Articulation d'échelles en simulation de mobilité

Xavier Boulet

UNIVERSITÉ PARIS-EST

Pôle de recherche et d'enseignement supérieur

École doctorale MSTIC

Articulation d'échelles en simulation de mobilité

THESE

Présentée pour l'obtention du titre de

DOCTEUR EN INFORMATIQUE

Par

Xavier Boulet

JURY

Directeurs de thèse :	**M. Gérard Scemama** Directeur de recherche émérite à l'université Gustave Eiffel
	M. Fabien Leurent Professeur à l'école des Ponts ParisTech
Encadrant :	**M. Mahdi Zargayouna** Chargé de recherche HDR à l'université Gustave Eiffel
Rapporteurs :	**M. Flavien Balbo** Professeur à Mines Saint-Etienne
	M. Fabien Michel Maître de conférence HDR à l'Université de Montpellier
Examinateurs :	**Mme Chirine Ghédira** Professeur à l'Université Lyon III
	Mme Aurore Rémy Directrice à AIMSUN

Table des matières

Table des figures

Chapitre 1

Introduction générale

Sommaire

1.1 Contexte général

1.1.1 La gestion des transports et les modèles de trafic

Les opérateurs de transport multimodal sont responsables de la régulation du trafic, de la définition des limites de vitesse ou des tableaux de marche des transports en commun.

Ils ont besoin d'études de transport leur permettant de comprendre le fonctionnement des réseaux de transport et de prévoir les conséquences des modifications qu'ils souhaitent mettre en place. Parmi les méthodes les plus populaires pour réaliser une étude de transport figure le modèle à 4 étapes [McNally2000]. D'autres modèles existent, tels que les modèles à base d'activités [Castiglione *et al.*2015], mais le modèle à 4 étapes permet de définir les éléments essentiels qui sont utilisés tout au long de cette thèse. Ce modèle est constitué de quatre étapes que nous pouvons résumer ainsi :

1. La génération des déplacements : le réseau de transport est découpé en plusieurs zones qui correspondent aux possibles origines et destinations dont partent et arrivent les voyageurs. Cette étape consiste à déterminer combien de voyageurs entrent et sortent de chacune de ces zones pendant la période de temps étudiée.
2. La distribution des déplacements : cette étape consiste à faire le lien entre ces origines et ces destinations pour chaque voyageur. La première étape consistait uniquement à trouver le nombre de voyageurs entrant et sortant dans chaque zone, lors de cette nouvelle étape on détermine le nombre de voyageurs qui partent d'une origine et arrivent à une destination afin de générer les matrices origine-destination.
3. Le choix de mode consiste à déterminer quel mode de transport les voyageurs vont utiliser. Le réseau de transport est composé de plusieurs réseaux différents (transports publics, réseau routier, réseau piétons, etc.) avec des points de liaison qui permettent aux voyageurs de passer d'un réseau à un autre, ces points de liaisons peuvent par exemple être une gare pour passer du réseau piéton au réseau de transports en commun ou un parking pour passer du réseau routier au réseau piéton. Cette étape consiste à savoir quels modes de transport les voyageurs vont utiliser durant leurs trajets.
4. L'affectation des voyageurs est la dernière étape et consiste à trouver le chemin emprunté par les voyageurs. Grâce aux étapes précédentes, on sait déjà d'où partent, où vont les voyageurs, quels modes de transport ils utilisent et donc sur quels réseaux ils vont se déplacer. Cette étape permet de déterminer les chemins précis empruntés dans le réseau de transport.

L'affectation - et parfois le choix de mode - est l'étape durant laquelle la simulation intervient. L'affectation peut être statique ou dynamique, c'est à dire prendre en compte (dynamique) ou non (statique) de la dimension temporelle, renseignant l'évolution de l'état du réseau de transport au cours du temps. Le nombre de voyageurs sur les différents arcs du réseau augmente le temps de parcours de ces arcs. D'autres phénomènes peuvent également perturber ce temps de parcours comme les accidents, les bouchons etc.

