intuitif et simplifié pour la création de modèles. De fait, même si la vitesse d'exécution des modèles implémentés est moindre que sur un langage de programmation généraliste et complet (C++, Python, Java) [5], NetLogo a été conçu pour la simulation multi-agents, et toutes les composantes du logiciel sont ainsi orientées dans cet objectif ainsi que dans celui de l'enseignement de la modélisation. L'orientation pédagogique de NetLogo soutenu par une importante bibliothèque de modèles pré-installés, expliqués et commentés pas-à-pas, est au cœur des volontés de l'équipe de développement, composée d'universitaires. S'appuyant sur des primitives explicites et exhaustivement documentées [6] qui rendent le code expressif, NetLogo se démarque particulièrement de ses concurrents [7], grâce à son interface complète permettant notamment un rendu graphique immédiat des simulations en cours. Bien que potentiellement accessoire, la possibilité de visualisation des mécanismes en action est indispensable dans le cadre d'un modèle inter-disciplinaire et spatialisé, mêlant qui-plus-est des chercheurs plus ou moins familiarisés à la modélisation informatique.

4. http ://ccl.northwestern.edu/netlogo/

5. NB : NetLogo est en fait un interpréteur écrit en Java et tournant donc dans la JVM (*Java Virtual Machine*).

6. Le manuel de Netlogo 4.1.3 (version utilisée pour le stage) est disponible à l'url `http://ccl.northwestern.edu/netlogo/docs/`.

7. Repast, AnyLogic ou Swarm par exemple

Une plate-forme extensible par ses utilisateurs. Notons enfin que NetLogo propose une API complète, permettant à sa communauté d'utilisateurs d'enrichir la base du logiciel par le biais d'extensions écrites en Java, lesquelles permettent par exemple d'importer, de manipuler et d'exporter des données vectorielles géographiques [8], ce qui rend alors possible la réalisation de simulations sur des réseaux réels ainsi que l'analyse des résultats grâce à des SIG (voir la figure 5 page 17).

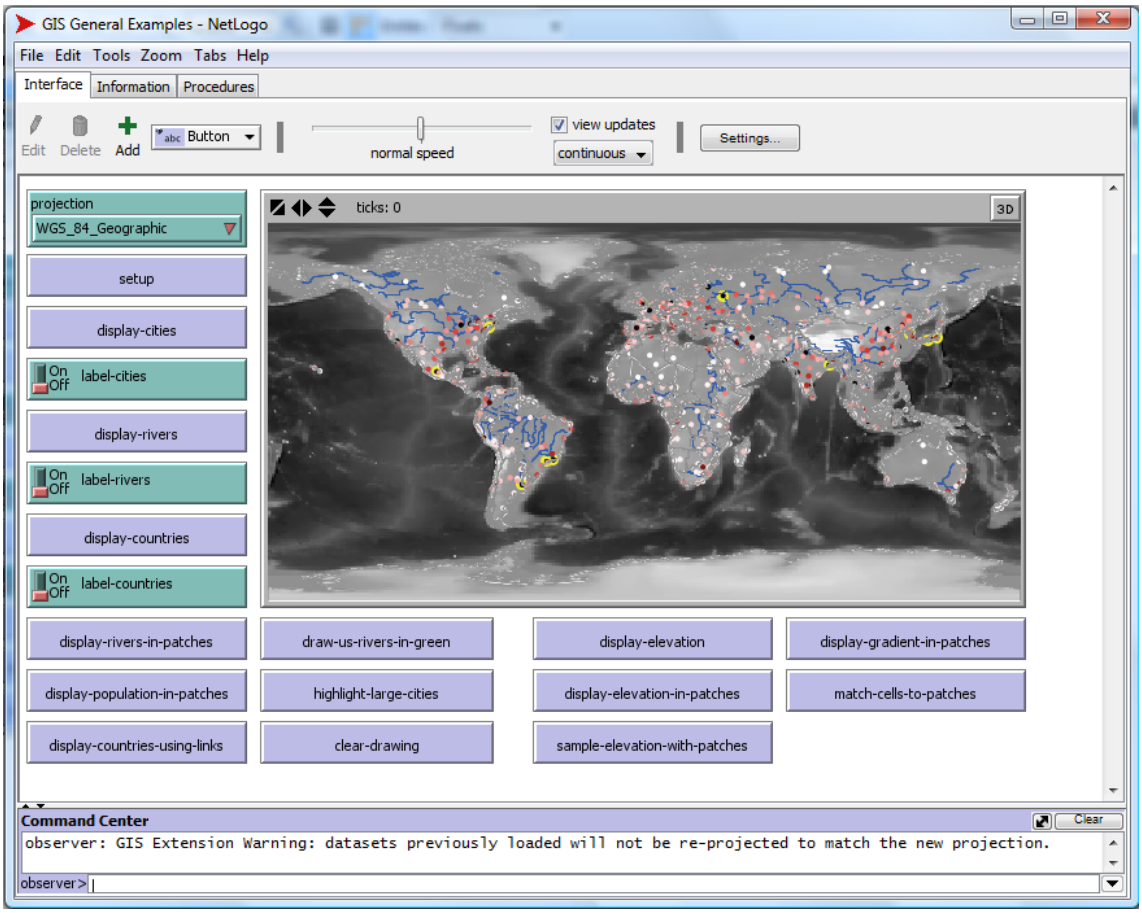

FIGURE 5 – L'un des modèles de la bibliothèque de NetLogo, utilisant l'extension SIG.

2.1.3 Un exemple pratique et formateur

Modéliser les processus de constitution de réseaux de collaboration chez les chercheurs européens. Pour se familiariser aux logiques de la simulation à base d'agents ainsi qu'à NetLogo il a été décidé de créer un modèle répondant à une demande d'une chercheuse de l'ISC-PIF, Marie-Noëlle Comin. Son travail porte sur la compréhension de l'émergence de réseaux de collaborations de laboratoires de recherches européens, dans le secteur des NBIC (*Nanotechnology, Biotechnology, Information technology and Cognitive science*) dans le cadre de la constitution d'équipes pour répondre aux appels à projets de recherche des institutions européennes. Cherchant à modéliser ces processus complexes, le modèle proposé présentait un large panel de possibilités de constitution des équipes (voir figure 6 page suivante), toutes basées sur les mécanismes de base de l'attachement préférentiel [AB02]. Les principaux critères d'association implémentés et déclinés sont :

8. Via l'extension GIS, décrite ici : `http://ccl.northwestern.edu/netlogo/docs/gis.html`

1. la préférence absolue : un laboratoire choisissant de collaborer avec celui qui a la meilleure réputation ;
2. la préférence au passé, prenant principalement appui sur le succès ou non des collaborations précédentes avec chaque laboratoire ;
3. l'appel aux collaborateurs présents ou passés, pour les aider à choisir, que ce soit en choisissant un ancien collaborateur d'un membre de l'équipe ou encore en demandant à un équipier de choisir lui-même le laboratoire à intégrer dans l'équipe.

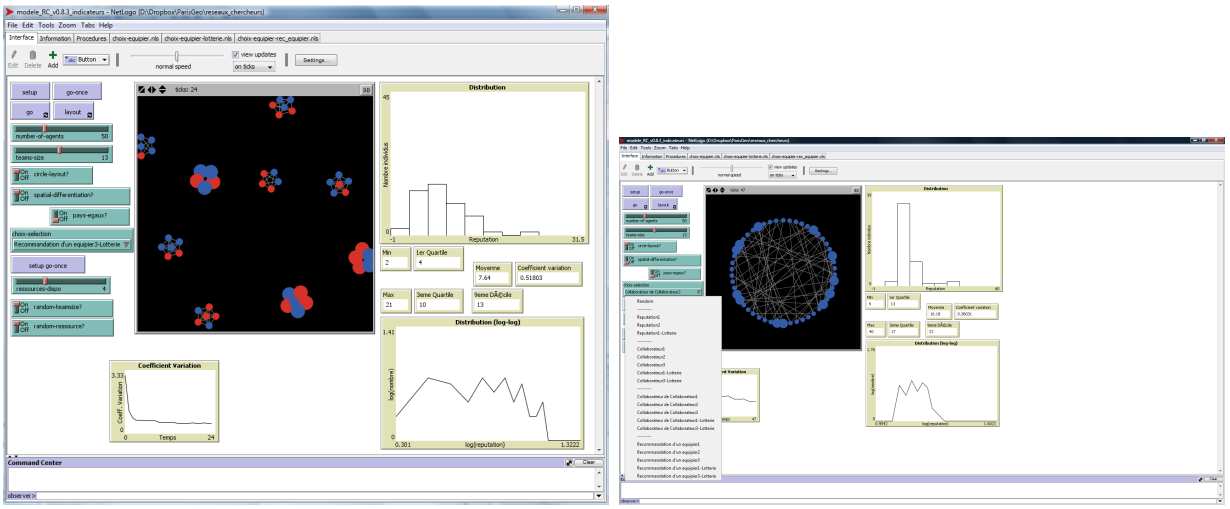

FIGURE 6 – Interface et paramétrage du modèle de collaboration de laboratoires européens

En dehors de l'intérêt thématique, la conception de ce modèle a permis de tester le processus itératif de travail plébiscité par les membres de l'équipe, caractérisé par une mise en contexte et en production rapide. Il a également joué le rôle de tutoriel en permettant de se former aux logiques modélisation de réseaux dynamiques, avec des graphes, sous NetLogo. Si le travail sur ce modèle n'a jusqu'ici pas été mené à une étude de ses dynamiques, il a ouvert la voie à la compréhension et à la mise en place du réseau dans SimpopNet. Le modèle est consultable et utilisable en ligne à l'adresse suivante : `http://robin.cura.free.fr/netlogo/`

2.2 L'enrichissement d'un modèle : SimpopLocal.

Le travail mené sur SimpopLocal a consisté en la réalisation d'un mécanisme simulant les effets de catastrophes naturelles venant perturber la croissance d'un système de peuplements. Bien que minime en considération du modèle pré-existant, a mise en place de ce module a permis une découverte par l'expérience de SimpopLocal et de ses mécanismes principaux, découverte nécessaire pour la suite, un certain nombre de mécanismes du modèle SimpopNet en étant tirés.

2.2.1 Mécanismes de SimpopLocal.

En se basant sur une logique modulaire, le développement de SimpopLocal a facilité l'intégration d'un ensemble de scénarios, que ce soit sur la forme du réseau initial du système de peuplements, des mécanismes d'innovation ou encore des modes de diffusion de celles-ci. Si

tous les choix effectués ont permis de tester différentes hypothèses, nous nous contenterons ici de décrire le jeu de paramètres et de mécanismes adopté pour l'implémentation et l'analyse du module de catastrophes.

Création des réseaux. Le réseau de départ est fixe, mais résulte d'une expérimentation où celui-ci présentait les caractéristiques souhaitées. On crée de manière aléatoire cents foyers de peuplement, dont la population de départ suit une distribution log-normale, en l'occurrence une répartition aléatoire de moyenne 80 et d'écart-type 20. Chaque foyer de peuplement est ensuite associé à une classe de taille, et on les réparti spatialement en fonction de leurs classes, suivant la théorie des lieux centraux [Chr33], permettant alors la création de liens entre les foyers de peuplement, liens qui constitueront les arrêtes du réseau.

Croissance démographique de la population et limitation par les ressources. A chaque itération du modèle, les foyers de peuplement voient leur population croître selon un taux fixe (2%), pondéré par la quantité de ressources disponibles, suivant cette équation, tirée du modèle de Verhulst [Ver38] :

$$Pop(t+1) = Pop(t) + 0.02 \cdot Pop(t) \cdot \left(1 - \frac{Pop(t)}{Ressources}\right)$$

Cette ressource, élément central du modèle, est initialement attribuée de la même manière que la population, et tend à mimer la disponibilité des ressources naturelles d'un site, laissant croître la population jusqu'à ce que les ressources, devenues limitées, contraignent l'évolution du foyer de peuplement et mènent ainsi à un équilibre entre ressources et population, à un seuil non franchissable sans autre apport.

Création d'innovations. Un des mécanismes principaux du modèle est l'apparition d'innovations, figurant les découvertes et améliorations amenées par la création d'outils, de techniques et d'organisations du travail permettant une meilleure exploitation des ressources naturelles. Une innovation apparaît de manière aléatoire, sa probabilité étant plus que proportionnelle à la taille du peuplement la voyant naître. L'innovation permet de repousser les limitations environnementales à la croissance du foyer de peuplement, en augmentant les ressources disponibles, ce qui engendrera une croissance démographique. Dès lors, une boucle de rétroaction positive se met en place, l'apparition d'innovations faisant augmenter la population, ce qui donnera au foyer de peuplement une plus grande probabilité d'engendrer de nouvelles innovations (voir [Art09]).

Diffusion d'innovations. La diffusion des innovations se fait en adaptant un modèle classique en géographie, le modèle de gravité. Ainsi, le modèle cherche à privilégier les interactions et donc les diffusions d'innovations entre les foyers les plus peuplés et les plus proches. En pratique, à chaque tour, les innovations tentent de se diffuser vers les peuplements voisins qui ne les possèdent pas déjà, et ont une probabilité d'y parvenir proportionnelle à leur potentiel de propagation.

Nous posons le postulat que la copie d'innovation est plus facile que la création de nouvelle innovation ([Pen10]). Les facteurs paramétrant le mécanismes stochastiques de la copie sont donc plus forts que ceux paramétrant la création d'innovation. Comme on a vu que l'acquisition d'innovations conditionnait le rang du foyer de peuplement dans son système, le mécanisme de copie, en partie basé sur le modèle gravitaire, joue donc ici un rôle majeur.

C'est sur cette base qu'a été mis en place le module de catastrophes. Au delà d'un simple ajout, il fallait tenir compte du fonctionnement global pour que l'intervention d'une catastrophe impacte en profondeur les foyers de peuplement modélisés.

2.2.2 Mise en place du module de catastrophes.

La logique des catastrophes

Les catastrophes. Le mot catastrophe recouvre un large panel d'événements perturbants un système. Si l'on peut penser de prime abord aux catastrophes naturelles (séismes, éruptions par exemple) ou climatiques (comme les périodes de sécheresse), on doit aussi pouvoir modéliser les catastrophes politiques (invasions barbares, guerres entre autre) ou encore les catastrophes humaines (à l'instar des épidémies et famines). L'enjeu du module était de pouvoir s'adapter, via modification des paramètres, à tous ces types de catastrophes, qui ne ciblent pas les mêmes foyers de peuplement (selon topologie, état politique ou administratif, répartition sociale) et ne les affectent pas de la même manière. L'implémentation ici présentée se positionne plutôt dans le cadre de catastrophes climatiques, mais la disposition du code rend possible et facile les adaptations aux autres contextes à modéliser.

Aléa et calibration. Comme l'ensemble des modèles découlant de Simpop2, SimpopLocal est un modèle à forte composante stochastique, que ce soit par l'appel explicite à des fonctions d'aléa ou simplement par l'ordre de tirage des agents dans la réalisation de chaque tâche. Si l'aléa est central dans la modélisation, il doit cependant être circonscrit dans certaines limites de fonctionnement attendu, d'une part pour que les résultats laissent apparaître une logique d'ensemble d'une simulation à l'autre, et d'autre part pour pondérer le poids de l'aléa de chaque élément par rapport aux autres. Dans le module de catastrophes de SimpopLocal, il était particulièrement important de borner le phénomène : on ne peut ainsi comparer une simulation ayant vu 5 catastrophes en 2000 ans et une autre en voyant une survenir près de 3 fois par siècle. A l'aide d'un générateur d'aléa basé sur la loi de Poisson, on a ainsi pu définir une périodicité moyenne d'apparition de catastrophes. Cela a notamment permis de calibrer plus aisément la force d'impact des catastrophes afin que leur action au cours des différentes simulations produise des effets similaires.

Mécanismes. L'ajout du module de catastrophes devait reposer sur le modèle SimpopLocal existant, et dès lors, faire en sorte que l'impact soit endogène à la simulation. Pour ce faire, les mécanismes suivants ont été adoptés :

1. *Apparition.* Le mécanisme d'apparition des catastrophes est stochastique et pondéré par une loi de Poisson. Ainsi, à chaque itération de la simulation, une catastrophe peut survenir selon une probabilité fixée. L'appel aux probabilités peut produire des phénomènes inattendus, par exemple le fait que deux catastrophes surviennent quasi simultanément, puis qu'il n'en apparaisse plus de nouvelles avant longtemps. Néanmoins, dans la majorité des cas, une catastrophe surviendra toutes les n années, où n est égal à l'inverse de la probabilité d'apparition. Ainsi, si la probabilité de catastrophe est fixée à 2%, on aura en moyenne une catastrophe toutes les 50 itérations.
2. *Impact.* Quand une catastrophe survient, on tire de manière aléatoire, suivant la loi uniforme (bornée par un maximum) son rayon d'action. En prenant l'exemple des catastrophes naturelles, il fallait ainsi qu'un événement de ce type n'impacte pas nécessairement qu'un foyer de peuplement, mais atteigne tous ceux présents dans un rayon

aléatoire autour de l'épicentre du désastre. La localisation de cet épicentre est aléatoirement fixée sur l'un des foyers de peuplement, et cet aléa aura donc un rôle majeur, selon la centralité géographique et l'importance de celui touché.

3. *Effets.* Il faut tout d'abord noter que la force de la catastrophe est variable selon les foyers de peuplement impactées. Ainsi, on a choisi de ne pas mettre en place de logique de distance où l'on aurait pu définir que plus les foyers de peuplements touchées étaient proches de l'épicentre, plus les effets y étaient importants. Nous éloigner de ce modèle permet d'englober un plus large panel de catastrophes, c'est-à-dire de modéliser des événements qui ne sont pas nécessairement naturels et "cataclysmiques". L'autre choix majeur effectué ici est qu'une catastrophe agit sur la population d'un foyer de peuplement tout autant que sur ses ressources disponibles. Dès lors, un foyer de peuplement atteint ne subira pas qu'un contre-temps dans sa croissance, mais sera véritablement affecté dans son développement, via la remise à un certain pourcentage des valeurs pré-catastrophes de population et de ressources disponibles du foyer de peuplement.