Dans les cas d'affectation dynamique, on utilise généralement la simulation qui permet de représenter l'évolution l'état du réseau au cours du temps.

1.1.2 La modélisation et la simulation

La simulation numérique des déplacements des voyageurs peut servir à observer en temps réel l'état du réseau de transport, mais également à prédire des situations futures, à tester différents scénarios et différentes solutions en étudiant leur impact sur la circulation des voyageurs sans impacter le trafic réel. Les opérateurs et les planificateurs de transport se servent des simulations comme systèmes d'aide à la décision pour pouvoir planifier la régulation, les limites de vitesse, la taille de la flotte de véhicules de transport en commun, leurs horaires de passage aux arrêts, etc.

Chaque simulation est fondée sur un modèle de déplacement de voyageurs (cf. chapitre 2). Ces modèles utilisent généralement des équations permettant d'exprimer la vitesse, la position et la densité de voyageurs au cours du temps. La simulation discrétise ce temps et fait avancer le modèle par pas de temps discrets. La simulation correspond donc à l'évolution du modèle de déplacement au cours du temps à intervalles réguliers.

Parmi les modèles de simulations de mobilité figure le paradigme multi-agents. Le paradigme multi-agents fournit un niveau élevé de détails et permet de représenter des phénomènes et des modèles non linéaires qu'il serait difficile de modéliser avec des approches analytiques [Bonabeau2002]. En l'occurrence, la modélisation multi-agents permet de représenter les comportements humains et les interactions des voyageurs avec un environnement complexe et dynamique. La programmation multi-agents peut être décrite comme une branche de la programmation orientée objet, dans laquelle on retrouve au minimum des objets « agent »et un objet « contexte », les agents interagissent non seulement avec le contexte mais également avec les autres agents à proximité. Par définition, les agents disposent d'informations limitées sur le reste de la simulation. Cette approche est adaptée à la simulation de voyageurs, qui sont représentés par les agents, et qui évoluent dans un contexte qui est le réseau de transport.

D'autres paradigmes de simulation existent, comme la modélisation macroscopique, dans laquelle les voyageurs ne sont pas représentés individuellement mais où chaque partie du réseau possède une vitesse moyenne, une densité moyenne des flux de voyageurs. Dans la représentation en flux, les voyageurs sont donc semblables à un liquide qui s'écoule dans le réseau hydraulique. Les différences entre les différentes manières de représenter

les voyageurs sont étudiées plus en détails dans le chapitre 2.

Le réseau de transport peut également être représenté de plusieurs manières. La manière la plus simple de le modéliser est de le représenter comme un graphe - au sens mathématique - composé de nœuds et d'arcs. Les arcs correspondent aux routes et aux lignes de transport en commun alors que les nœuds représentent des intersections, des arrêts ou des points de liaisons entre les différents réseaux de transport. Les arcs possèdent des caractéristiques comme la longueur, la capacité, la vitesse maximale et peuvent également posséder plusieurs voies. Le réseau de transport peut également être représenté à l'aide de polylignes à la place d'arcs simples pour mieux correspondre au réseau réel de transport. Dans les modèles les plus détaillés, les arcs peuvent même être représentés en deux dimensions pour prendre en compte le déplacement latéral des voyageurs.

Un simulateur est donc composé de voyageurs avec leurs origines et leurs destinations, d'un réseau de transport et d'un modèle de déplacement permettant de représenter l'avancement des voyageurs dans le réseau à chaque pas de temps. On représente un simulateur par la Figure 1.1 : les matrices origine-destination (OD) et le réseau de transport sont des entrées du simulateur. Le modèle de déplacement et le pas de temps sont au cœur du simulateur. Tout au long de la simulation, l'utilisateur peut observer des indicateurs de congestion, de vitesse moyenne ou toute autre indicateur qui l'intéresse. Ces indicateurs correspondent donc aux sorties du simulateur.

Il existe de nombreux simulateurs de déplacements différents. Afin de déterminer le meilleur simulateur à utiliser, l'utilisateur doit tout d'abord déterminer ses besoins précis : quels modes de transport souhaite-t-il simuler ? Quelle zone est à simuler ? Quels phénomènes doivent être observés ? Quelles sont les données disponibles ? Quel niveau de détails est nécessaire ? Cette dernière question est une question centrale et permet d'introduire un des concepts clés de la thèse : Les échelles de simulation. D'où la question centrale suivante : qu'est ce qu'une échelle de simulation et quelle est l'échelle de simulation la plus appropriée à la problématique traitée ?

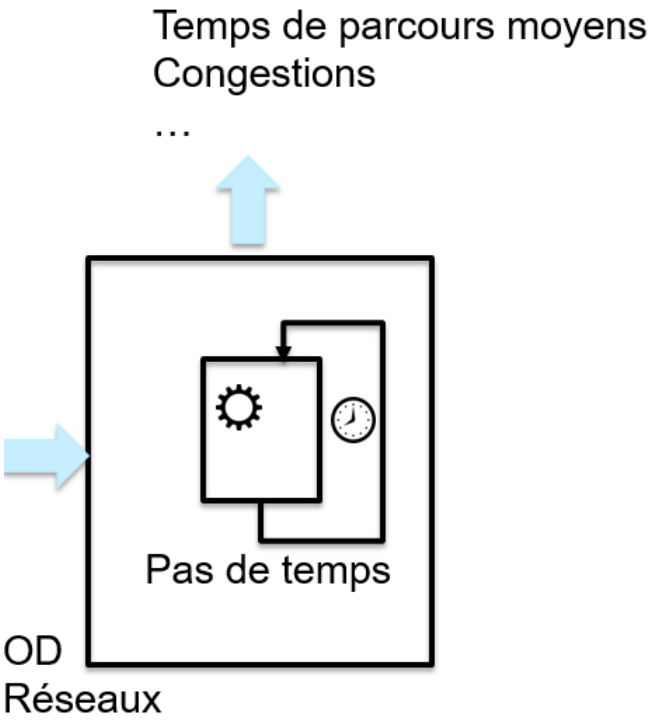

FIGURE 1.1 – Fonctionnement d'un simulateur

1.2 Problématique : articuler des échelles de représentation

1.2.1 Échelles microscopiques et macroscopiques

Dans de nombreux domaines, l'étude d'un objet peut se faire à des échelles différentes. On peut l'étudier de manière très détaillée et chercher à représenter précisément tous les éléments qui le constituent, en analysant leurs relations avec les autres éléments. On peut également chercher à prendre du recul, à avoir une vision plus globale avec beaucoup moins de détails en considérant les objets du système de manière agrégée. Chaque approche possède des avantages et des inconvénients, et permet d'observer des phénomènes différents. Si on étudie du gaz de manière détaillée par exemple, on s'intéressera aux phénomènes de collisions entre molécules alors que l'étude du gaz comme un agrégat utilisera des variables comme la densité ou le débit de ce gaz.

De la même manière, le fonctionnement d'un réseau de transport met en jeu des phénomènes à des échelles très différentes. La prise en compte de tous ces phénomènes

dans les simulations numériques demanderait d'utiliser des modèles extrêmement détaillés sur l'ensemble du réseau, entraînant des coûts de calcul trop importants. Ainsi, les modèles macroscopiques permettent de simuler une large zone pour un temps de calcul raisonnable alors que les modèles microscopiques sont plus souvent utilisés sur des zones réduites sur lesquelles beaucoup de détails sont voulus.