Implémentation

Paramétrage. Comme dans l'ensemble de SimpopLocal, on a tenu d'une part à la modularité, laissant la possibilité de ne pas utiliser le module de catastrophes, mais aussi à la liberté de paramétrage pour l'utilisateur, laissant aussi la possibilité de court-circuiter le fonctionnement standard des catastrophes pour imposer manuellement l'apparition d'une catastrophe au moment voulu par le simulateur, possibilité très utile pour comparer différents scénarios, tels que l'action d'une catastrophe sur le système de ville selon la date d'apparition.

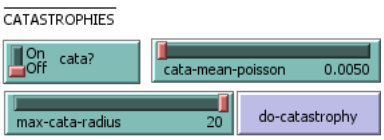

FIGURE 7 – Paramétrage des catastrophes

Les graines. On a vu que NetLogo faisait un usage intensif de l'aléatoire, mais pour autant, cet aléa, pour des raisons évidentes de reproductibilité des expériences, doit être contrôlable. Cela se fait dans cet environnement par l'appel à une *graine aléatoire* qui définit l'ordre de tirage de l'aléa à chaque appel. Nativement, cette graine est basée sur l'état du système et est unique, deux simulations n'ayant qu'une probabilité infinitésimale de tirer la même graine. Afin de pouvoir reproduire et comparer des simulations, NetLogo permet de fixer la graine qui sera utilisée. Dès lors, deux simulations utilisant les mêmes graines se dérouleront exactement de la même manière et fourniront les mêmes résultats. Pour autant, avec ce système, toute chose égale par ailleurs, on ne peut comparer une simulation avec catastrophes, qui fera un usage supplémentaire de l'aléa, avec une simulation standard. En effet, la catastrophe étant basée sur cette même graine, elle décalera systématiquement la génération de nombres aléatoires et n'aboutira pas aux mêmes résultats en dehors des catastrophes. On a donc choisi d'utiliser une primitive créée à cet effet (*with-local-randomness*), dont le rôle est de circonscrire certains aléas pour ne pas influer sur l'utilisation de la graine pour les autres. Dès lors, on peut comparer plusieurs simulations utilisant un même graine, selon qu'elles utilisent ou non le mécanisme de catastrophes ainsi que plusieurs simulations identiques voyant les catastrophes survenir à des périodes différentes de la simulation.

Développement. Le code en lui même a été assez rapide à écrire, en ce qu'il est court mais ce n'est pas tant le développement en lui-même qui occupe l'essentiel du temps du modélisateur, mais les aller-retour entre la conceptualisation des faits à représenter, la ré-écriture du code, et la vérification des résultats escomptés, comme on l'a vu précédemment (cf. 1.3.1 page 11). De plus, comme indiqué, il a fallu concevoir cette implémentation comme assez souple pour être adaptable à d'autres types de catastrophes.

2.2.3 Résultat des catastrophes.

Le module étant assez récent, et surtout, sa calibration n'étant pas encore achevée (cf. SimProcess 1.2.2 page 9), l'étude de l'influence des catastrophes sur le modèle SimpopLocal n'a pas encore pu être menée. Pour autant, les sorties de simulations ont tout de même permis d'émettre quelques hypothèses sur leur effet. En premier lieu, SimpopLocal visant en partie à montrer l'émergence d'une hiérarchie urbaine et sa stabilité, on a pu en faire émerger quelques logiques. Les documents suivants (figure 8 page suivante) illustrent l'impact de catastrophes sur le système de peuplements.

Dans l'exemple ici présenté, on compare deux simulations faisant appel à la même graine, avec et sans catastrophes. Les premiers graphiques montrent l'évolution de la population totale du système, les deux traits sur la seconde courbe correspondants aux dates des catastrophes. On y constate que les catastrophes ne font que ralentir temporairement la croissance du système, sans toutefois porter atteinte à sa dynamique. Par exemple, dans la simulation sans catastrophes, le système atteint 50000 habitants en 1450 tours alors que dans l'autre cas, il faut attendre 300 tours de plus pour arriver à la même somme de population. La seconde série de graphiques, de type *RankClocks*[9], montre bien l'impact sur la hiérarchie de ville des catastrophes : les modifications sont profondes, en ce que les foyers de peuplements touchés, même ceux en tête de hiérarchie, ne parviennent pas à se relever dans les délais de la simulation. La hiérarchie est fortement restructurée, les foyers de peuplement touchés, ici principalement de taille moyenne, se retrouvant en majorité dans le bas de la hiérarchie, et y demeurant de manière stable. Le dernier graphique, dont l'on expliquera la construction plus loin (cf. 3.2 page 33), montre la trajectoire des différentes classes de villes résultant d'une Classification Ascendante Hiérarchique. Si, dans la simulation témoin, malgré quelques légers croisements, on retrouve bien le même ordre en début qu'en fin de simulation, c'est-à-dire que la structure de la hiérarchie reste stable, le graphique permet de réaliser l'impact des catastrophes : les villes les plus touchées par la première catastrophe sont quasiment détruites et ne montrent plus de croissance (classe 1), tandis que celles les plus impactées par la seconde catastrophe (classe 3) parviennent à se relever, sans doute grâce aux innovations acquises, sans toutefois réussir à revenir au niveau des villes non affectées (classes 2 et 4) ou même des moins impactées (classe 5).

2.3 SimpopNet-Réseaux.

2.3.1 Introduction.

SimpopNet, un modèle subdivisé Le développement du module de catastrophes de SimpopLocal avait pour principal objectif d'en assimiler les mécanismes afin de pouvoir en réintroduire

9. Ces graphiques montrent l'évolution de la hiérarchie autour d'un axe circulaire, les rangs supérieurs (foyers de peuplements les plus peuplés) étant au centre, et chaque rayon constituant un pas de temps. Pour plus d'explication, voir 3.1 page 32.

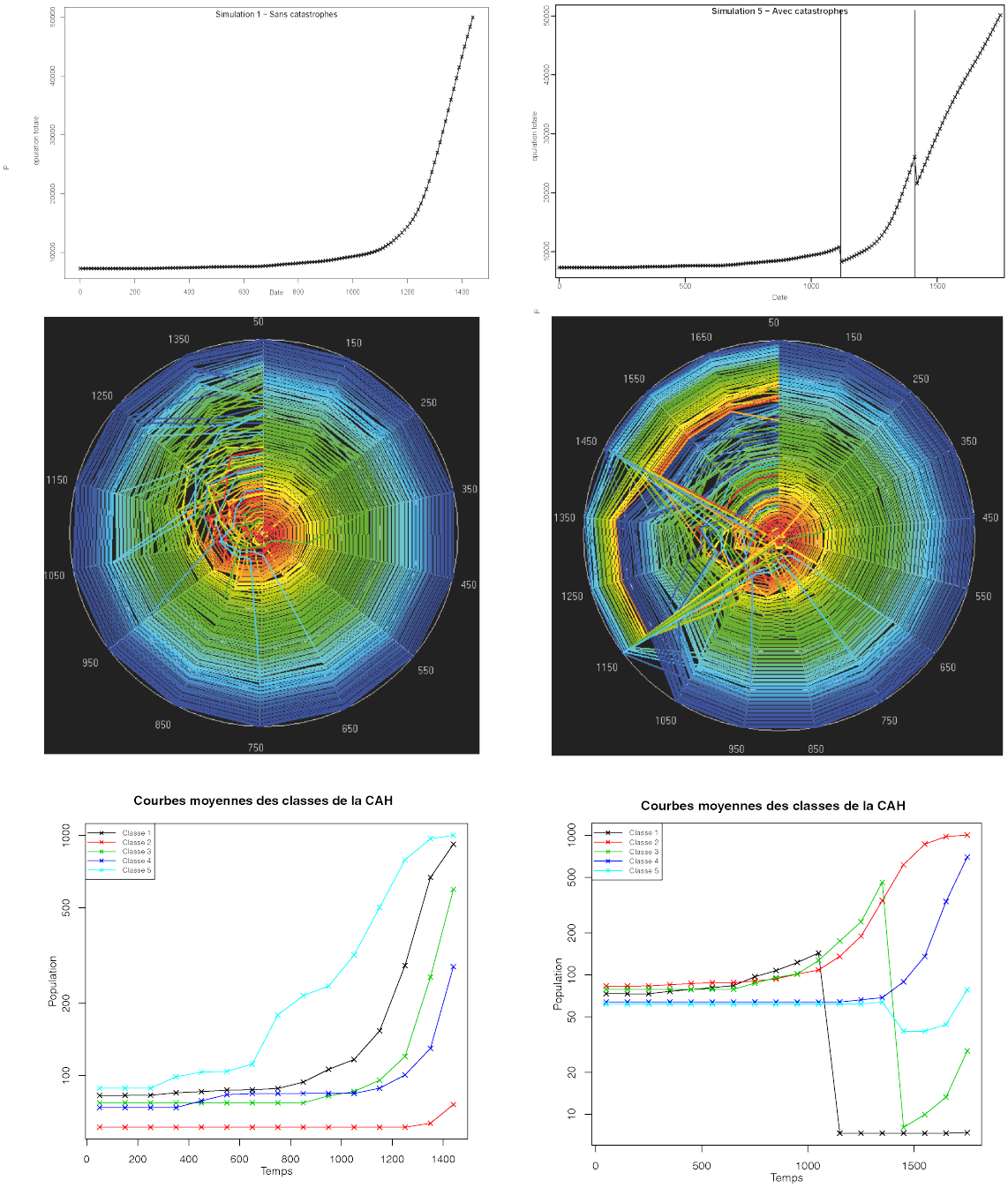

FIGURE 8 – Résultats de deux simulations de SimpopLocal : sans et avec catastrophes

certaines dans SimpopNet. Si ce modèle devait à l'origine être conçu dans son ensemble, la multitude d'interrogations et d'hypothèses à tester a abouti à la division en 3 modules, chacun devant donner lieu à un prototype. Dans chaque prototype, les mécanismes détaillés dans les autres prototypes sont traités de façons simplifiées pour ce concentrer sur l'étude du mécanisme phare du prototype. Ces trois modules traitent séparément les trois mécanismes de SimpopNet

1. SimpopNet-Croissance
   Ce prototype doit aboutir à trouver un type de croissance efficace dans la prise en compte des effets de contexte (situation, concurrence du voisinage et effets des acquisitions des innovations) pour la croissance de chacune des villes. Afin de coller au contexte du développement des villes dans un système urbain, les mécanismes choisis doivent aboutir à des boucles de rétroactions positives ou négatives dans la croissance urbaine.

(mécanisme détaillé plus loin, 2.3.2 page 24). Dans les autres modules, le mécanisme de croissance est traité de façon simplifié grâce à l'implémentation du modèle de Gibrat (mécanisme détaillé plus loin, cf. 2.3.2 page 24).

2. SimpopNet-Innovation
   Tiré du fonctionnement de l'apparition et de la diffusion des innovations dans SimpopLocal, ce module doit permettre d'agir sur la croissance, de concert avec le mécanisme choisi. Ce faisant, il influera sur le réseau créé et donc à nouveau sur l'évolution du système urbain. Il se peut que ce module ne donne pas lieu à prototypage, mais soit directement importé depuis SimpopLocal, tout en subissant les modifications nécessaires à son intégration et à son ajustement dans SimpopNet.
3. SimpopNet-Réseaux
   Constituant une importante composante du futur SimpopNet, le prototype doit viser à la mise en place d'un réseau dynamique. Servant de support d'échange entre les villes, le réseau doit pouvoir évoluer par la création de nouveaux liens, voire par la suppression de ceux devenus inutiles.

C'est sur ce dernier module que s'est axé notre travail, en ce qu'il apportait une logique originale (c'est-à-dire sans aucun lien avec SimpopLocal) au modèle. De plus, les effets d'interaction spatiales amenées relevaient tout autant de l'analyse des réseaux que de l'analyse spatiale, et constituaient donc une passerelle intéressante entre ces deux univers.

2.3.2 Mécanismes et hypothèses de départ.

Si le module SimpopNet-Réseaux n'a pour but que de travailler sur la création et les dynamiques du réseau de villes dans SimpopNet, il fallait, pour qu'un prototype soit exploitable, y greffer un système simple de gestion de la population, dans sa distribution initiale et dans sa croissance, ainsi qu'une base de hiérarchisation du système de villes. On présentera donc ici brièvement les mécanismes employés pour rendre le prototype viable.

La création du réseau initial. À l'initialisation du modèle, on crée un certain nombre de villes, 100 dans le paramétrage utilisé, que l'on place toutes au même endroit. Afin d'obtenir une hiérarchie urbaine marquée dès le départ, on dote ces villes en population suivant une distribution log-normale de large étendue. À partir de cette distribution, on peut classer les villes en 3 catégories de taille, de la manière suivante. On trie les villes par ordre décroissant de population, et on découpe alors notre distribution en 3 catégories de population à peu près égale (chaque classe possédant donc environ 1/3 de la population, et concernant d'autant plus de villes que la classe est faible).
On peut alors répartir spatialement les villes en fonction de leur catégorie, selon le schéma d'organisation hiérarchisé d'un réseau de ville défini par Christaller [Chr33], c'est-à-dire en laissant un espace minimal d'autant plus faible, entre les villes d'un même catégorie, que celle-ci l'est. On va ensuite relier, dans des rayons décroissants selon la catégorie, les villes de chaque catégorie entre elles, puis les villes de catégories inférieures aux villes de catégories supérieures.

Croissance de population : la loi de Gibrat. Le mécanisme de croissance est plus simple et repose sur la théorie développée par Robert Gibrat [Gib31] qu'on aboutit à une distribution de population log-normale si l'on respecte 3 conditions :

1. A chaque intervalle de temps, les villes gagnent une quantité de population proportionnelle à leur population. Le taux de croissance résultant est en moyenne le même

pour l'ensemble des villes considérées avec des fluctuations d'un écart-type proche de la moyenne.

2. Les taux de variation de chaque ville ne sont en aucun cas corrélés à la population de ces villes. Ainsi, le taux de croissance d'une ville importante ne sera pas nécessairement supérieur à celui d'une ville moindre.
3. Les taux de variation d'une ville n'est pas corrélé aux taux de variations des périodes précédentes. Si une ville a montré une forte croissance dans l'intervalle précédent, le taux de croissance à suivre ne sera pas nécessairement important.

Dans la mise en œuvre du mécanisme de croissance, on a donc veillé à ce que le taux de croissance, dont l'espérance et l'écart-type sont paramétrables, soit tiré de manière aléatoire à chaque tour et pour chaque ville.

Les interactions. Pour que notre réseau soit dynamique, il fallait définir des règles de base pour la création et éventuellement la suppression de liens. Ce dernier mécanisme n'a pas été mis en place en ce qu'il convenait de d'abord tester l'enrichissement du réseau et que le mécanisme de suppression découle très fortement de celui de création, ne demandant quasiment qu'une inversion de celui-ci. Pour les créations de lien, on a choisi de se placer dans le modèle gravitaire, modélisant le potentiel d'interaction entre deux villes par la division du produit de leurs populations par leur distance.
Afin de prendre en compte de manière plus importante les inégalités de liaisons des villes selon leur importance démographique, on a décidé d'adapter ce modèle en passant d'une logique de distance à une logique de temps. De fait, en considérant qu'on reliait, à distance équivalente, deux villes importantes plus rapidement que deux villes mineures, il a été décidé d'apporter une pondération à l'ensemble des distances considérées, pondération prenant en compte la vitesse et amenant donc à des comparaisons de durée.
Les vitesses ayant un rôle discriminant mais devant garder une certaine cohérence entre elles, il a été choisi de les faire correspondre aux catégories de villes, un lien reliant deux villes ayant la même vitesse qu'un autre reliant deux villes de mêmes catégories. Notons que cette vitesse est intrinsèque au lien et au moment de sa création (initiale ou au cours de la simulation) : sauf ré-actualisation de sa vitesse, un lien conservera la vitesse attribuée à la création, même si les villes reliées ont entre-temps changé de catégorie.
Enfin, pour évaluer l'utilité de créer un nouveau lien entre deux villes, on a choisi de se baser sur la différence entre la distance-temps réseau et la distance-temps euclidienne. En divisant le premier par le second, on obtient un indice souvent utilisé en hydrologie et en analyse des réseaux routiers : un indice de sinuosité. On peut dès lors, en fixant un seuil, juger que le temps de parcours entre deux villes est trop important (via le réseau) comparé à ce qu'il devrait théoriquement être (en distance-temps euclidienne), et dans ce cas, décider de la création d'un nouveau lien entre ces deux villes mal connectées.

Tirages aléatoires pondérés. Si NetLogo propose nativement plusieurs primitives permettant un tirage aléatoire, il n'est pas possible de pondérer les probabilités pour une entité d'être choisi. Nous avions besoin d'une pondération, car la probabilité de tirer une ville de grande taille devait être nettement plus forte que la probabilité de tirage d'une petite ville. Dans les exemples de code fournis avec NetLogo, un algorithme est cependant proposé, et c'est celui-ci que nous employons à chaque fois que le besoin d'un tirage aléatoire pondéré émerge. Cet algorithme, nommé *Lottery* peut être décrit ainsi :

Soit n le nombre d'entités

Soit E_i l'entité i avec $\{i \in \Re\,;\,[1\,;n]\}$
Soit $P(E_i)$ le poids de l'entité i
$SommePoids \leftarrow \sum_{i=1}^{n} P(E_i)$
$NombreTiré \leftarrow random(SommePoids)$
Pour chaque E_i, avec i dans un ordre aléatoire :
    Si $P(E_i) > NombreTiré$:
        Retourner E_i
    Sinon :
        $NombreTiré \leftarrow NombreTiré$ - $P(E_i)$

2.3.3 Création de SimpopNet-Réseaux.

Création du réseau de villes.