Ces différents modèles sont discutés plus en détails dans le chapitre 2, nous ne nous intéressons pour le moment qu'aux échelles de représentation. Chaque échelle possède des avantages et des inconvénients et leur utilisation dépend des besoins et des objectifs de chaque simulation. Les modèles microscopiques demandent non seulement une plus grande puissance de calcul pour simuler la même zone qu'un modèle macroscopique mais sont également beaucoup plus difficiles à calibrer. On retrouve dans la littérature de nombreuses méthodes pour calibrer de simulations microscopiques [yu and Fan2017, Brockfeld *et al.*2005, Hoogendoorn and Hoogendoorn2010]. Les modèles microscopiques nécessitent également de récolter des données plus précises pour leur application à un cas concret comme les détails concernant les différentes routes du réseau ou le nombre de voyageurs souhaitant utiliser ce réseau. De petites erreurs sur ce genre de données peut causer de grandes différences de résultats. Les modèles macroscopiques quant à eux sont plus faciles à calibrer puisqu'ils utilisent moins de paramètres mesurables et leur utilisation pour simuler de larges zones géographiques permet un temps de calcul réduit. Ils ne prennent cependant en compte ni les paramètres individuels des voyageurs (écarts entre véhicules, temps de réaction des individus, agressivité, etc.) ni les interactions entre ceux-ci, qui sont parfois nécessaires afin de comprendre certains phénomènes. Les modèles macroscopiques considèrent que ces données individuelles et ces interactions ne changent rien à la dynamique du trafic en général et sont donc limités à des cas d'utilisation où l'étude de cette dynamique globale est suffisante.

La simulation multi-échelles est une réponse à ces limites des échelles existantes, prises en isolation. Elle consiste à simuler chaque phénomène à l'échelle la plus pertinente, c'est-à-dire en utilisant plusieurs modèles de tailles et de finesses différentes et en les faisant interagir. Les simulations multi-échelles utilisent des modèles différents, qui fournissent donc des résultats différents pour les mêmes données d'entrée, il faut donc assurer la cohérence entre ces modèles afin de pouvoir observer des phénomènes cohérents sur tous les modèles. Pour cela, il faut assurer la transmission d'informations entre les différentes échelles en déterminant quelles informations sont nécessaires, quand les échanger et comment les modifier pour qu'elles soient compréhensibles par tous.

Cette cohérence entre les différentes échelles et la manière d'échanger les informations constituent l'articulation entre les échelles et est le cœur de cette thèse.

1.2.2 Une instance concrète : la gestion du trafic dans un quartier de gare

Cette thèse s'inscrit dans le projet MSM (Modélisation de Solutions de Mobilité) de l'institut de recherche SystemX. Le projet MSM s'intéresse particulièrement aux problématiques de la meilleure connaissance et gestion de la mobilité des personnes à l'échelle d'un « quartier gare »ou encore à l'échelle d'un bassin de vie. Il consiste donc à effectuer des études de transports prenant en compte les différents modes de transports utilisés dans une zone d'intérêt afin de déterminer les flux de déplacements à l'intérieur de celle-ci. La zone d'intérêt est en permanence en interaction avec un plus vaste réseau de transport qui correspond à toute l'île de France. Afin de modéliser et de simuler le quartier de manière pertinente il faut donc également simuler le reste de la région afin de pouvoir prendre en compte les voyageurs qui entrent et sortent de la zone depuis le réseau régional. On est bien ici dans le cadre d'une simulation multi-échelles avec deux réseaux à simuler de manières différentes, le quartier qui nous intéresse doit être simulé en détails et le reste de la région de manière plus grossière.

Cette simulation multi-échelles a la particularité de mettre en interaction deux simulateurs travaillant sur une même zone. En effet, le simulateur local simule la zone d'intérêt qui appartient à la région simulée par le simulateur régional qui prend donc également en compte la zone d'intérêt. Ce n'est pas toujours le cas dans les simulations multi-échelles où les simulateurs peuvent également simuler des zones totalement distinctes, bien qu'en interaction.

1.2.3 Viser une solution générique

On peut déjà trouver de nombreux exemples de simulations multi-échelles dans la littérature et il ne s'agit pas ici d'en proposer simplement une nouvelle. Dans cette thèse, nous souhaitons aborder une problématique plus globale en proposant une solution générique et ré-utilisable pour le couplage de deux simulateurs de mobilité indépendamment de leur échelle. Aucune solution n'existe dans ce cadre et faire fonctionner ensemble deux simulateurs d'échelles différentes demande toujours un travail *ad hoc* correspondant au cas

d'étude particulier considéré. L'objectif de cette thèse est donc d'avancer vers une solution ré-utilisable, qui demande le moins de modification possible des simulateurs existants afin d'en simplifier l'utilisation.

Les deux critères principaux de la solution proposée sont donc les suivants :

— Pouvoir faire fonctionner ensemble, de manière cohérente, deux simulateurs d'échelles différentes afin de permettre la réutilisation de simulateurs existants,

— Faciliter au maximum l'utilisation de cette solution par le modélisateur.

1.3 Objectif de la thèse

Dans le travail reporté dans ce manuscrit, les questions de recherche suivantes sont traitées :

— Comment définir les échelles de simulations et comment caractériser les simulateurs ? Il s'agit ici de mettre en évidence les différences entre deux simulateurs d'échelles différentes, de comprendre les difficultés à surmonter pour que les simulateurs communiquent entre eux.

— Comment transmettre les données d'une échelle à l'autre malgré des représentations de données différentes ? Même si deux simulateurs utilisent les mêmes données, ils ne les représentent pas forcément de la même manière, comme vu plus haut par exemple les voyageurs peuvent être représentés individuellement ou sous forme de flux. Il s'agit donc d'être capable de traduire ces données d'une représentation vers une autre de manière automatique.

— Comment définir la cohérence d'une simulation multi-échelles et comment l'assurer ? Si plusieurs modèles différents sont utilisés, leurs résultats seront différents, nous souhaitons cependant être capables d'observer les mêmes phénomènes d'un simulateur à l'autre. Il s'agit donc de déterminer les caractéristiques et les données que doivent partager les deux simulateurs pour assurer cette cohérence.

— Existe-t-il des conditions minimales pour que deux simulateurs puissent fonctionner ensemble et si oui, quelles sont-elles ? Dans la méthodologie proposée, il faut dans un premier temps définir les conditions dans lesquelles la solution multi-échelles pourra être utilisée.

— Quelle est la meilleure architecture dans laquelle intégrer la solution pour la rendre le plus facilement réutilisable possible ? De nombreuses architectures informatiques existent dans le cadre de l'informatique distribuée pour laquelle les différents com-

posants en interactions travaillent séparément les uns des autres. L'objectif est d'explorer ces architectures et d'en adopter une adaptée à notre solution.

En résumé, cette thèse vise à définir la notion de cohérence d'une simulation multi-échelles et à proposer un outil capable d'assurer cette cohérence afin de faire fonctionner ensemble deux simulateurs quelque soient leurs échelles. Il s'agit donc de proposer une solution générique mais également facilement utilisable en définissant une architecture simple et efficace. L'implémentation de la solution sera validée à l'aide d'expérimentations.

1.4 Méthodologie

La thèse aborde la problématique de simulation multi-échelles du trafic du point de vue des études de transport mais également du point de vue informatique. Dans la suite de ce manuscrit on considère donc l'affectation des entités mobiles, leurs comportements et leurs vitesses mais également des problématiques comme la synchronisation des simulateurs ou l'architecture informatique favorisant l'utilisation de plusieurs simulateurs en interaction. Ce positionnement à l'intersection des deux disciplines explique la dualité de l'état de l'art et des contributions de la thèse.