On a vu dans la sous-partie précédente que la création initiale du système de villes et du réseau répondaient à des mécanismes précis et documentés. A partir de là, on peut donc maintenant présenter les modalités de leurs implémentations dans NetLogo, pour créer le module SimpopNet-Réseaux.

Répartition des villes. On a vu que les villes étaient originellement créées en un seul endroit, puis dispersés en fonction de leur catégorie. Pour ce faire, on fait appel aux agents de type *patches* de NetLogo qui représentent les pixels sur lesquels sont créées les villes. On commence par demander aux villes de première catégorie, dans un ordre quelconque, de se déplacer vers un *patch* quelconque et de marquer celui-ci, interdisant qu'une autre ville viennent s'y établir. Ce faisant, on marque aussi les *patches* environnants dans un rayon paramétrable afin qu'une autre ville de catégorie 1 ne s'y installe pas non plus. Une fois toutes les villes de catégorie 1 placées, on répète le même processus pour la catégorie 2, avec un rayon différent cette-fois ci, correspondant au rayon choisi pour la classe 2. Cela terminé, on peut déplacer les villes de catégorie 3, toujours selon le même processus, qui viendront donc, en raison de leur rayon plus faible, se situer entre les villes des catégories supérieures.

Création des liens. On peut alors relier les villes entre elles, selon leur répartition spatiale et leur catégorie. De fait, même si les rayons maximum de liaison sont au choix du simulateur et peuvent donc différer des rayons utilisés pour la localisation de villes, le fait de connecter les villes ensemble selon leur proximité spatiale et de catégorie complète la répartition des villes afin de donner un réseau initial très hiérarchisé dans sa structure. Une fois les liens créés, ils se voient attribuer leur vitesse, qui est en fait un pondérateur inverse de la vitesse. Ainsi, la vitesse est égale à la somme des catégories des deux villes reliées. Un lien entre deux villes de première catégorie aura donc une vitesse de 2 $(1 + 1)$, tandis qu'un lien entre deux des plus petites villes vaudra 6 $(3 + 3)$.

Vérification du fonctionnement du réseau. Après cette phase d'initialisation du réseau, il est nécessaire de tester celui-ci. En effet, pour que les interactions puissent se faire potentiellement entre chaque ville, il faut que celles-ci soient toutes reliées, c'est-à-dire que le réseau ne présente qu'une unique composante et pas de composantes connexes. Pour cela, la méthode la plus simple est de partir d'une ville et de vérifier que toutes les villes y sont directement ou indirectement connectées. NetLogo propose jusque dans ses versions actuelles des primitives de calcul de distance réseau simple, c'est-à-dire non pondérées, et pour des questions de rapidité d'exécution, elles ont été préférées aux algorithmes plus lourds mis en place dans le module.

On parcourt donc l'ensemble des plus courts chemins possibles depuis une ville aléatoire vers les autres, et si l'un de ces cheminements n'est pas possible, on recommence alors toute la création du réseau, jusqu'à ce qu'un réseau entièrement connecté apparaisse.

Le choix d'un couple de villes.

Le mécanisme de création de lien examine la pertinence de créer un lien entre deux villes données, lesquelles sont tirées selon l'algorithme de *Lottery* présenté auparavant. Bien que ce mécanisme soit utilisé pour choisir chacune des villes, le tirage aléatoire pondéré ne se fait pas sur les mêmes poids.

Tirage aléatoire. La première étape est le choix de la ville-hôte : pour cela, on mène un tirage sur l'ensemble des villes, utilisant la population comme pondération — plus une ville est peuplée, plus elle aura de chance d'être sélectionnée.

Les calculs des potentiels. Depuis cette ville-hôte, on va calculer le potentiel d'interaction de chacune des autres villes vis-à-vis de celle-ci. Ce potentiel est basé sur le modèle gravitaire, ajusté à notre contexte de distance-temps. Il se présente ainsi :

$$I_{ij} = \frac{P_i \cdot P_j}{(T_{ij})^{\alpha}}$$

Avec I_{ij} le potentiel d'interaction entre la ville i et la ville j, P_i et P_j les populations de i et j, T_{ij} la distance temps euclidienne de i à j, et enfin, α qui est une constante à paramétrer.

Ceci fait, on pourra effectuer un nouveau tirage au sort basé sur ces potentiels, et la possibilité de création d'un lien entre la ville-hôte et la ville choisie sera étudiée.

Dynamisme du réseau.

Une fois que le lien potentiel est tiré, plusieurs hypothèses se présentent :

1. **Le lien potentiel existe déjà dans le réseau.**
   Si on pensait à l'origine exclure les liens existants du tirage au sort, on a finalement choisi de les y conserver pour que ce cas de figure ait une conséquence. Ainsi, si un lien existant est tiré, la conséquence est que sa vitesse est actualisée. On a vu auparavant que la vitesse d'un lien ne changeait pas après sa création, et pouvait donc être fausse si les villes reliées changeaient de catégorie. Ce mécanisme minime permet de tenir actualisés les vitesses des villes, ce qui a déjà un rôle important dans le modèle.
2. **Les deux villes tirées n'ont pas de lien direct.** C'est dans ce cas que l'on examine l'intérêt de construire un nouveau lien dans le réseau. On a vu que ce choix dépendait de la valeur de l'Indice de Sinuosité calculé entre les deux villes : Si l'indice est supérieur au seuil paramétré, le lien est créé, sinon, on passe à l'itération suivante.

Nous allons maintenant commenter cet indice, qui constitue le cœur du prototype SimpopNet-Réseaux.

Distances euclidiennes et distances réseau.

On a vu que tout le mécanisme de création de nouveaux liens était basé sur la comparaison entre distance-temps euclidienne et distance-temps réseau. Le passage d'une distance à une distance-temps se fait par l'action de la vitesse, mais restait encore à définir l'impact de la vitesse de circulation sur les distances étudiées.

Le problème des distances réseaux. Ainsi, l'une des problématiques soulevées était de donner une place à la vitesse qui ne soit pas déséquilibrée, et donc de trouver de quelle manière la vitesse influencerait la distance-temps. Pour ce faire, on a adopté un mécanisme permettant d'utiliser des informations qualitatives, les numéros de catégorie, en tant que facteur quantitatif. Partant du postulat que la vitesse de circulation sur un lien reliant des villes de grande taille était plus rapide que la vitesse de circulation sur un lien reliant deux villes de plus petite taille, nous avons noté que la vitesse était inversement proportionnelle aux numéros de catégories des villes. De plus, on a choisi de normaliser toutes les vitesses sur la plus grande vitesse possible : un lien direct entre deux villes de première catégorie.

On a ainsi abouti à cette équation pour le calcul d'une distance-temps euclidienne :

$$Te_{ij} = De_{ij} \cdot \left(0.8 + \frac{Cat_i + Cat_j}{10}\right)$$

Avec De_{ij} la distance euclidienne entre i et j, et Cat_i et Cat_j les catégories des villes i et j. La distance-temps réseau est calculée de la même manière :

$$Tr_{ij} = \sum_{i'j'}^{i \cdots j} \left(De_{i'j'} \cdot \left(0.8 + \frac{Cat_{i'} + Cat_{j'}}{10}\right)\right)$$

Avec i' et j' les villes constituant le plus court chemin à travers le réseau entre i et j.

Le calcul de plus court chemin avec Dijkstra Afin de calculer ce plus court chemin réseau, chaque lien étant pondéré par une vitesse, il a fallu mettre en place un algorithme de recherche de plus court chemin (PCC) supportant la pondération, ce qui n'est pas le cas, rappelons-le, des primitives de NetLogo. L'algorithme de PCC le plus utilisé de manière universelle porte le nom de son inventeur, Edsger Dijkstra [Dij59]. Cet algorithme garanti de trouver le chemin optimal entre deux nœuds nécessairement interconnectés d'un graphe. Son fonctionnement est assez intuitif, et repose sur le processus suivant (figure 9 page suivante).

Optimisation de la recherche de PCC. Si cet algorithme fonctionnait parfaitement, son temps d'exécution, bien que faible (de l'ordre du dixième de seconde) était tout de même trop long pour notre modèle, car répété des milliers de fois au cours d'un simulation. Voulant accélérer ce calcul, on a cherché d'autres algorithmes de plus court chemin nécessitant un coût-temps plus faible. Nos recherches nous ont rapidement orienté vers l'algorithme A*, variante plus efficace de Dijkstra.

L'utilisation de l'algorithme A*. A* est un algorithme basé sur le choix d'une heuristique [10]. En effet, quand Dijkstra reclasse sa liste de nœuds à parcourir en fonction du coût total parcouru pour y arriver, A* diversifie et raccourcit ce mécanisme en ajoutant un coût vers l'arrivée. Le poids d'un nœud dans la liste de parcours est ainsi égal au coût parcouru, pondéré par le coût estimé vers la destination. C'est la forme que prend cette pondération qui constitue l'heuristique de A*, selon que l'on choisisse de privilégier le coût parcouru, restant, ou encore que l'on décide de leur accorder une place équivalente (heuristique admissible). Selon le choix de l'heuristique, A* trouvera plus ou moins vite un PCC, mais sauf en utilisant une heuristique admissible, ce PCC ne sera que l'un des PCC possibles, et pas nécessairement le

10. Une heuristique est, en informatique, le choix d'une méthode de résolution d'un problème qui ne garantit pas forcément le résultat exact (on parle d'une heuristique admissible dans le cas contraire), mais peut accélérer le processus de recherche de la solution.

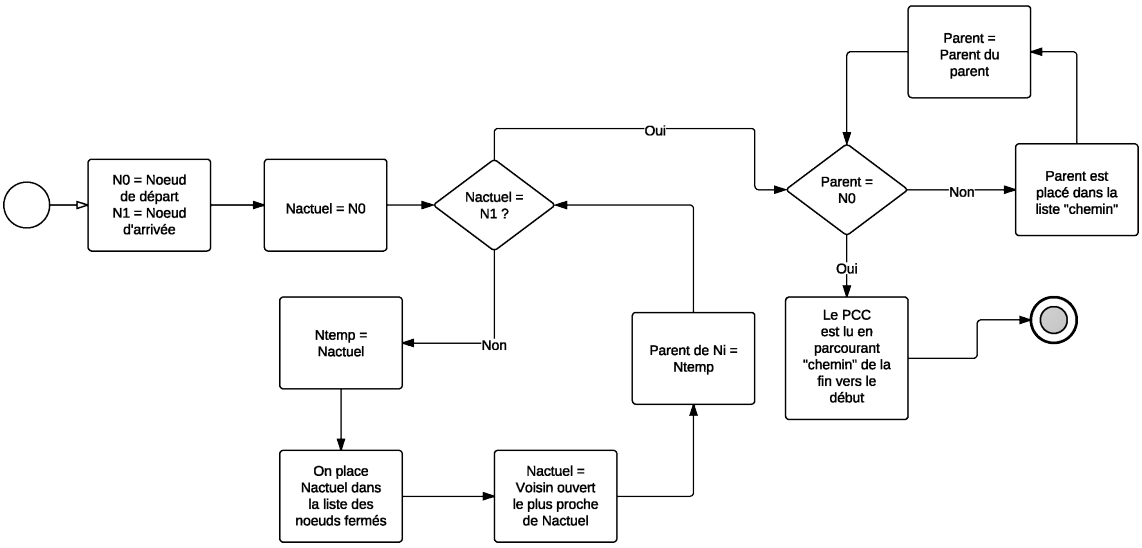

FIGURE 9 – Diagramme d'activité de l'algorithme de Dijkstra

véritable PCC. Dans notre cas, on ne cherchait pas forcément à avoir le temps de calcul le plus faible, mais à avoir un temps de calcul plus faible qui respecte cependant l'optimum du chemin trouvé. On a donc adopté une heuristique admissible basée sur la distance euclidienne séparant le nœud examiné du nœud d'arrivé.
Si un modèle NetLogo existait implémentant A* existait déjà [11], celui-ci se basait sur des parcours à travers les *patches*, quand nous souhaitions suivre un réseau pondéré. Il a donc fallu repartir d'une base quasi-nulle pour mettre en place l'algorithme. De plus, si A* (et Dijkstra) procèdent en temps normal par la mise à jour d'une liste de nœuds ouverts contenant leurs poids respectifs, nous avons choisi d'utiliser les spécificités de l'utilisation d'agents dans NetLogo en n'utilisant pas de liste, mais en incluant ces propriétés comme attributs des villes.
L'implémentation de cet algorithme dans NetLogo a requis un travail important, mais a porté ses fruits en ce qu'il a entrainé une diminution radicale du temps de calcul nécessaire. Nos réseaux de villes modélisés sont ainsi largement interconnectés, couvrant de manière importante l'ensemble du territoire modélisé, et ne présentant que peu d'isthmes [12]. Dès lors, A* tire pleinement parti de son potentiel, en ce que le chemin le plus court s'éloigne assez peu de la ligne droite, et est donc trouvé bien plus rapidement (c'est-à-dire en parcourant moins de nœuds) par A* que par Dijkstra.

2.3.4 Résultats.

Si nous n'avons pas eu le temps de tester en profondeur le modèle, des premières analyses ont toutefois permis d'en cerner les limitations actuelles.

Vers un réseau couvrant plus efficace. On souhaitait de ce prototype qu'il constitue, à partir d'un réseau initial quelconque ou en début de hiérarchisation, un réseau encore plus

11. cf. `http://turtlezero.com/models/view.php?model=a-star_2009`.
12. Un isthme est, dans le vocabulaire des graphes, une arrête dont la suppression diviserait le graphe en composantes connexes.

hiérarchisé et plus efficace par la création de liens rapides,c'est à dire que les liaisons créées permettent d'accélérer le cheminement sur le réseau.

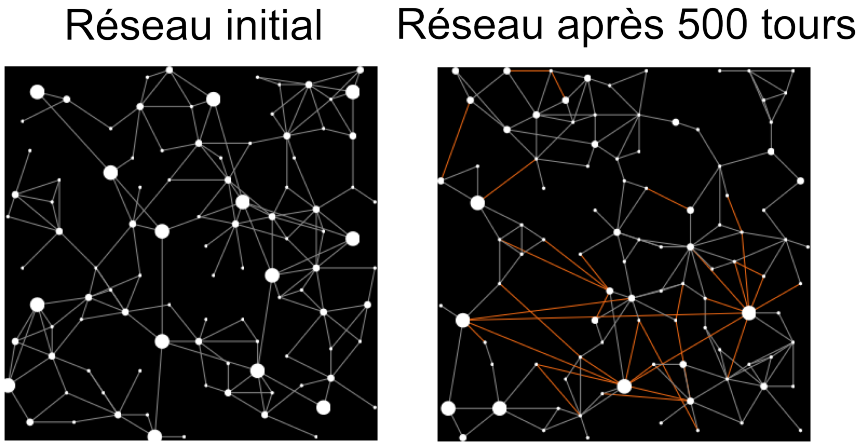

FIGURE 10 – Réseau initial et réseau obtenu en fin de simulation.

	Tous les liens	Liens initiaux	Liens créés
Poids des liens	1 043 267	956 221	1 174 298
Écart à la moyenne	0	-8,34%	12,56%

Préciser le modèle. Les expérimentations menées sur SimpopNet-Réseaux ont fait apparaître des limites. Tout d'abord, les mécanismes de création et de hiérarchisation des réseaux ne sont peut-être pas assez forts. Ainsi, les liens créés ne renforcent que peu la hiérarchie du réseau modélisé En deuxième lieu, un important besoin de calibrage : si les paramètres sont bien moins nombreux que dans SimpopLocal par exemple, ils jouent un rôle très important au sein de ce prototype pourtant simple. Ainsi, en faisant varier le paramétrage de création du réseau (les portées de disposition des villes ainsi que les portées de liaison initiales) et le seuil de sinuosité, on parvient à des formes de réseaux complètement différents. Il faudra donc bien veiller à adapter les paramètres pour obtenir un réseau le plus réaliste possible, voir trouver finalement un autre mécanisme de création de réseau. L'objectif de ce module est bien de se baser sur un réseau représentatif d'un réseau de transport (voies ferrées ou routières/autoroutières). Suite aux premières simulations, on peut constater le manque de réalisme des réseaux modélisés. Une étude plus poussée de la morphologie des réseaux réels permettrait sûrement de trouver des mécanismes produisant des morphologies plus représentatives. D'autre part, la topologie (relief) peut aussi fortement influencer la structure d'un réseau de communication. Aussi si certaines liaisons attendues directes n'existent pas entre des grandes villes, cela peut être expliqué par le relief. Une suite du développement deSimpopNet-Réseaux peut être menée sur l'incorporation des contraintes topologiques, une première implémentation ayant maintenant été effectuée par Clara Schmitt.

Troisième partie

Création d'outils d'analyses de trajectoires spatio-temporelles.

3.1 Permettre la visualisation - Première approche pour SimpopLocal.

Un besoin d'outils d'analyse. Chaque simulation produit un large volume de données, qu'il faut nécessairement analyser pour plusieurs raisons : vérifier que la simulation produit les résultats escomptés, analyser l'impact de la variation des valeurs de paramètre sur les sorties, mesurer la robustesse à l'aléa, ou encore comparer plusieurs scénarios de simulation. Si le traitement des sorties d'une simulation peut se faire épisodiquement, on a vu qu'à chaque fois, de multiples réplications étaient nécessaires, et dès lors, le volume de données à analyser augmente rapidement. Devant la multitude de données à traiter, le modélisateur ne peut mener des analyses individuelles et doit trouver ou créer des outils de traitement automatique des données. Puisqu'il n'existe pas d'outil résumant l'ensemble des phénomènes que l'on peut constater dans un jeu de données, il convient de se décider sur quelques indices à suivre, matérialisés par des indicateurs, et autant que faire se peut, veiller à automatiser leur production. Dans le cadre de ce stage, le premier besoin de faire appel à des outils d'analyse s'est fait sentir lors de l'analyse de l'impact des catastrophes dans SimpopLocal. Si l'on a déjà présenté les résultats de ces analyses (cf. figure 8 page 23), nous allons maintenant décrire les outils les ayant permis.

Les *RankClocks*. Le premier de ces outils a été développé par Michael Batty et son équipe du laboratoire CASA, qui l'ont conçu et créé pour permettre un nouveau type de visualisation des trajectoires de villes. Nommé *RankClock*, ce type de représentation graphique mise beaucoup sur l'impact visuel : il consiste à représenter le rang de chaque élément dans une hiérarchie, sur une période de temps donnée (spécifiée en paramètre), en organisant cette hiérarchie selon le rang initial. Cette méthode est originale en ce que la trajectoire des éléments est représentée sur un graphique circulaire et non linéaire, ce qui facilite le suivi des trajectoires individuelles. La définition d'un gradient de couleurs largement différencié permet aussi un aperçu rapide de l'évolution de sous-classes de la hiérarchie, en vérifiant les différences entre le gradient de départ et celui d'arrivée, comparaison facilitée en ce qu'avec la mise en forme circulaire, on peut immédiatement visualiser les différences entre les états initiaux et finaux du système. L'outil *RankClock* est pourtant assez peu pratique dans son utilisation, ne servant pour l'instant que de « Proof of Concept » : la manipulation de données externes n'est pas prévu, ce qui entraîne donc une procédure lourde pour réussir à exécuter l'outil sur celles-ci.

Traitements sous R. Il nous a donc fallu nous tourner vers une production propre d'analyses et de graphiques, que ce soit pour représenter l'évolution d'indicateurs au cours d'une simulation, ou encore, pour vérifier des hypothèses statistiques simples comme des corrélations. En raison principalement de son utilisation dans la communauté internationale de recherche et de sa disponibilité sous l'environnement GNU/Linux disponible au laboratoire, le logiciel libre R [13] a été adopté pour mener l'ensemble de ces analyses [14]. Au sein de l'IDE RStudio, facilitant l'utilisation de ce logiciel basé sur la ligne de commande, nous avons rapidement pu produire des graphiques présentant différents aspects d'une simulation, et dès lors, semi-automatiser ces séquences d'instructions pour une reproduction de ces routines sur différents jeux de données.

Analyse de l'impact des catastrophes Le besoin de production automatisée de rapports graphiques pour SimpopLocal s'est d'abord exprimé en prévision d'une réunion de présentation

13. `http://www.r-project.org`
14. On reviendra plus en détail sur le choix de cette plate-forme, cf. 3.2.1 page 34

du module de catastrophes de SimpopLocal. Pour ce faire, nous avons utilisé l'implémentation de l'outil de production de *RankClock* mise à disposition par l'Université du Nouveau-Mexique [15], qui nous avait été présenté lors d'un colloque par Michael Batty lui-même. Additionnée d'un graphique produit avec R, le rapport présenté a toutefois tant permis de sentir les limites du *RankClock* que le besoin de création d'un outil spécifique adapté à l'analyse des structures hiérarchiques des systèmes de peuplements, et donc utilisé pour analyser les sorties de simulation de SimpopLocal et plus tard de SimpopNet.

3.2 TrajPop.

3.2.1 Un outil d'analyse automatique des trajectoires de villes.

Observer et analyser le comportement d'un système de villes.

Besoin d'observation multi-scalaire. Avant de créer un outil d'analyse, il fallait définir précisément les besoins qui le motivaient. Dans le cadre des modèles de type SimpopLocal ou SimpopNet, on s'intéresse à l'évolution de la hiérarchie urbaine en tant que système. Cet axe d'étude entraîne la nécessité d'observer le déroulement d'une simulation à travers deux prismes :

1. L'évolution du système dans son ensemble - Échelle macroscopique.
   L'analyse de l'évolution d'un système de villes est très bien documentée et largement utilisée, on utilise pour cela des indicateurs tels que l'évolution de la population totale qui donne un aperçu du type de croissance à l'œuvre, ou encore l'analyse de la courbe rang-taille, selon les formes qu'elle prend à chaque pas de temps et sa forme finale. Les outils et mécanismes pour mener ce type d'analyses sont nombreux et leur usage est entré dans l'habitude des géographes, notre outil ne devait donc pas se focaliser sur ce point.
2. L'évolution spécifique des villes au sein du système - Échelle mezzoscopique.
   C'est à l'étude des composantes du système de villes que s'attachent désormais les géographes. En effet, l'analyse de l'évolution de chaque ville est fastidieuse et ne permet pas forcément de dégager les comportements d'ensemble chers au modélisateur. Le regroupement de villes selon leurs comportements dans le système permet par contre de donner autant un aperçu des composantes du système que de catégoriser les villes elles-mêmes dans ces composantes. C'est donc cette échelle d'analyse que notre outil devra privilégier, car les *RankClocks* ne permettent que la visualisation des mouvements des individus villes (échelle microscopique), sans une classification qui permettrait de synthétiser les données tout en leur ajoutant une information supplémentaire.

Limites du *RankClock*. En dehors de la faible utilisabilité de l'outil, qui aurait pu être résolue par un portage de la logique dans R par exemple, nous ne remettons pas ici en cause non plus la logique des *RankClocks* en elle-même, mais simplement, son adéquation à nos problématiques. En effet, l'outil de M. Batty met en exergue les variations de rang des différentes villes d'un système, tout en mettant ces variations sur un plan d'égalité : un perte de 2 rangs dans le bas du classement sera aussi visible qu'une perte de 2 rangs dans les premières villes. Pour autant, ces deux évolutions ne sont pas identiques, en ce que les dynamiques internes d'un système de ville ne sont pas semblables pour toutes villes, un basculement en tête de hiérarchie ayant bien plus d'importance qu'en sa traîne. Dès lors, il nous

15. `http://sev.lternet.edu/RankClocks/`

fallait concevoir un outil montrant aussi bien les faibles variations que les plus importantes, tout en les différenciant nettement, c'est-à-dire un outil dressant le portait de catégories de villes, regroupés autant par leurs tailles que par leurs évolution dans le système.

Un nouvel outil de classification.

Logique de fonctionnement. TrajPop vise à classifier automatiquement les villes en fonction de leurs trajectoires individuelles au cours du temps. Pour ce faire, il a été choisi de procéder à une Classification Ascendante Hiérarchique (CAH), afin de grouper les villes en un certain nombre — défini par le simulateur — de classes. Cette CAH est précédée de la réalisation d'une Analyse Factorielle des Correspondances (AFC), utile pour l'affichage des classes, dont on conserve l'ensemble des axes pour créer une matrice de distance au χ^2. Cette méthode est indispensable dans notre cas, car on souhaite classifier les villes selon leur évolution autant que selon leurs poids. Le χ^2 permet donc d'obtenir une distance qui tient en compte les lignes ainsi que les colonnes, c'est-à-dire le poids dans le système de chaque ville pour chaque pas de temps ainsi que les progression relatives des villes entre deux pas de temps. On réalise enfin la CAH selon le critère de Ward qui minimise l'inertie intra-classe et permet donc d'avoir des classes homogènes. Une fois la classification effectuée, on peut représenter les moyennes de chaque classe et ainsi observer les trajectoires qui caractérisent chacune de ces classes.

La mise en place avec R. Pour réaliser ces analyses automatiques, nous avons choisi de nous appuyer sur le logiciel R, que nous avions déjà utilisé pour produire quelques graphiques de sorties de SimpopLocal. En sus de cet usage passé, R a été préféré aux autres solutions disponibles pour les raisons suivantes :

R est un logiciel libre. En dehors de NetLogo, nous avons voulu, pendant ce stage, ne faire appel qu'à des logiciels libres. Pour une raison pratique d'abord, en ce que les logiciels libres sont le plus souvent fonctionnels sous les systèmes GNU/Linux. Pour des raisons plus idéologiques ensuite, en ce que les réalisations faites au cours du stage devaient pouvoir servir à qui en aurait besoin, en dehors même des membres du laboratoire. De plus, les logiciels libres reposant le plus souvent sur une communauté d'utilisateurs et de développeurs, l'accès à de la documentation et à du support communautaire est facilité et important.

R est largement utilisé dans la communauté scientifique. Prenant peu à peu la place de logiciels propriétaires tels que SAS, SPAD ou encore SPSS, R devient, si ce n'est pas encore le cas, un nouveau standard *de facto* dans le monde de la recherche universitaire, mais aussi dans les institutions de recherches publique mondiales, ou même de grandes sociétés telles que Google ou Pfizer. Cette utilisation massive entraîne une large diversification des possibilités du logiciel de base, au travers de modules ("packages"), chacun étant susceptible d'ajouter ce dont il a besoin au logiciel sous la forme d'un package.

R est scriptable. Si l'interface graphique n'est pas la préoccupation majeure de l'équipe de développeurs du logiciel, c'est que R est avant tout un langage de programmation, le logiciel interprétant les commandes entrées par l'utilisateur, dans le même esprit que SAS. Cette orientation vers l'exécution de code peut se révéler difficile d'approche au départ, car nécessitant une compréhension de la logique utilisée en programmation, mais l'apprentissage en est facilité dans R par la présence de nombreux exemples et démonstrations. Dans notre cas, on voulait pouvoir proposer aux expérimentateurs un outil simple à manier, où il y aurait un minimum de paramètres à entrer, tant par souci de de facilité d'usage pour un utilisateur novice que pour un utilisateur avancé ayant besoin d'effectuer rapidement de nombreuses analyses de sorties de simulations. On peut

donc créer tout un scénario d'analyse de données et ne laisser pour tâche à l'utilisateur que de cliquer sur le bouton de lancement.

R propose des interfaces graphiques simples d'accès. Si R peut pécher par une difficulté d'entrée trop importante pour le novice, de nombreux utilisateurs ont décidé de remédier à cette situation en créant des IDE (Environnement de Développement Intégrés en français) faisant office d'interface graphique tout autant que de plate-forme de développement. Parmi eux, RStudio permet une visualisation simple des tableaux de données ainsi que des graphiques produits, et propose une auto-complétion complète ainsi qu'une aide bien intégrée et facile d'accès.

Autres avantages. Notons encore que R est compatible avec une large gamme de formats de données, dont le support est apporté par des packages développés spécifiquement par la communauté, est est donc capable de se plier aux besoins des utilisateurs sans forcer ceux-ci à préparer leurs données par de complexes opérations d'export et de modifications. R dispose aussi d'un module cartographique lui permettant de produire des cartes de qualité au sein même du logiciel, et qu'à l'inverse, il est intégrable par le biais de plugins dans des SIG tels que ArcGIS (depuis la version 10) ou encore Quantum-GIS (via le plugin ManageR). Enfin, et raison non des moindres, R est maîtrisé par une bonne partie de l'équipe de l'UMR Géographie-cités, et un groupe de travail visant à l'auto-formation est même en train de se mettre en place.

R semblait donc tout désigné pour réaliser cet outil, permettant à l'utilisateur final de TrajPop de traiter aisément ses données, tout autant qu'à le former, et à nous former, à l'utilisation d'un logiciel d'analyse statistique avancé.

Principe de base du code Si l'analyse des données en elle-même est assez rapide, via l'utilisation du package ade4 développé par une équipe du Laboratoire de Biométrie et Biologie Évolutive (UMR 5558) de l'Université Lyon 1 [DDC], l'adaptation aux données de l'utilisateur complexifie considérablement le code, de même que la production des graphiques.

Les traitements peuvent se résumer à ces lignes :

```
# On importe nos tables dans la variable mydata.data
mydata.data <- read.csv(FilePath)
# On crée un nouveau tableau sans la première colonne de nom (v1...)
mydata.calc <- as.data.frame(mydata.data[FirstDataCol:EndDataCol])
# On supprime les levels, inutiles ici.
levels(mydata.calc) <- NA
# On s'assure que le package ade4 est bien installé
if (is.element('ade4', installed.packages()[,1]) == FALSE
) { install.packages('ade4') }
# Chargement de la librairie ade4 qui calcule AFC et CAH
library('ade4')

# AFC ET CAH

# Réalisation de l'AFC, on conserve tous les axes (1 axe/colonne)
mydata.afc <- dudi.coa(df=mydata.calc, scannf=FALSE,
nf=(EndDataCol - FirstDataCol + 1))
# Création des matrices de distance (khi2) à partir des coordonnées de l'AFC
# (sur l'ensemble des axes)
mydata.dist <- dist.dudi(mydata.afc)
```

```
# On execute la CAH sur la matrice des distance élevées au carré
# (calcul de l'inertie avec critère de Ward)
mydata.cah <- hclust(mydata.dist^2, method="ward")
# On découpe notre CAH en nbclasses classes
mydata.classes <- cutree(mydata.cah, k = nbclasses)
```

On notera que le code est entièrement commenté, dans un but didactique, afin que les utilisateurs puissent le ré-utiliser et/ou le modifier à leur guise. L'usage qui est fait de TrajPop par différents chercheurs, sur lequel nous reviendrons plus en détail, montre que cet outil est totalement fonctionnel, ayant mê me été adapté grâce à sa lisibilité à des formats de données non prévus au départ.

3.2.2 Les rapports produits.

Une production de graphiques.

Afin que l'outil présente une véritable simplification de la tâche pour le modélisateur, nous avons voulu que les résultats produits soient accessibles directement (sans avoir besoin de passer par un autre logiciel), compréhensibles et analysables. Pour ce faire, le rendu de TrajPop est constitué d'un fichier PDF présentant plusieurs graphiques, lesquels tendent, par leur complémentarité, à donner un premier aperçu de l'évolution du système de villes représenté.

Graphiques de contexte. On a vu que TrajPop réalise une classification des villes. Pour en rendre compte tout autant que pour permettre l'explication de la composition des classes, il fallait synthétiser leur constitution. On a choisi de rendre celle-ci par la création de deux graphiques (figure 11 page suivante) : Un dendrogramme, ou arbre de classe, permet de constater les écarts entre les classes ainsi que leur ordre de succession. On peut ainsi rapidement noter quelles classes sont les plus proches les unes des autres et avoir un aperçu de la proportion de villes incluses dans chaque classe. Le second graphique est une simple représentation des coordonnées de chaque classe selon les deux principaux axes des l'AFC. Ce graphique permet lui aussi de distinguer les classes en fonction de leur proximité les unes des autres, ainsi qu'en fonction de l'importance de la participation de chaque axe dans leur définition.

Graphique 1 : Population moyenne des classes. Les graphiques de la figure 12 page 38 montrent l'évolution des classes de ville au cours du temps.

Le premier représente, pour chaque classe, la population moyenne (qui constitue donc la population totale de chaque classe divisée par le nombre de villes qu'elle contient) à chaque pas de temps. C'est véritablement ce graphique qui permet de suivre la trajectoire des villes du système, en ce que cet appel à la moyenne de population sur toute la durée de la simulation donne aussi une idée de la place hiérarchique des villes composant la classe.
Notons que pour ces graphiques comme pour les suivants, on a choisi de créer deux pages distinctes de graphiques, l'une comportant le tracé d'une classe par graphique, et contenant le nombre de villes de cette classe comme information supplémentaire, et l'autre regroupant les tracés de chaque classe dans un graphique unique. Le choix de conserver ces deux modes de représentation a été motivé la volonté d'avoir à chaque fois une représentation claire des trajectoires de classe, quelque soit la constitution et la morphologie de chacune d'elles.

Graphique 2 : Poids des classes dans le système de villes. Le deuxième type de représentation (figure 13 page 39) montre l'évolution de la part de chaque classe dans l'ensemble du système de villes. En croisant les tracés et le nombre de villes de chaque classe, les

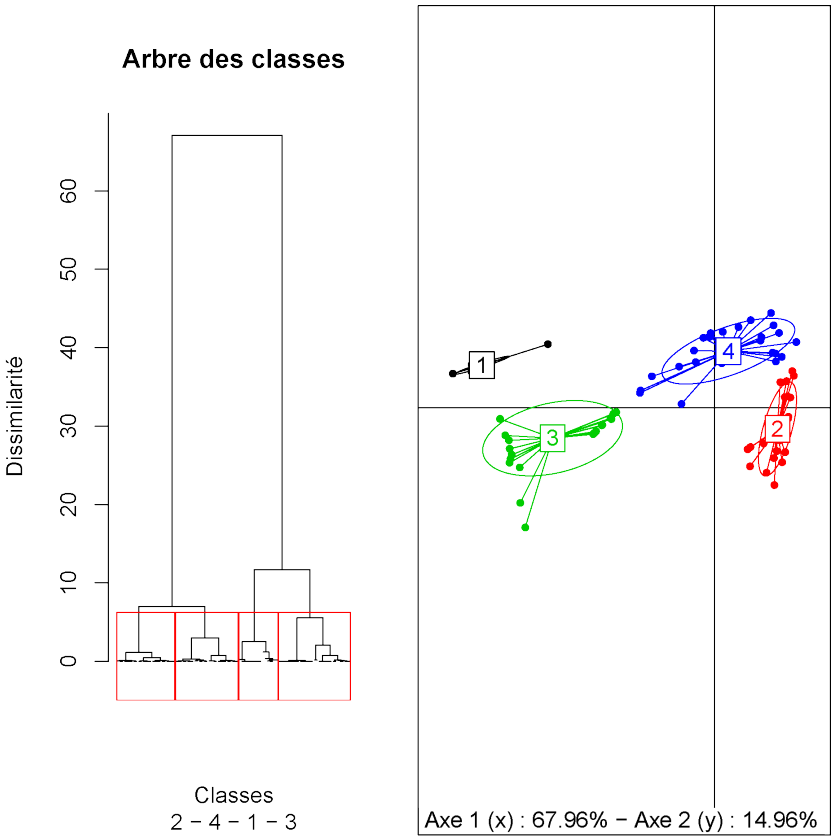

FIGURE 11 – Les graphiques de contexte de TrajPop.