Dans le vaste domaine des études de transport, seules les simulations dynamiques de déplacement sont étudiées ici. Ces simulations sont des outils pour aider à calculer un équilibre d'affectation dans un modèle à 4 étapes et pour déterminer certains résultats de cette affectation comme les temps de parcours de chaque section routière du réseau de transport. Les étapes de génération, distribution et de choix modal ne sont pas du tout considérées dans cette thèse. D'un point de vue informatique, on cherche à obtenir un outil efficace, robuste et réutilisable facilement, pour s'en assurer on étudie les architectures de simulations distribuées en essayant de retrouver les caractéristiques principales d'intergiciels existants et de les reproduire dans notre solution. En effet, la solution proposée est une brique logicielle à laquelle seront connectés les deux simulateurs et qui assure la cohérence de simulation : un intergiciel (*middleware* en anglais, cf. chapitre 4). L'utilisation d'un logiciel tiers indépendant des simulateurs existants a pour but de demander le travail le moins important possible du modélisateur sur les simulateurs qu'il souhaite utiliser. Ainsi, puisque la solution sera découplée des simulateurs utilisés, l'intergiciel permettra de proposer une solution la plus générique possible.

Cet intergiciel a pour objectif d'assurer la cohérence du système de simulation comportant deux simulateurs d'échelles différentes et donc utilisant des modèles différents.

Cette cohérence est définie dans le chapitre 4 et vise à assurer que les voyageurs des deux simulateurs se comportent de la même manière en prenant notamment des chemins équivalents dans l'un ou l'autre des deux modèles. Ainsi il permettra d'articuler des échelles de simulation locales et régionales à l'aide de trois axes d'agrégation qui représentent trois composantes de ces échelles :

— Les entités mobiles que sont les voyageurs et/ou les véhicules
— le réseau de transport qui est l'axe spatial
— le pas de temps de la simulation qui est l'axe temporel.

Ces trois axes serviront d'outils et seront traités par l'intergiciel pour assurer la cohérence entre les deux simulations.

La priorité et le centre d'intérêt de ce manuscrit est l'articulation des échelles. La multimodalité est au second plan d'intérêt et sera visible dans une implémentation particulière des expérimentations. Du point de vue informatique, on étudie particulièrement l'architecture HLA (*High Level Architecture*), les intergiciels orientés service (SOM) et différentes solutions génériques avec des fédérations de composants et des protocoles d'interaction établis spécialement pour coordonner des simulations.

1.5 Structure du manuscrit

Après ce premier chapitre introductif, la thèse est composée de deux parties et six chapitres supplémentaires :

— La première partie est un état de l'art composé de deux chapitres.
  — Le chapitre 2 est un état de l'art des simulations de trafic multi-échelles et multi-niveaux existantes, elle est introduite par la présentation de plusieurs simulateurs d'échelles différentes.
  — Le chapitre 3 est un état de l'art sur les architectures de simulation distribuées et l'utilisation des architectures orientées services dans le domaine de la simulation.
— La deuxième partie relate les contributions de cette thèse, elle est composée de quatre chapitres.
  — Le chapitre 4 correspond aux contributions théoriques concernant la modélisation multi-échelles apportées par cette thèse. Nous y présentons entre autres 3

axes d'agrégation permettant de définir les échelles de simulation. Ce chapitre discute également de la cohérence entre deux simulations d'échelles différentes. Ensuite, les principales fonctionnalités de l'intergiciel sont présentées (synchronisation, transformation des voyageurs, cohérence du réseau de transport).

— Le chapitre 5 expose les solutions techniques et l'architecture informatique retenues pour le développement de l'intergiciel. On y trouve l'ensemble des paramètres à configurer afin de pouvoir utiliser l'intergiciel, son fonctionnement détaillé avec chaque étape de son exécution ainsi qu'une interface présentant les différentes méthodes que doivent implémenter les simulateurs pour pouvoir fonctionner ensemble. Ce chapitre regroupe également les différentes expérimentations qui servent de support au chapitre précédent. Ces expérimentations servent à comparer plusieurs solutions proposées dans l'intergiciel et à vérifier les effets de ce dernier sur des simulateurs simplifiés développés dans le cadre de la thèse mais également sur des simulateurs connus et régulièrement utilisés dans la littérature.

— Le dernier chapitre conclue la thèse et proposes de nouvelles pistes pour des recherches futures.