éventuels bouleversements de la hiérarchie urbaine sont illustrés. Par rapport au *RankClocks*, les évolutions sont ici relatives au poids occupé dans le système, et laissent donc bien à voir les perturbations majeures dans la hiérarchie.

Sorties numériques

Pour clarifier les sorties de TrajPop et pour que de plus fines analyses puissent être menées, il était voulu que le tableau de donnée entré par l'utilisateur soit enrichi des informations calculées par l'outil, et que tous les indicateurs représentés dans les graphiques soient accessibles en tant que données brutes, donc exportables.

Le tableau initial enrichi. En premier lieu, on exporte donc, à l'endroit choisi par l'utilisateur, le tableau d'entrées. Celui-ci comporte l'identification des villes et leurs populations à chaque pas de temps. On ajoute à ces informations les classes dont font parti chaque ville, que ce soit pour que l'utilisateur puisse mener des analyses plus approfondies (par exemple en regardant quelles sont les villes rassemblées dans chaque catégorie et en cherchant les points historiques, sociaux ou culturels qu'elles ont en commun) ou encore pour que ces classes puissent être cartographiées de manière simple (par exemple via import dans un SIG ou un dans logiciel de cartographie automatique).

Tableau des calculs Afin que l'utilisateur puisse aussi contrôler les informations présentées dans les graphiques, dans une volonté de vérification, ou bien qu'il puisse créer de nouveaux graphiques depuis les données calculées par l'outil, TrajPop génère aussi un tableau de données (figure 14 page 39), toujours au format CSV, contenant les indicateurs représentés dans les graphiques, qui permettent de résumer l'évolution des classes.

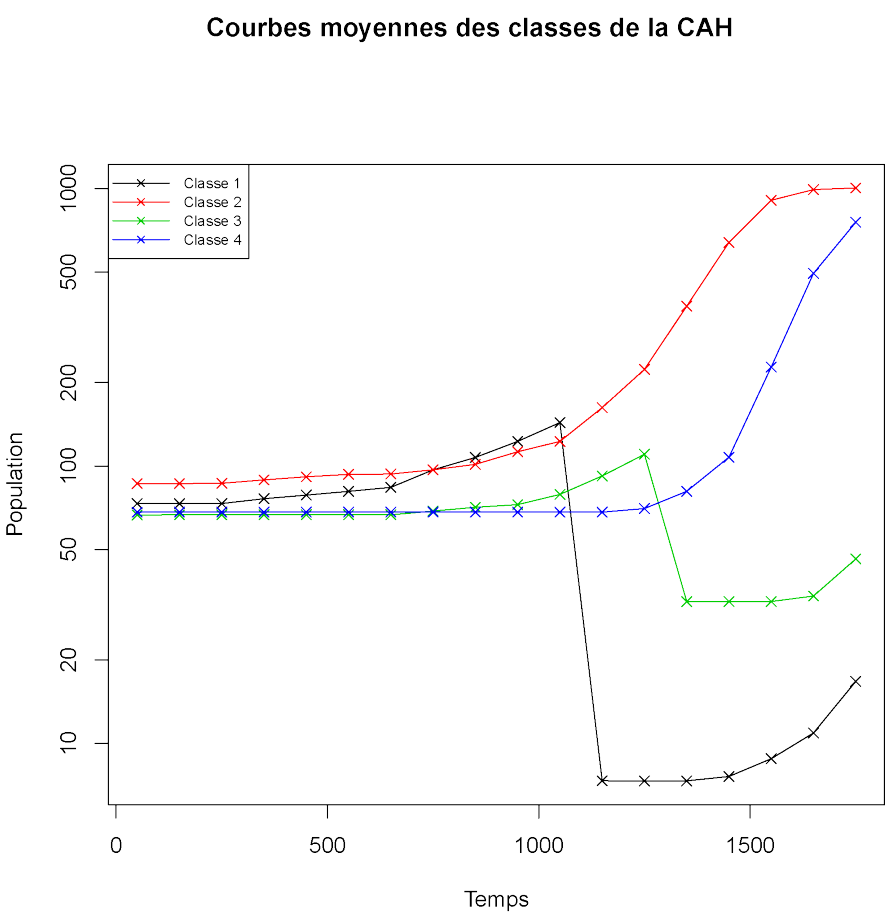

FIGURE 12 – Évolution de la population moyenne des classes.

Généralisation de l'outil. Il a été vu que l'objectif initial de TrajPop consistait au traitement des sorties de simulation des modèles SimpopLocal, avec et sans catastrophes, ainsi que du modèle SimpopNet à venir. Pour autant, les cas d'utilisation de cet outil dépassent assez largement ces usages spécifiques, en ce qu'il permet d'étudier les trajectoires de villes dans un système aussi bien modélisé que réel. Ainsi, des doctorants de l'UMR Géographie-cités utilisent d'ores et déjà TrajPop dans le cadre des systèmes de villes qu'ils étudient. C'est notamment le cas de Clémentine Cottineau sur les villes russes (voir figure 15 page 40), de Antonio Ignazzi pour l'analyse de simulations de Simpop2, ou encore de Elfie Swerts sur les systèmes urbains indiens et chinois. Pour leurs recherches, TrajPop constitue un outil efficace de schématisation de l'évolution des villes et une condition préalable à des analyses plus approfondies, qu'elles soient à d'autres échelles, ou en tenant compte de la dimension spatiale, absente de TrajPop.

Perspectives d'évolution de TrajPop. TrajPop a été conçu et développé en peu de temps, afin de répondre rapidement à un manquement dans les outils de visualisation des données. Pour autant, ce script pourrait servir de base à un outil plus large, lequel viserait à diversifier les informations apportées, que ce soit en apportant des analyses à d'autres échelles, en précisant les graphiques déjà créés, ou encore en permettant une cartographie automatique des résultats. Pour ce dernier point, R propose des outils et packages facilitant la création de cartes d'une grande qualité tant thématique qu'esthétique (via le package maps pour la cartographie basique, rgdal pour le traitement des fichiers vectoriels géographiques, et par exemple RColorBrewer pour les rampes de couleurs). La création de graphiques Rang-Taille et la représentation de leur évolution dans le temps donnerait à voir la forme de la distribution des villes dans le système et de la dynamique d'ensemble à l'œuvre. Il est bien plus complexe

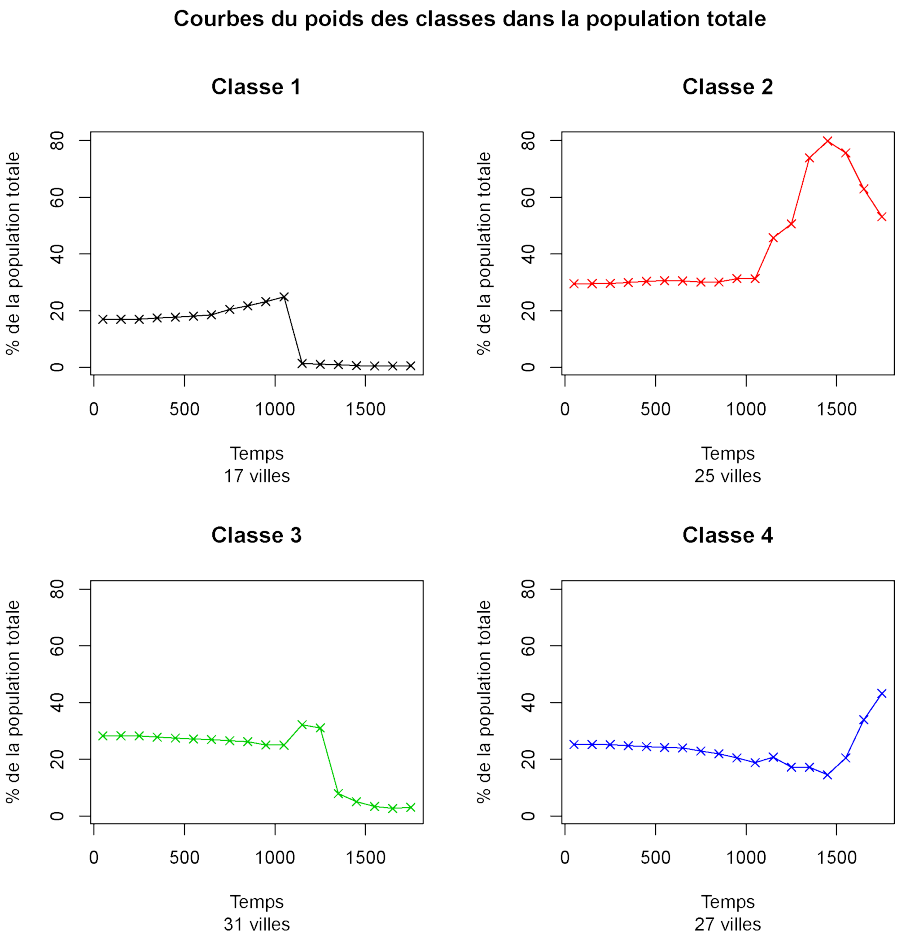

FIGURE 13 – Évolution du poids des classes dans le système de villes.

	MeanPop Classe1	MeanPop Classe2	MeanPop Classe3	MeanPop Classe4	MeanPop Classe5	MeanPop Classe6	TotalPop	MeanPop	PcPop_ Classe1	PcPop_ Classe2	PcPop_ Classe3	PcPop_ Classe4	PcPop_ Classe5	PcPop_ Classe6
50	73,1	86,4	73,6	71,8	64,2	59,2	7313,4	73,1	17,0%	29,6%	16,1%	13,7%	11,4%	12,1%
150	73,2	86,4	73,6	71,8	64,3	59,3	7314,0	73,1	17,0%	29,5%	16,1%	13,7%	11,4%	12,2%
250	73,2	86,7	73,6	71,8	64,3	59,3	7319,4	73,2	17,0%	29,6%	16,1%	13,7%	11,4%	12,1%
350	76,2	89,1	73,6	71,8	64,3	59,3	7432,2	74,3	17,4%	30,0%	15,8%	13,5%	11,2%	12,0%
450	78,5	91,5	73,6	71,8	64,3	59,3	7530,1	75,3	17,7%	30,4%	15,6%	13,3%	11,1%	11,8%
550	81,0	93,3	73,6	71,8	64,3	59,3	7617,9	76,2	18,1%	30,6%	15,5%	13,2%	11,0%	11,7%
650	82,8	93,8	73,6	71,8	64,3	59,3	7671,9	76,7	18,6%	30,5%	15,4%	13,1%	10,9%	11,6%
750	96,9	96,9	77,8	71,8	64,3	59,3	8043,6	80,4	20,5%	30,1%	15,5%	12,5%	10,4%	11,1%
850	107,4	101,4	82,0	71,8	64,3	59,3	8403,7	84,0	21,7%	30,2%	15,6%	12,0%	9,9%	10,6%
950	122,7	112,6	85,1	71,8	64,3	59,3	8991,6	89,9	23,2%	31,3%	15,1%	11,2%	9,3%	9,9%
1050	143,5	122,7	97,5	71,8	64,3	59,3	9796,1	98,0	24,9%	31,3%	15,9%	10,3%	8,5%	9,1%
1150	7,3	162,4	122,9	71,8	64,3	59,3	8880,8	88,8	1,4%	45,7%	22,1%	11,3%	9,4%	10,0%
1250	7,3	223,1	158,1	75,1	65,0	59,3	11017,3	110,2	1,1%	50,6%	23,0%	9,5%	7,7%	8,1%
1350	7,3	376,9	7,4	87,2	74,5	59,3	12742,7	127,4	1,0%	73,9%	0,9%	9,6%	7,6%	7,0%
1450	7,6	640,3	7,4	133,0	80,4	59,3	20050,2	200,5	0,6%	79,8%	0,6%	9,3%	5,2%	4,4%
1550	8,8	908,4	7,4	346,3	99,8	59,3	30011,5	300,1	0,5%	75,7%	0,4%	16,2%	4,3%	3,0%
1650	10,9	992,5	7,4	769,7	199,6	62,4	39422,2	394,2	0,5%	62,9%	0,3%	27,3%	6,6%	2,4%
1750	16,7	1006,0	8,5	965,5	532,0	86,5	47300,6	473,0	0,6%	53,2%	0,3%	28,6%	14,6%	2,7%

FIGURE 14 – Aperçu d'une des sorties numériques de TrajPop.

d'apporter des renseignements à l'échelle microscopiques dans un format statique. Plusieurs pistes peuvent être envisagées pour remédier à cela, par exemple l'usage du package GoogleViz qui permet de réaliser des graphiques et cartes dynamiques, pour un affichage sur internet. Cela permettrait autant d'avoir une vision d'ensemble, dans les graphiques présentés, correspondant aux informations déjà produites par TrajPop, que de permettre de « zoomer »sur une ville précise. Autre piste, la production de rapports dynamiques ouvrirait de plus la voie à une interaction entre l'utilisateur et les données à représenter, en simplifiant d'autant l'utilisation de TrajPopvia une interface web, R fonctionnant en arrière plan pour produire des rapports à la demande.

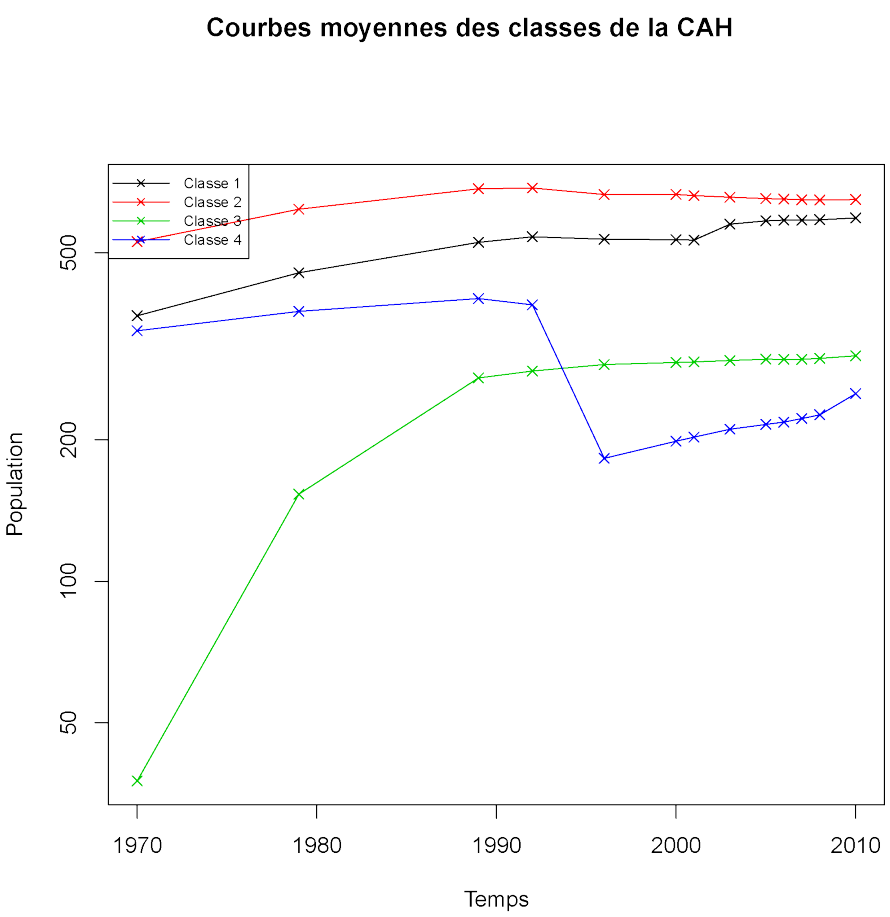

FIGURE 15 – Un exemple d'application de TrajPop à un système de villes réel : trajectoire des principales villes russes de 1970 à 2010.

3.3 qFlowMapper.

3.3.1 État de l'art et besoins.

Besoins métiers.

Objectifs du plugin. En pratique, les sorties de simulations sont la plupart du temps étudiés via des indicateurs numériques créé sur mesure, et affichable sous la forme de graphiques ou de tableaux de mesures. TrajPopest un bon exemple des outils qui peuvent découler d'une telle pratique. Si ces outils sont bien adaptés à une lecture systématique des résultats, ils s'avèrent en revanche beaucoup plus limité lorsqu'il faut analyser des imbrications complexes de flux. L'appel à des représentation et des outils issues de la cartographie améliorent certes cette lecture, mais amène nous le verrons d'autres problèmes : l'impact lié à l'insertion d'une nouvelle dimension spatiale dans la représentation des données n'étant pas négligeable dans l'image de la réalité rendu au lecteur « naïf »de la carte. Les trajectoires sont en effet « réalisées »dès lors qu'on leur impose une spatialité, que celle-ci soit réelle ou issue d'un algorithme de regroupement des flux. En un sens, nous pouvons dire ici que « traitements »et « résultats des traitements »deviennent difficilement dissociable. De ce constat apparaissait le besoin d'une représentation des flux sous forme cartographique, si possible via intégration dans une plate-forme capable autant de traitement que d'affichage des résultats. Pour autant, on a vu que les visualisations cartographiques présentaient jusque là un manque d'adaptation à nos résultats, en particulier concernant la représentation et l'analyse des flux. La principale difficulté dans la cartographie de flux est que selon leur quantité ou leur organisation,

le cartographe doit gérer différents types de représentation. En effet, selon que les flux aient une importante densité sur un espace réduit, présentent une couverture globale de l'espace cartographié, ou encore soient en faible nombre, le type de représentation approprié à une visualisation claire variera largement. Il fallait donc que notre mode de visualisation permette autant de traiter les flux de manière individuelle que collective, en ce que les modes de représentation doivent être adaptés aux données. On doit par exemple pouvoir représenter les flux via une carte « en oursin »quand ils présentent une forte polarité, et au contraire, via des courbes de Bézier quand la couverture du territoire est plus homogène ou le nombre de flux réduit.

Intégration dans un SIG. En dehors de cet aspect visuel, nous avions besoin de pouvoir mener de véritables analyses spatiales sur les flux produits par les simulations. Pour cela, il a été décidé d'utiliser un SIG comme support de la production cartographique. Le choix de ce type de plate-forme offrait en effet des possibilités vastes en traitement de l'information, que ce soit par des croisements avec des données géographiques, par l'intégration de données externes, ou encore, à un niveau plus cartographique, par la possibilité de sélectionner les flux à cartographier selon des critères spatiaux ou attributaires.

Qui plus est, les SIG permettent à leurs utilisateurs de se déplacer, d'ajouter des fonds de carte, et ainsi, dépassent très largement le simple cadre d'une visualisation statique. L'export de cartes, facilité, est ainsi la dernière étape d'un réel traitement, ce qui ne serait pas possible si la production de cartes était entièrement automatisée et statique.

État de l'art.

Comme pour les autres types de données géographiques, la représentation des flux est possible dans la totalité des SIG existants, en ce qu'ils ne constituent au final que des figurés linéaires simples. Pour autant, la cartographie des flux est véritablement le parent pauvre des SIG, ceux-ci demandant un traitement différent de la représentation des réseaux par exemple. Nous n'avons ainsi trouvé aucun SIG capable de produire, nativement, une carte de flux. Nous présenterons donc ici les solutions existantes, ainsi que les défauts qui y ont été vus.

ArcGIS - Oursins et Outils de Flux. Malgré sa position dominante dans le marché et l'importante panoplies d'outils déployés dans ce SIG, ArcGIS ne permet pas de travailler sur des flux nativement. Ce n'est que par le biais de scripts externes que cela devient possible. Gaëtan Lavenu, employé chez ESRI France, a ainsi créé deux outils répondant au besoin de représentation de flux :

Oursins. Cet outil permet la création de flux, sous forme de figurés linéaires, depuis deux fichiers de forme ponctuels, le premier comprenant l'identifiant de la destination à chercher dans le second. Cet outil est très incomplet pour notre usage en ce qu'il ne permet de créer qu'un unique flux par origine, ce qui ne peut déboucher que sur un « oursin »simple.

Outils de Flux. Ce second script offre de plus larges possibilités : il prend en entrée un fichier de forme quelconque ainsi qu'une table origine-destination. Des flux soit linéaires soit circulaires sont alors créés entre les géométries définies par les identifiants indiqués dans la table.
Ces deux outils (cf. figure 16 page suivante), s'ils ont le mérite de permettre la création d'une couche de flux dans ArcMap, se bornent toutefois à celle-ci et dès lors, ne répondent pas à nos

besoins, tout le traitement cartographique devant alors être réalisé après export des flux dans un format vectoriel, au sein d'un logiciel de dessin tel qu'Inkscape ou Adobe Illustrator.

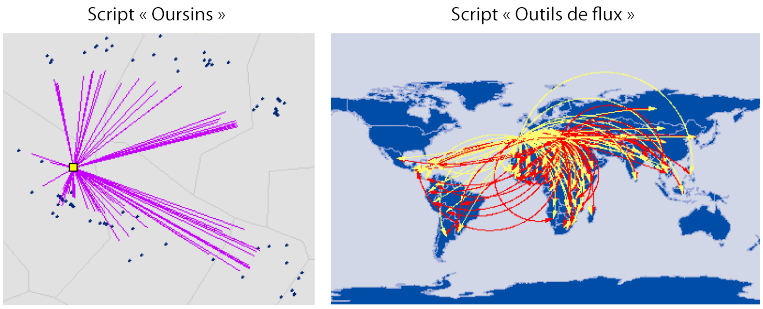

FIGURE 16 – Illustration des résultats produits par les scripts « Outils de Flux »et « Oursins ».

Notons que depuis la dernière version de la suite ArcGIS (ArcGIS 10), ces outils ont été intégrés nativement dans ArcMap. Si leur refonte rend la création des flux plus aisée et permet de donner une forme plus complexe à ceux-ci (prenant en compte la géodésie dans leur parcours), ils se cantonnent toujours à la création de couches, et ne permettent donc pas de traitement véritablement cartographique.

MapInfo - Outil de Stéphanie Julinet. Stéphanie Julinet, ancienne élève de notre formation, a développé un outil de représentation des flux, basé sur MapInfo, en 2005, dans le cadre du projet de développement. Cet outil vise à la création de flux entre des entités polygonales, flux décrits dans une table origine-destination externe. Le plus grand rôle de l'outil est de permettre la représentation d'une fraction des flux, l'utilisateur ayant le choix entre différents indicateurs (voir la figure 17 page 42).

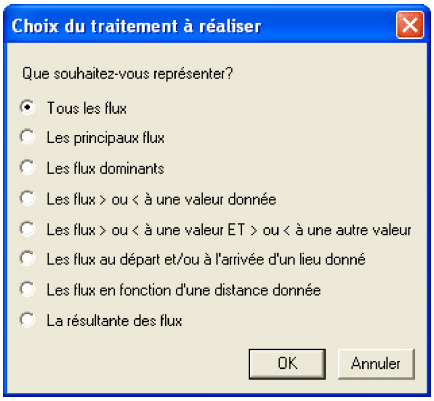

FIGURE 17 – Choix des flux à représenter dans l'outil de Stéphanie Julinet.

Si cet outil facilite la création et l'analyse de flux pour l'utilisateur, il ne propose en fait qu'une simplification, au moyen de scénarios courants, des opérations de traitements des flux les plus fréquentes. Ainsi, la partie de choix des flux à représenter qui constitue le cœur du programme ne consiste qu'en la création d'un menu effectuant une sélection que l'utilisateur pourrait tout à fait définir lui-même au moyen d'une requête SQL. Un point important est aussi consacré à l'esthétique des flux, par le choix de l'attribut définissant leur taille, leur couleur, ou encore par le type de discrétisation à employer. Tous ces traitements sont, dans les logiciels SIG récents, inclus, et cet outil n'apporte donc plus aujourd'hui une réponse satisfaisante à nos besoins.

Outils dédiés et bibliothèques de fonctions. Si les autres logiciels SIG ne proposent pas d'outils spécifiques à la cartographie de flux, ils peuvent s'appuyer, avec des intégrations plus ou moins poussées, sur des bibliothèques de fonctions dédiées, c'est par exemple le cas des logiciels libres uDig et gvSIG. Ces deux logiciels peuvent ainsi utiliser la bibliothèque d'algorithmes SEXTANTE, développée en Java, qui offre des fonctions avancées de création et de traitement des flux. Pour autant, l'appel à SEXTANTE est peu pratique pour l'utilisateur, en ce que l'interface graphique des fonctions est tout au moins rudimentaire. A l'instar de quelques fonctions proposées par GRASS, SEXTANTE offre ainsi des possibilités de traitement de flux complexes, mais leur visualisation ne constitue pas une priorité, de même que la simplicité de mise en place.

Notons enfin qu'il existe aussi des outils dédiés, qui ne sont pas des SIG, mais permettent d'utiliser des données géographiques et d'y appliquer des traitements spécifiques, c'est par exemple le cas de *FlowMapper*[16], outil basique de création de flux et d'export cartographique, ou encore de *jFlowMap*[17] permettant l'application de certains algorithmes sur des fichiers géographiques. On pourrait aussi citer *Flowmap* ([18] qui permet un partitionnement des régions d'études en fonction des flux qui les traversent, ou encore le très utilisé Philcarto [19] qui, dans sa version 4, permettait le tracé de flux proportionnels.

Conclusion de l'état de l'art. Tous les outils présentés ici ne correspondaient au mieux que partiellement à nos besoins de cartographie des flux de sortie de nos simulations. Que ce soit par des fonctions trop simples ou par une intégration absente ou insuffisante dans des logiciels SIG, ils ne pouvaient permettre de traiter l'ensemble des étapes nécessaires à nos sorties cartographiques. Ainsi, certains de ces outils proposent juste de la visualisation sans possibilité d'export ou de récupération des données. Ce problème d'export des données est généralisable en ce que les solutions présentées ne sont que difficilement interfaçables avec d'autres outils d'analyse de données. Les outils les plus aboutis, comme le plugin MapInfo, souffrent d'une absence de suivi qui en plus de les rendre aujourd'hui obsolètes, empêche leur enrichissement par le choix de ne pas diffuser les sources. D'une manière plus globale, le manque de cohésion et de volonté de créer un outil généraliste de cartographie et d'analyse des flux force l'utilisateur à alterner entre toutes ces solutions disponibles : elles sont toutes conçues comme des outils spécifiques, et rien ne permet de les assembler pour avoir un outil intégré aux possibilités plus larges. Il a donc été décidé de créer un nouvel outil de cartographie des flux.

Les choix effectués.

Le choix d'un SIG libre. Tout au long de ce stage, on a tenu à utiliser en priorité des outils libres, comme on l'expliquait pour R. Dans la galaxie du libre, il existe quatre principaux logiciels de SIG :

GRASS. GRASS est le plus ancien SIG libre, puisque le projet a été initié en 1982. Il propose une grande quantité de traitements possibles, et constitue en ce domaine la référence toutes catégories confondues des logiciels SIG. Si son évolution lui a apporté une interface graphique, il reste toutefois difficile d'accès, la partie visualisation n'étant que

16. `http://www.csiss.org/clearinghouse/FlowMapper/`
17. `http://code.google.com/p/jflowmap/`
18. `http://www.spatialdatamining.org/software/flowmap`
19. Développé par Philippe Waniez, téléchargeable et documenté à cette adresse : `http://philcarto.free.fr`

peu développée et la majorité des outils se lançant en ligne de commande. Son utilisation encore réservée à des géomaticiens chevronnés ne répondait pas à nos besoins, d'autant qu'on peut faire appel à la majorité de ses fonctions dans des SIG plus « user-friendly ».

uDig. Ce logiciel est en fait une extension pour l'Environnement de Développement Intégré (IDE en Anglais) Eclipse. Conçu en Java, il sert surtout à la visualisation d'informations géographiques, même s'il peut utiliser des fonctions de GRASS ou encore de la bibliothèque SEXTANTE. En terme de communauté, uDig est majoritairement utilisé et améliorer par un public qui est plus informaticien que géographe, public pour lequel l'utilisation d'informations géographiques ne constitue pas une spécialité, mais un besoin d'accès à des données de ce type parmi d'autres.

*gvSIG.*Créé par la collectivité locale de Valence, gvSIG est sans conteste le logiciel SIG le plus utilisé dans le monde hispanique. Solution capable de faire concurrence aux SIG non libres, il se repose sur une base d'utilisateurs, en particuliers dans des institutions publiques, très large, et son développement avance rapidement grâce à de nombreux contrats qui tendent à lui ajouter les fonctionnalités dont les contractants ont besoin. Cet appui quasi exclusif sur les institutions publiques l'empêche toutefois d'avoir une communauté d'entraide active et de s'ouvrir aux utilisateurs plus occasionnels.

Quantum-GIS. Développé en 2002 et projet phare de l'Open Source Geospatial Foundation (OSGeo) depuis 2007, QGIS est le plus récent mais peut-être déjà le plus abouti des SIG libres. Basé sur une communauté de contributeurs et d'utilisateurs avancés et novices imposante, il se voulait au départ simple interface graphique pour GRASS, mais a rapidement évolué en héraut du SIG libre, particulièrement dans le monde anglophone et francophone, jouant le rôle de concentrateur de fonctions de traitement géomatiques au centre de la stratégie de l'OSGeo, respectant qui plus est les consignes de l'Open Geospatial Consortium. Le fort taux de contribution et d'ajout est en grande partie du à sa conception modulaire à l'extrême, la vaste majorité des fonctions qu'il propose étant l'œuvre d'extensions qui s'installent facilement pour l'utilisateur, au moyen d'un gestionnaire d'extension recensant et décrivant toutes celles-ci (plus de 150), et peuvent être créés par des non spécialistes grâce à la mise à disposition d'une Interface de Programmation (API) très complète. C'est sur ce SIG que notre décision s'est arrêtée, principalement en raison de sa forte croissance, de sa simplicité d'utilisation ainsi que de l'aisance avec laquelle des extensions peuvent être développées, notamment grâce à son importante communauté d'utilisateurs « avancés », gage qui plus est de la qualité de ce logiciel dans le monde open source.

L'utilisation de Python. En dehors des raisons données ci-dessus, l'un des arguments décisifs dans le choix de QGIS a été la possibilité d'utilisation des fonctionnalités du logiciel à travers des scripts Python. Si uDig et gvSIG privilégient l'emploi du Java, QGIS est développé en C++ et est conçu depuis le départ pour que des programmes en C++, intégrés ou externes, puissent faire appel à ses fonctions. Rapidement, cette API a été étendue au langage Python, que l'on peut même exécuter dans QGIS grâce à une console inclue. Python est un langage simple d'accès, en ce que sa syntaxe est globalement explicite, et qu'il ressemble fortement au VBA étudié en cours. Bien intégré dans QGIS, il se montre rapide, simple à manipuler, et il est depuis longtemps très utilisé en géomatique, via l'appel à des bibliothèques telles que GDAL. Sa puissance, sa large base d'utilisateurs et sa simplicité l'ont d'ailleurs fait préférer, dans le monde propriétaire, au VBA dans la dernière mouture de la suite d'ESRI. Pour un étudiant novice en programmation, le défaut principal semblait être la gestion non intégrée des interfaces graphiques, simple en VBA mais nécessairement plus complexe dans les autres langages. QGIS est initialement développé en C++ et fait appel à la bibliothèque graphique

Qt. Dès lors, via son portage Python (PyQt), on peut toutefois assez simplement créer des interfaces graphiques de qualité, la partie la plus compliquée étant de les faire communiquer avec les scripts.
A travers l'appel aux bibliothèques PyQGIS et PyQt, le développement d'extensions pour QGIS se révèle donc assez simple, le plus dur étant, comme on l'avait déjà constaté en cours de VBA pour ArcGIS, de s'orienter parmi les centaines d'objets mis à disposition par l'API de QGIS.

La création d'un plugin QGIS. Une des grandes différences entre le développement d'un plugin QGIS et la mise en place d'une extension VBA dans ArcMap consiste en l'architecture complètement différente qu'il faut mettre en œuvre : sur ArcGIS, comme dans la suite Microsoft Office, l'environnement de développement est intégré et les extensions se présentent comme des macros dotées d'interface graphique. Ici, et au contraire de cette logique de script, les plugins QGIS sont indépendants du logiciel et même s'ils requièrent celui-ci pour fonctionner, peuvent interagir librement avec des librairies ou des programmes externes en python. Ces plugins viennent ensuite s'appuyer sur l'interface et les composants graphique usuelle. Un plugin QGIS est donc structuré comme pourrait l'être un projet indépendant, et demande donc le respect d'une architecture stricte pour pouvoir être compris par l'instance de QGIS qui l'appelle. Ce squelette peut être généré grâce à un plugin dédié [20]. Ce système de squelette a pour défaut de multiplier les classes même quand celles-ci n'ont pas de raison propre d'exister. De plus cette « relative »indépendance entre le plugin et QGIS rend plus difficile le débugage du programme, qui, appelé par QGIS, ne communique plus qu'avec celui-ci. La bibliothèque PyQGIS n'est pas nativement en Python, elle est compilée à la volée. Cela signifie qu'on ne peut l'importer dans un environnement externe, empêchant dès lors toute compréhension des classes par l'IDE choisi, et donc l'auto-complétion et la documentation des fonctions. Il n'y a alors aucun moyen d'explorer les classes et leurs usages sans recourir à la documentation en ligne [21], ce qui force à un recours préalable à cette documentation à chaque fois que l'on veut utiliser une nouvelle classe ou un de leurs attributs.
Sur un autre plan, le recours au framework Qt a été complexe, car il se base sur une logique de signaux inspiré d'un pattern « Observer »bien connu en informatique, mais un peu abstrait à appréhender pour un non informaticien. Ce mécanisme fait que certains événements peuvent émettre un signal plus ou moins permanent auquel des fonctions, dotées d'un mécanisme d'écoute qui filtre ces signaux, peuvent alors réagir.

La création d'un « outil » [22] reposant entièrement sur ce principe de signaux s'est donc révélée beaucoup plus longue et ardue que ce qu'aurait demandé l'équivalent dans un environnement plus habituel [23].

3.3.2 Outil de création et d'export de courbes de Bézier.

Les courbes de Bézier en représentation de flux.

Notre outil devant permettre la cartographie de tous types de flux, il ne pouvait faire l'impasse sur l'utilisation de courbes de Bézier pour représenter les flux. Celles-si sont très largement utilisées dans la production cartographique pour plusieurs raisons : En premier lieu, en comparaison des segments, elles apportent une impression de mouvement, impression

20. Plugin Builder
21. Que l'on peut consulter ici : `http://doc.qgis.org/head/`
22. Classe QgsMapTool
23. En VBA voir en Python sur ArcMap, qui n'a pas recourt aux signaux, hérités de la programmation en C et C++.

renforcée par l'ajout de pointes de flèches. De plus, leur forme de toute évidence stylisée permet de faire immédiatement sentir au lecteur que l'on a affaire à des flux immatériels, ou du moins, dont le cheminement exact n'est pas connu. Ils sont enfin très précieux en ce que leur malléabilité permet de les déformer autant que nécessaire pour qu'ils n'empiètent pas sur un élément important de la carte, ou encore, pour représenter des flux opposés, de mieux donner à voir la différence entre les arrivées et les départs entre deux lieux.
Traditionnellement, pour cartographier des flux de ce type, on exporte une carte dans un format vectoriel où les flux sont représentés par des segments, et on les retouche pour en faire des courbes de Bézier dans des logiciels de dessin vectoriel. Il était ici souhaitable que le plus gros de ces opérations puisse se faire au sein même du SIG, tant pour une question de rapidité qu'en ce que cela apportait comme information. En effet, pouvoir créer une mise en page plus lisible dans le SIG permet d'y retravailler directement en analysant des structures de flux non visibles sans ces courbes. L'intégration d'une fonction de création de courbes de Bézier dans le plugin visait donc temps à réduire le besoin d'utilisation d'outils de DAO qu'à rendre inutiles les aller-retour entre ces outils et le SIG.