1.6 Publications

Ce travail de thèse a donné lieu à trois articles présentés à des conférences internationales, et un article soumis à un journal international :

1.6.1 Articles publiés

— **Boulet, X., Zargayouna, M., Scemama, G., Leurent, F. (2020). Coupling Multi-agent and Macroscopic Simulators of Traffic. In *Agents and Multi-agent Systems : Technologies and Applications 2019* (pp. 323-332). Springer, Singapore.**

Cet article présente une solution générique pour combiner un simulateur macroscopique travaillant sur une grande zone, qui contient une zone plus petite simulée par un simulateur multi-agents. Le principal défi est d'assurer la cohérence entre les deux simulateurs sur la plus petite zone, puisqu'elle est simulée par les deux simulateurs en même temps. Nous mettons d'abord en évidence les questions à aborder

et les problèmes à résoudre lors du couplage de deux simulateurs existants, puis nous proposons des solutions pour le couplage, et enfin nous les évaluons sur un exemple de scénario.

— **Boulet, X., Zargayouna, M., Scemama, G., Leurent, F. (2019, November). Service-Oriented Architecture for Multiscale Traffic Simulations. In *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)* (pp. 1-8). IEEE.**

Dans cet article, nous proposons un modèle d'intergiciel permettant de coupler des simulations indépendantes. Nous prenons en compte tous les traitements et flux nécessaires pour permettre une orchestration cohérente des simulations. Les résultats montrent que l'intergiciel est capable de créer une nouvelle simulation multi-échelle à partir de celles qui existent, tout en minimisant l'incohérence entre elles

— **Boulet, X., Zargayouna, M., Leurent, F., Kabalan, B., Ksontini, F. (2017). A dynamic multiagent simulation of mobility in a train station. *Transportation research procedia*, 27, 744-751.**

Avant de nous intéresser aux simulations multi-échelles, nous nous sommes intéressés à la simulation à l'échelle du quartier. Dans cet article, nous proposons un modèle multi-agents pour de telles simulations. La proposition comprend trois parties. La première partie est un modèle de simulation multi-agents des voyageurs et de leur interaction avec le quai de la gare. Les passagers adaptent leur comportement aux autres voyageurs tout en poursuivant leur objectif de voyage, et en interaction avec les sources d'information disponibles, soit locales à la gare, soit personnalisées (par le biais de leurs *smartphones* par exemple). La modélisation du système d'information de ce voyageur est la deuxième partie du modèle, où les opérateurs de transport sont en mesure de définir des stratégies concernant le type d'informations à fournir, leur fréquence et le support utilisé pour les diffuser. Enfin, la troisième partie du modèle est la plate-forme de gestion de la gare, qui surveille à la fois les trains et les comportements des piétons dans la gare. La plate-forme est responsable de l'optimisation de la fluidité du trafic et de la sécurité des passagers et propose des actions liées à la déviation de la foule et au contrôle d'accès. L'architecture préconise une modélisation en boucle fermée entre la simulation, la gestion de la gare et l'information aux voyageurs, permettant de tester des systèmes de contrôle complets plutôt que des stratégies spécifiques.

1.6.2 Article soumis

— **Boulet, X., Zargayouna, M., Scemama, G., Leurent, F. Middleware for multiscale mobility simulations. Journal of Intelligent Transportation Systems, Taylor & Francis, 30 pages.**

Cet article est la version la plus complète relative à ce travail de thèse. Dans cet article, nous préconisons l'utilisation des simulations de déplacement existantes, travaillant à des échelles différentes. Pour ce faire, nous proposons un modèle d'intergiciel et une API permettant de coupler des simulations de déplacement indépendantes, fonctionnant à différentes échelles de représentations, spatiales, temporelles et celles des entités mobiles. Nous considérons tous les traitements et *workflows* nécessaires pour permettre une orchestration cohérente des simulations en une seule. Les résultats montrent que l'intergiciel est capable de créer une nouvelle simulation multi-échelles cohérente en réutilisant et en orchestrant celles qui existent.