Mise en place et utilisation.

le problème de l'affichage : segmentation A l'heure actuelle, aucun SIG ni même aucun SGBD spatial n'est capable de gérer les courbes de Bézier en tant qu'objets géométriques, celles-ci n'étant définies dans aucun système de stockage de géométrie.

Si ArcGIS et le SGBD PostGIS [24] peuvent stocker et effectuer des calculs sur des géométries de type arcs de cercles, la majorité des SIG ne supporte que les géométries linéaires [25]. Les courbes de Bézier n'étant pas définissables par le biais d'arcs, il fallait de toute manière travailler sur l'affichage. Au sein de QGIS, on a donc choisi de représenter les courbes comme une suite de segments, au nombre défini et paramétrable, segments cherchant à épouser au mieux la forme de la courbe. Notons que si ArcMap propose dans sa dernière version un outil de numérisation de courbes de Bézier, il ne fait au final que procéder comme nous, c'est-à-dire en segmentant, d'une manière très importante et non paramétrable dans son cas, les courbes de Bézier. Cette segmentation, si elle est nécessaire, pose pourtant de nombreux problèmes dès lors qu'elle n'est pas paramétrable, en ce qu'avec une multitude de flux, le nombre de segments devient rapidement important, entraînant une lenteur dans l'affichage et la manipulation.

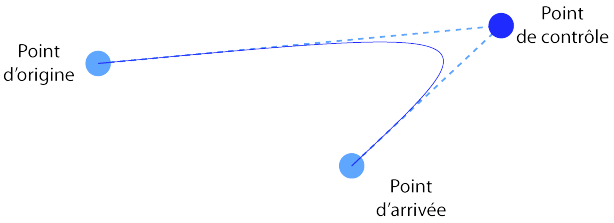

FIGURE 18 – Illustration d'une courbe de Bézier

24. Ce dernier en accord avec la norme SQLMM qui déclare des géométries de type « CircularString », voir : `http://postgis.refractions.net/docs/ch04.html#SQL_MM_Part3`

25. Pour des raisons de gestion de topologie : les calculs d'aire, d'intersection et d'inclusion devenant bien plus complexes quand les géométries ne sont plus polygonales. Ceci explique que si les courbes de Bézier sont des composantes indispensables des logiciels de DAO depuis des années, elles tardent à être implémentées dans les SIG en raison de la refonte complète des méthodes de calculs qui seraient requises.

Logique des points de contrôle. Une courbe de Bézier est défini par 3 points [26], qui correspondent respectivement au point de départ de la courbe, à un point de contrôle et au point d'arrivée (voir figure 18 page précédente). Il est important de noter que la courbe ne passe pas par le point de contrôle, celui-ci est donc un point d'attirance théorique qui permettra de tirer plus au moins loin vers lui, selon son écart aux points de départ et d'arrivée, la courbe. Comme la courbe ne passe pas par le point de contrôle, il est toujours compliqué pour l'utilisateur de prévoir la forme de la courbe de Bézier définit par les trois points choisis, il fallait donc impérativement que la courbe soit tracée au fur et à mesure du déplacement des points de contrôle, à la manière des logiciels de DAO habituels en cartographie. Un point important était de fait mis sur la convivialité de l'outil, lequel ne devait pas dépayser l'utilisateur plus habitué aux logiciels de DAO.

Modification individuelle et collective. En dehors de cette visualisation, on a essayé de prévoir les cas d'utilisation les plus fréquents et de faciliter le travail du cartographe.

Pour cela, on a pensé qu'en cas de large nombre de flux à convertir en courbes, il fallait que l'utilisateur puisse choisir d'agir sur un flux unique ou sur l'ensemble des flux représentés. En appliquant d'abord une conversion des flux linéaires en courbes de Bézier à l'ensemble des flux, on peut donc alors ajuster flux par flux, quand nécessaire, et ainsi gagner un temps précieux.

Export des flux. Même si le but du plugin est la simplification de la cartographie de flux par le rassemblement du maximum d'outils possibles au sein même du SIG, on ne peut penser qu'une carte juste sorti du SIG pourra répondre aux attentes d'un cartographe. Il fallait donc que les courbes de Bézier puissent être importées dans un logiciel de DAO. Si la segmentation utilisée pour l'affichage dans QGIS aurait été la méthode la plus simple, elle empêchait toute retouche des flux de manière vectorielle, puisque les courbes ne seraient plus interprétées comme des courbes mais comme des polylignes, et dès lors, ne pourraient voir leur forme changer qu'en modifiant tous les segments composant la courbe. On a donc choisi d'intégrer un outil d'export des courbes de Bézier. On utilise pour cela le format vectoriel SVG, qui supporte les courbes de Bézier. On peut dès lors exporter les flux depuis QGIS et les modifier sous Inkscape ou Illustrator.

3.3.3 Regroupement de flux selon la méthode FDEB.

Le clustering spatial

Les besoins. Le second cas d'utilisation qui posait le plus de problème en cartographie de flux, typique des sorties de SimpopLocal, est la trop forte densité de flux, fréquente dans l'analyse de réseaux issus de systèmes complexes : données de télécommunications, flux piétons, flux migratoires etc. Dans ce cas, si l'on souhaite conserver l'ensemble des flux et donc ne pas effectuer de sélections, il peut s'avérer presque impossible de réussir à distinguer un flux d'un autre, et de manière plus générale, de comprendre l'organisation des flux dans la carte. Pour répondre à cette problématique, il existe une méthode employée depuis a minima le XIX$^{\text{ème}}$ siècle, consistant à agréger les flux ayant une trajectoire proche. C'est le cas dans les très célèbres cartes de Minard, que ce soit la carte de l'avancée de l'armée napoléonienne pendant la campagne de Russie, ou celle des exportations de vin depuis la France. Si des cartes comme

26. Il peut y en avoir plus, mais nous nous sommes cantonnés aux courbes quadratiques, plus simples à gérer que ce soit pour la manipulation ou pour l'export en SVG, tout autant que pour l'utilisateur, la manipulation d'un point de contrôle avec affichage en temps réel étant plus aisée que celle de deux.

celles-ci ne peuvent être réalisées que manuellement et avec une forte quantité de travail, on peut utiliser la même logique qui consiste à regrouper les flux qui sont les plus similaires. Dès lors, si les flux suivent cette organisation, on sera capable de discerner des grands axes ou au moins des figures constituées par les flux, facilitant la tâche du modélisateur qui pourra s'appuyer sur cette base pour mener ses requêtes spatiales et attributaires.

Le clustering en cartographie. On a évoqué dans l'état de l'art l'outil jFlowMap, qui consiste en une implémentation d'une méthode de *bundling* (rassemblement) nommée FDEB (*Force-Directed Edge Bundling*) [HVW09]. Cette méthode vise à regrouper les flux selon les principes de l'attraction électro-magnétique, c'est-à-dire que plus deux flux seront proches, parallèles, de même taille et alignés, plus ils seront rassemblés dans leur parcours. Cette méthode n'est pas la seule existante, mais au contraire d'autres (comme GBEC, *Geometry-Based Edge Clustering* [CZQ+08]), elle n'agit que sur le cours des flux, et non sur leurs origines et destinations. Dans le cadre d'un travail cartographique tel que celui requit pour l'analyse des simulations, il est en effet indispensable que les flux relient bien deux villes distinctes, on ne peut donc regrouper les villes les plus proches comme GBEC le fait. La méthode FDEB a de plus l'important avantage d'avoir été implémentée dans jFlowMap, outil dont les sources sont accessibles, et dès lors, on pouvait partir d'une base existante pour mettre en place cet algorithme sous forme de plugin Python dans QGIS.

Mise en place de FDEB dans QGIS.

Logique de l'algorithme. FDEB fonctionne sur une logique de subdivision des lignes représentant les flux. On débute donc avec des flux simples, composés d'un segment entre le point d'origine et la destination. A chaque cycle, on subdivise le segment originel d'un nouveau niveau. Au premier cycle, le flux sera donc composé de deux tronçons, et au second, de trois. Dans chacun de ces cycles se déroulent un certain nombre d'itérations. Durant chacune de ces itérations, le ou les points de subdivision de chaque segment se déplacent d'une très faible distance en direction des points des segments les plus compatibles. On définit cette compatibilité par quatre indicateurs (illustrés dans la figure 19 page suivante) que l'on a brièvement mentionné plus haut, lesquels varient tous de 0 à 1 :

1. *La compatibilité d'angle - C_a :* Plus les deux segments examinés approchent de la situation parallèle, plus C_a tend vers 1. A contrario, deux segments perpendiculaires ont un C_a de 0.
2. *La compatibilité d'échelle - C_s :* On considère que deux segments ayant une longueur proche sont plus compatibles que deux segments de longueur très inégale. On a donc C_s qui vaut 1 si les deux segments examinés sont de même longueur, et vaut 0 si le rapport de longueur tend vers l'infini.
3. *La compatibilité de position - C_p :* Comme dans toute attraction de type gravitaire, la distance entre deux segments joue aussi un rôle important. Si les deux flux sont confondus, C_p vaudra 1, et tendra vers 0 à mesure que la distance entre les deux augmentera.
4. *La compatibilité de visibilité - C_v :* Cette mesure n'est calculée que si les précédentes résultent en une forte compatibilité. Dans l'espace plan qui caractérise les flux dans l'algorithme, il faut aussi que ceux-ci soient alignés, c'est à dire que leur décalage soit le plus faible possible. Dans ce cas, C_v tendra vers 1.

La compatibilité de deux flux résulte du produit de ces quatre indicateurs, et fluctue donc entre 0 et 1 elle aussi.
A chaque cycle, la distance parcourue par les points de subdivision à chaque itération diminue,

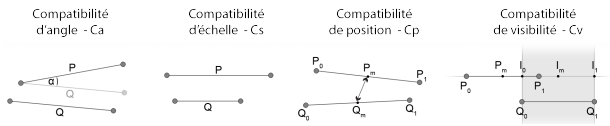

FIGURE 19 – Description des indicateurs de compatibilité, D. Holten et Jarke J. van Wijk

et le nombre d'itérations diminue lui aussi. On obtient donc des flux composés d'autant de segments qu'il y a eu de cycles, dont les points de subdivision les plus proches extrémités sont plus éloignés des segments compatibles. L'apparence est donc celle d'une courbe régulière.

Développement et problèmes rencontrés. On a dit que l'on s'était basé sur jFlowMap. la première difficulté a consisté à comprendre la logique de l'implémentation effectuée dans ce logiciel. Celui-ci étant en Java, et usant d'un grand nombre de classes propres destinées au stockage des géométries, à l'affichage des flux et autres nécessités, il a fallu passer un temps non négligeable pour décortiquer l'algorithme, et ensuite le retranscrire de manière non fonctionnelle mais en Python dans le plugin. Notons tout de même que les sources de jFlowMap présentent une absence quasi-totale de commentaires, lesquels auraient pu grandement accélérer la compréhension du code.
Une fois cette phase de traduction effectuée, il a fallu adapter les logiques, classes et calculs géométriques aux spécificités de QGIS, c'est à dire en utilisant autant que possible les objets PyQGIS, mais aussi en adaptant les entrées de l'algorithme aux données obtenues dans le plugin. De manière plus générale, on a du ajouter des éléments de paramétrage à jFlowMap pour pouvoir accepter des cas d'utilisation non prévus. Ainsi, le programme n'accepte en entrée que des flux avec des coordonnées Latitude/Longitude. Si le travail de conversion des coordonnées est effectué par QGIS, les différences d'ordre de grandeur entre des coordonnées Lat Long (de -90^o à 90^o) et par exemple des coordonnées en Lambert93 (de 1,5e5m à 1e6m en longitude et de 6e6m à 7e6m en latitude pour la France Métropolitaine) implique un travail supplémentaire sur la définition des pas de distance à parcourir à chaque itération. Ne trouvant de formule convenant à tous les cas essayés, on a donc du laisser ce réglage au choix de l'utilisateur.
La plus grande difficulté est pourtant inhérente à l'algorithme, et nous a autant handicapé qu'elle handicapera les utilisateurs du plugin. Il s'agit du temps requis pour exécuter FDEB. Si ce traitement est quasi instantané sur quelques flux, sur plusieurs milliers, il peut demander des dizaines de minutes, voire des heures de calcul, et dès lors, l'utilisateur ne pourra le lancer de multiples fois avec des paramétrages différents pour essayer de faire ressortir des informations de ses données. Si l'implémentation peut sans doute être optimisée sur de multiples aspects, il n'en reste pas moins que le temps de calcul augmente de manière plus que proportionnelle au nombre de flux. Sauf à avoir un ordinateur très puissant et beaucoup de temps devant soi, il est donc peu concevable que cette fonction soit appliquée à des données comprenant des dizaines de milliers de flux.

Paramétrage et résultats graphiques Lorsque l'utilisateur souhaite appliquer FDEB à ses flux, il doit renseigner quatre paramètres, lesquels sont décrits dans l'onglet d'aide (voir l'interface utilisateur, figure 20 page suivante). Ces paramètres sont :

1. *Le seuil de compatibilité.* Afin d'accélérer le traitement et de schématiser le rendu cartographique, on peut choisir de fixer un seuil de compatibilité. Si deux segments n'atteignent pas ce seuil, alors ils ne s'attireront pas. Seuls les plus fortes compatibilités joueront alors dans la déformation des flux, ce qui peut permettre de résumer l'informa-

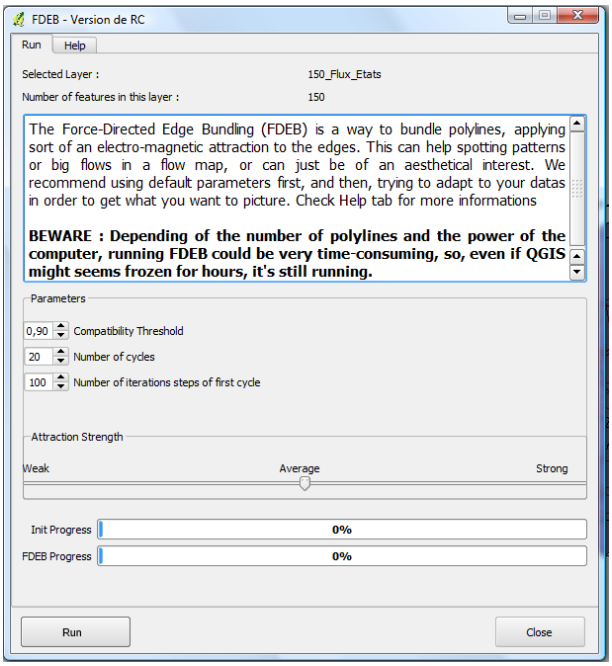

FIGURE 20 – Fenêtre de configuration et de lancement de FDEB.

tion mais aussi en faire disparaître des composantes importantes. On peut partir d'un seuil élevé, et le diminuer jusqu'à voir apparaître des figures dans l'organisation des flux.

2. *Le nombre de cycles.* On a vu que ce nombre correspondait au nombre de subdivision final des flux. Plus il sera élevé, plus les flux auront une forme curviligne lissée. Pour autant chaque cycle ajoute un temps de calcul important, il faut donc trouver un compromis pour l'utilisateur dès lors que les données deviennent trop importantes.
3. *Le nombre d'itérations initiales.* À chaque itération, les points de subdivision sont attirés par les segments compatibles. Augmenter le nombre d'itération provoquera un plus large regroupement des flux, tandis qu'en le diminuant, les flux auront moins tendance à s'agréger. Sachant que ce nombre est divisé par deux à chaque cycle, rien ne sert pour autant de trop le réduire, sous peine d'avoir des cycles de calcul où rien ne se passera.
4. *La force d'attractivité.* C'est un paramètre non présent dans jFlowMap, que l'on a du ajouter pour les raisons citées peu avant. Il est volontairement peu précis car devant être adapté à chaque jeu de données. Plus il sera élevé (curseur vers la droite), plus la distance parcourue par les points de subdivision, dépendante de la compatibilité des segments qui les attire, sera importante. Un exemple d'application de FDEB à des données migratoires est présenté en figure 21 page 50.

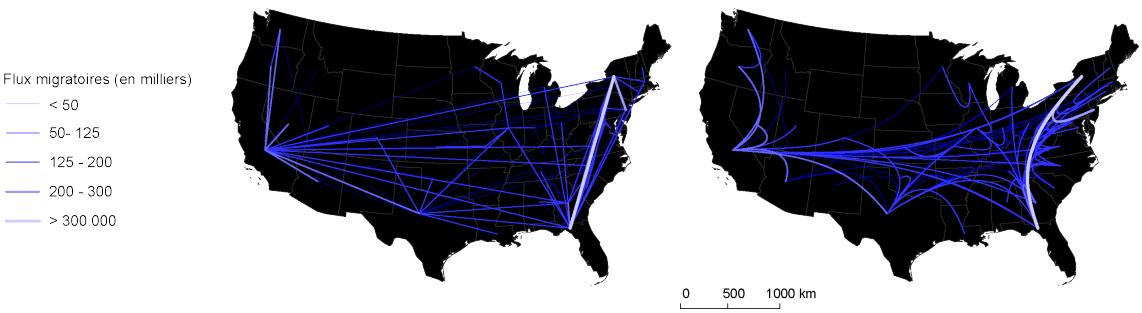

FIGURE 21 – Un exemple de carte générée avec FDEB.

3.3.4 Continuer le développement.

qFlowMapper n'est pas achevé, et pour qu'il soit aussi convivial que souhaité, il lui manque encore plusieurs fonctions.

Création des flux. C'était une des volonté premières du plugin, qui finalement a été repoussée à plus tard, l'implémentation de l'algorithme FDEB ayant pris la priorité au cours du développement.. Si un travail de recensement des types de données pouvant nécessiter une conversion en flux géométriques a été effectué la gestion des cas simples (table origine-destination avec shapefile ponctuel ou polygonal, ou table origine-destination avec les coordonnées des origines et destinations) requiert un développement minime qui n'a pas eu le temps d'être effectué. Cet import de données devra constituer une priorité dans le développement futur du plugin.

Gestion des sorties. Pour la suite du développement, il est indispensable que les résultats des traitements puissent être exportés dans différents formats. Si l'on travaille nativement avec des fichiers shapefiles pour la manipulation ultérieure des données et avec du SVG pour la réalisation cartographique, on pourrait de même exporter les flux générés dans des formats moins spécifiques, à l'instar du KML, qui, additionné d'une dimension z (altitude), permettrait d'augmenter la lisibilité des flux dans un visualisateur tel que Google Earth, et éviterait la confusion possible sur la fausse inscription des flux dans l'espace, en les en détachant.
Il faudrait aussi que les traitements effectués sur les flux soient ré-utilisables simplement dans le plugin, notamment les courbes de Bézier créées. Pour cela, on modifierait l'export des SVG en y ajoutant des informations sur les coordonnées X/Y des points de départ, de contrôle et d'arrivée, permettant de reprendre le travail de modification des flux là où l'utilisateur s'était arrêté. L'import d'un fichier SVG de ce type serait alors permis par le plugin, qui le convertirai en couche géométrique comme cela se passe actuellement après conversion d'un fichier linéaire en courbes de Bézier.

Flux polarisés. Si la représentation de ce type de flux sous forme d'oursins est fréquente, on a cherché à améliorer ce mode de visualisation pour le clarifier, en particulier quand les flux polarisés sont trop nombreux ou denses pour qu'on puisse simplement définir leur destination. Pour cela, nous avions considéré l'idée d'implémenter un algorithme de « circularisation »des flux, lequel courberait les flux polarisés selon des « grands cercles », les flux étant dès lors séparés les uns des autres. L'algorithme choisi [27] est bien expliqué, son implémentation devrait donc pouvoir se faire sans grande difficulté.

27. Visible à l'adresse suivante : `http://gis.stackexchange.com/questions/5204/curved-point-to-point-route-maps/5224#5224`

Quatrième partie

Conclusion : limites et perspectives.

4.1 Travail effectué et objectifs initiaux.

4.1.1 Les modèles.

Sur ce point, le sujet du stage proposait la participation à la création d'un modèle, SimpopNet, modèle qui dans son ensemble devait être fonctionnel à la fin du stage. Au lieu de ça, nous avons enrichi un modèle pré-existant (SimpopLocal) et conçu et implémenté un premier prototype de SimpopNet : SimpopNet-Réseaux.

SimpopLocal et le module de catastrophes. Non présent dans les objectifs initiaux, la mise en place d'un mécanisme de catastrophes dans SimpopLocal a permis de découvrir plusieurs domaines. Cette étape était cruciale pour pouvoir se lancer dans la création de SimpopNet, en ce que certains mécanismes en sont tirés. Sur un plan plus théorique, le volet « formation »à NetLogo introduit par le biais du « modèle de collaboration »de chercheurs n'apportait pas le cadre théorique nécessaire à l'étude des systèmes de peuplements (et de villes pour SimpopNet). La compréhension des faits stylisés de SimpopLocal, étape nécessaire à l'ajout du module de catastrophes, a constitué un premier pas dans cette partie de la recherche en géographie fondée sur l'étude des interactions spatiales au sein d'un système complexe. Sur un plan technique, les besoins de calibration des catastrophes ont introduit la nécessité de production de rapports appropriés. Ces rapports ont été créés sous R et donc constitué une première approche de ce logiciel de statistiques, de même qu'une réflexion sur le choix des indicateurs, préalable à la création de TrajPop.

SimpopNet-Réseaux. On pourrait regretter que le stage n'ait pas abouti à la livraison d'un modèle SimpopNet complet, et il convient déjà d'en comprendre les raisons. Sur l'aspect le plus visible, la conception d'un modèle de ce type demande nécessairement plus de temps que celui qui était disponible, en considérant que les cinq mois de stage ne se cantonnaient pas à la réalisation de SimpopNet. L'importance du temps nécessaire est expliquée par le contexte de la création de ce modèle : SimpopNet n'a pas été entièrement conçu avant le stage, les mécanismes à mettre en place mais aussi le rôle de ce modèle ont été et sont encore en redéfinition constante, en ce que ce modèle rentre dans le cadre de la thèse de Clara Schmitt, et donc de recherches sur un temps plus long que celui du stage. Une fois la décision prise de diviser SimpopNet en trois sous-modèles conçus comme des prototypes indépendants, il a été décidé que le stage porterait sur la partie qui vise à la création et à l'évolution d'un réseau dynamique entre des villes. Ce choix a été motivé par l'intérêt montré pour ces problématiques de réseau tout autant que par la nouveauté que ces questions constituaient pour Clara Schmitt.

4.1.2 Autour des modèles - Pilotage et analyse de résultats.

La dernière partie du stage devait être consacrée à « la conception et l'implémentation de plugins SIG pour piloter l'expérimentation en s'appuyant sur l'architecture existante (définition et réutilisabilité des plans d'expérience, personnalisation des rapports d'exécution par choix des modules), pour visualiser des données de flux ». Assez rapidement dans le cours du stage, la partie consacrée au pilotage des simulations a été abandonnée, en ce qu'elle a finalement été jugée postérieure au besoin de production de rapports de simulations. Ainsi, une plate-forme de pilotage sera réalisée dans le futur, mais elle devra s'appuyer sur SimProcess et ne pourra être conçue dans son ensemble qu'une fois sa conception et son implémentation définies. L'idée de conserver le SIG comme application centralisatrice est par contre restée.

qFlowMapper. Partant du constat que les solutions existantes étaient tout au moins insuffisantes pour représenter et analyser les flux, il a fallu définir précisément les besoins des modélisateurs. Si la création d'un outil englobant l'ensemble des cas d'utilisation théoriques aurait elle aussi nécessité que le stage lui soit uniquement consacré, on a choisi de pallier les problèmes pour lesquels il n'existait pas de solution un tant soit peu satisfaisante. Si la création d'un outil de conversion et de modification des flux en courbes de Bézier répondait à l'une des principales requêtes, on a ensuite choisi de mettre en place des méthodes de traitement plus complexes, dont un échantillon est représenté ici par l'implémentation de l'algorithme FDEB. Ce plugin de représentation des flux s'inscrit ainsi dans un cadre dépassant celui du stage, et doit permettre de doter le laboratoire mais aussi la communauté d'utilisateurs de QGIS d'un outil à même de créer, traiter, et visualiser des flux. Dès le départ, il a été convenu que la durée du stage ne permettrait pas d'en obtenir un outil à tout faire, et que son prolongement se poursuivrait ainsi après le stage, dans le cadre de la communauté de développeurs de plugin QGIS, par plaisir et en tant que projet personnel [28].

TrajPop. Si le besoin de création de rapports automatisés était présent dès le départ, il n'était pourtant pas prévu à ce moment là qu'il aboutisse à la création de TrajPop dans le cadre du stage. C'est donc d'un concours de circonstances qu'a résulté la décision d'inclure sa conception. En premier lieu, on a assisté à une journée du colloque INTERACTIONS 2011 [29] où Michael Batty présentait l'outil *RankClock* conçu par son laboratoire, le CASA. Lorsqu'il a fallu analyser les effets des catastrophes dans SimpopLocal, on a donc pensé à utiliser les *RankClocks* comme moyen efficace de visualisation des bouleversements dans la hiérarchie des systèmes de peuplement (d'autant plus que le potentiel d'utilisation de cet outil intéressait d'autres chercheurs du laboratoire). On s'est alors rendu compte des manquements explicités (voir 3.2.1 page 33), ce qui a entrainé la conception de TrajPop.

4.2 Acquis et débouchés.

Si les objectifs initiaux du stage ont été revus et adaptés de manière régulière, cela a trait principalement au cadre de stage. Bien loin d'un stage de production, la forte place accordée à la prospection technologique et théorique a permis de ne pas se contenter d'appliquer des techniques apprises en cours, l'une des caractéristiques de ces cinq mois ayant été la découverte de nouveaux domaines, méthodes et outils.

4.2.1 La découverte du monde de la recherche.

Le secteur de la recherche. Le travail dans les locaux d'un laboratoire de recherche a été enrichissant en lui-même par le climat d'entraide et d'échange. Le bon fonctionnement du laboratoire a d'ailleurs été remarqué par l'AERES lors de sa dernière évaluation, notant que « les experts ont été très favorablement impressionnés par l'esprit démocratique et la convivialité qui règlent les rapports humains entre tous les membres de l'UMR, jeunes doctorants compris. C'est sans doute une cause et une conséquence de la réussite de l'UMR ».

La modélisation et les SMA. Si Clara Schmitt et Sébastien Rey-Coyrehourcq nous avaient présenté leurs travaux dans le cadre d'un séminaire commun des masters Carthagéo

28. Quiconque peut y participer, les sources étant librement disponibles dans un dépôt Git situé à cette adresse : `https://github.com/RCura/PluginFlux`

29. Pour plus de détails, consulter le site du colloque : `http://samm.univ-paris1.fr/INTERACTIONS-2011-Mathematical`.

et Géoprisme, le champ de la modélisation informatique nous était encore largement inconnu. Ancrée dans plusieurs de ses axes, le stage a permis de découvrir en profondeur ce domaine d'étude, tout autant que d'en saisir les enjeux généraux par le contact avec des modélisateurs d'autres sciences, dans les locaux de l'ISC-PIF en particulier.

4.2.2 Une formation permanente.

Formation théorique. Sur le plan théorique, chaque étape du stage a été l'occasion de découvrir ou de renforcer des connaissances, sur des domaines aussi vastes que la théorie des graphes pour SimpopNet-Réseaux, les mécanismes de croissance tels que celui de Gibrat, ou encore la mise en application de techniques d'analyse statistiques dans TrajPop. Outre ces formations rendues nécessaires par le besoin de production d'outils et de modèles, le laboratoire a aussi encouragé la participation à des colloques, salons et groupes de travail, tout autant qu'à des réunions parfois pluri-disciplinaires, visant à présenter les domaines d'études de chercheurs extérieurs et du laboratoire, ou encore à renforcer des connaissances élémentaires.

L'apprentissage de nouveaux outils. Le stage a également permis de découvrir de nouvelles technologies dont la connaissance apporte un plus important au futur professionnel d'un géomaticien. En dehors de NetLogo dont l'utilisation semble pour l'heure cantonnée au monde de la recherche, j'ai appris à utiliser le logiciel R, le langage Python et son utilisation au sein de l'API de QGIS (PyQGIS) ainsi que le framework graphique Qt (PyQt). Le stage a aussi été l'occasion de constituer un environnement de travail dans un système GNU/Linux, via l'utilisation de logiciels tels que RStudio, Spyder et Qt Designer, Inkscape ou encore de l'outil de versionnement Git, fortement utilisé dans chacune des phases de développement. En fin de stage, la rédaction de ce rapport a permis de découvrir et d'apprendre l'utilisation du langage de production de document LaTeX.

Veille technologique. Poussé par les encadrants du stage, j'ai aussi assuré une veille technologique constante, dont les apports, s'ils n'ont pas forcément été directs, ouvrent pourtant des pistes pour les développements futurs. Dans ce cadre, nous avons régulièrement suivi les avancées des technologies employées, par la participation aux communautés d'utilisateurs et de développeurs de NetLogo, QGIS et R (lecture et participation aux mailing-list, signalement et participation à la correction de bugs), mais aussi par le suivi de technologies émergentes. Parmi ces technologies suivies, nous consacré un temps important aux tentatives d'utilisation de systèmes de gestion de bases de données de type NoSQL, en particulier autour du SGBD Neo4j, spécialisé dans le stockage et l'interrogation de bases de données de graphes, et doté d'un module spatial (Neo4j Spatial) couplable avec des fournisseurs de services tels que GeoServer. Pour les modèles existants, l'utilisation de bases de données non relationnelles auraient l'avantage de la séparation des données et du modèle, permettant de décentraliser la constitution des bases de données résultats sur chacun des nœuds de calcul d'une grille effectuant les simulations [30]. Le stockage sous forme NoSQL est aussi plus léger, et les requêtes, mêmes spatiales, plus rapides que dans les SGBD clients-serveur traditionnels.

30. Actuellement, dans le cas d'exploration massive de données, on est obligé d'agréger les données. Les systèmes des base de données embarquées apportent aussi des fonctions intégrées, par exemple de résolution de plus courts chemins, qui seraient utiles car complémentaires à NetLogo et pouvant accélérer le déroulement des simulations.

4.2.3 Continuer les travaux.

Enfin, il nous paraissait important de noter qu'en raison de l'expertise apportée dans les sujets traités et des connaissances acquises durant le stage, une suite à celui-ci a été acceptée, sous la forme d'une vacation pour l'UMR Géographie-cités, au sein du projet de recherche GeoDiverCity, sous la direction de Denise Pumain. Tel que définies à l'heure actuelle, les missions de cet emploi devraient concerner l'enrichissement de TrajPop, que ce soit par l'amélioration de cet outil ou par son utilisation comme base d'un nouvel outil d'analyses plus poussées des trajectoires de population des villes. Ce projet serait encadré par Denise Pumain. De plus, je devrais aussi contribuer au modèle SimpopNano avec Thomas Louail, en menant des simulations adaptées à l'évolution de villes réelles et en confrontant leurs résultats avec les croissances constatées.

Bibliographie

[AB02] R. Albert and A.L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1) :47, 2002.

[Art09] W.B. Arthur. *The nature of technology : What it is and how it evolves*. Free Pr, 2009.

[BGPM+96] Stéphane Bura, France Guérin-Pace, Hélène Mathian, Denise Pumain, and Lena Sanders. Multi-agents system and the dynamics of a settlement system. *Geographical Analysis*, 28 :161–178, 1996.

[Chr33] Walter Christaller. *Die zentralen Orte in Süddeutschland : eine ökonomisch-geographische Untersuchung über die Gesetzmässigkeit der Verbreitung und Entwicklung der Siedlungen mit städtischen Funktionen*. Iena, 1933.

[CZQ+08] W. Cui, H. Zhou, H. Qu, P.C. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, pages 1277–1284, 2008.

[DDC] S. Dray, A.B. Dufour, and D. Chessel. The ade4 package—ii : Two-table and k-table methods. *New Functions for Multivariate Analysis*, page 47.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1) :269–271, 1959.

[Gib31] Robert Gibrat. *Les Inégalités économiques, applications : aux inégalités des richesses, à la concentration des entreprises, aux populations des villes, aux statistiques des familles, etc. : d'une loi nouvelle, la loi de l'effet proportionnel*. Librairie du Recueil Sirey, 1931.

[HVW09] D. Holten and J.J. Van Wijk. Force-directed edge bundling for graph visualization. 28(3) :983–990, 2009.

[Lou10] Thomas Louail. *Comparer les morphogenèses urbaines en Europe et aux Etats-Unis par la simulation à base d'agents. Approches multi-niveaux et environnements de simulation spatiale*. Thèse de doctorat, Université d'Evry Val d'Essonne, 2010.

[Pen10] Pennesi. Conquering by copying. *Science*, 328(5975) :165, 2010.

[PSJS89] Denise Pumain, Thérèse Saint-Julien, and Lena Sanders. *Villes et auto-organisation*. Economica, Paris, 1989.

[TW04] Seth Tisue and Uri Wilensky. Netlogo : A simple environment for modeling complexity. In *in International Conference on Complex Systems*, pages 16–21, 2004.

[Ver38] P.F. Verhulst. Notice sur la loi que la population suit dans son accroissement. *Corr. Math. et Phys*, 10 :113–121, 1838.

Résumé

Les travaux présentés dans ce manuscrit ont été effectués au cours d'un stage d'ingénieur géomaticien de 6 mois dans l'équipe PARIS du laboratoire Géographie-Cités. Nous présentons dans une première partie les réalisations en lien avec des modèles de simulation spatiale, *SimpopLocal* et *SimpopNet*. Ce travail inclut la conception et le développement de mécanismes pour enrichir ces modèles et permettre d'aborder de nouvelles questions, ainsi que la réalisation d'un outil de visualisation de trajectoires de villes, *TrajPop*. Dans la seconde partie, nous présentons les travaux réalisés pour améliorer la cartographie des flux sous SIG. Nous avons développé un *plugin* pour le SIG libre QGIS, *qFlowMapper*, en repartant d'un algorithme efficace de la littérature en visualisation.

Mots-clefs

Modélisation et simulation à base d'agents, NetLogo, programmation d'un plugin SIG, Quantum-GIS, cartographie de flux, outil de visualisation des trajectoires de villes.

Abstract

This memoir presents some engineering problematics and achievements that have been tackled during a geomatics internship of 6 months in the PARIS team of Geographie-Cités research lab. In the first part, we present our realisations in relation with two geographical simulation models, *SimpopLocal* and *SimpopNet*. Our work includes the design and implementation of spatial mechanisms in order to enhance these two models, as well as *TrajPop*, a visualization tool that allows to map cities' trajectories through time. In the second part we present some realisations motivated by the wish to enhance flow mapping in the GIS. We present *qFlowMapper*, a QGIS plugin based on an efficient algorithm of the visualization community.

Keywords

Agent-based modeling, Netlogo, GIS plugin development, Quantum-GIS, flow mapping, tool for citie's trajectories visualization.