Première partie

État de l'art

Chapitre 2

Les échelles dans les simulations de mobilité

Sommaire

La modélisation et la simulation jouent un rôle important dans l'analyse des réseaux de transport. Plusieurs aspects des réseaux de transports peuvent être analysés, chaque aspect correspondant à une échelle de représentation particulière. De plus en plus d'études nécessitent la considération de plusieurs aspects et de plusieurs échelles simultanément. Dans cette thèse, nous faisons le choix de la réutilisation de simulateurs existants pour ce genre de problèmes. Deux problématiques principales sont abordées dans cette thèse :

— Comment assurer une simulation multi-échelles de mobilité cohérente ?
— Comment faire communiquer les différents programmes et au sein de quelle architecture ?

C'est la raison pour laquelle cette partie « état de l'art »comporte deux chapitres. Tout d'abord ce chapitre, qui concerne les simulations de transport et plus précisément les simulations multi-échelles. Ensuite, le chapitre suivant dans lequel nous présentons plusieurs intergiciels de communication entre programmes ainsi que des architectures pour l'informatique distribuée sur lesquelles nous nous appuyons pour développer notre solution.

2.1 Simulation indépendante de différentes échelles

Une manière de classer les modèles et les simulations de trafic consiste à le faire en fonction de le niveau de détail de représentation du monde. En fonction des besoins auxquels doit répondre la simulation, différentes échelles sont possibles. Les modèles macroscopiques sont généralement utilisés pour de la planification stratégique sur de grands réseaux de transport et sur une longue durée. Ces modèles peuvent par exemple servir à déterminer comment modifier le réseau de transport afin d'anticiper une croissance de la population dans certaines zones de la ville. Les modèles microscopiques en revanche sont utilisés pour de plus petits réseaux et des problématiques plus spécifiques comme la détermination du placement de panneaux de signalisations pour le trafic routier ou pour la définition des limites de vitesse sur une voie. Les modèles mésoscopiques se fondent sur une échelle intermédiaire entre les modèles microscopiques et macroscopiques, nécessaire lorsque la simulation est trop importante pour être simulée de manière microscopique, mais que certains détails macroscopiques importants manquent.

Le coût informatique de la simulation, i.e. le temps et la mémoire qu'utilise un ordinateur pour l'exécuter, est un paramètre important pour déterminer l'échelle de simulation. Ce critère est l'une des raisons pour lesquels un modèle microscopique ne peut pas toujours être utilisé, surtout pour les zones géographiques étendues. Le second paramètre qui empêche l'utilisation de modèles détaillés sur un problème de grande taille est la difficulté à calibrer ces modèles, qui demande beaucoup d'informations qui doivent être récoltées en amont de la simulation afin d'être valides [Balakrishna *et al.*2007, Brockfeld *et al.*2005].

Pour le trafic routier, les modèles macroscopiques supposent un nombre de véhicules présents sur les routes suffisamment important pour qu'ils puissent être considérés comme un flux. Ces modèles utilisent des équations mathématiques permettant d'exprimer l'écoulement de ces flux de voyageurs dans un réseau de transport. Ainsi, les voyageurs ne sont pas représentés individuellement, mais on utilise des informations telles que :

- la densité $\rho(x,t)$ qui est le nombre de véhicules sur une portion d'un arc routier avec x la position du véhicule et t l'instant de la simulation,
- la vitesse moyenne $v(x,t)$,
- ou encore $q(x,t)$ qui représente le nombre de véhicules qui passent en un point par unité de temps.

Ces trois variables sont connectées grâce aux équations suivantes :

- $q(x,t) = \rho(x,t)v(x,t)$
- $\frac{\delta\rho}{\delta t} + \frac{\delta(\rho v)}{\delta x} = 0$

Nous avons alors deux équations à trois inconnues. Pour qu'un modèle puisse décrire le trafic, une troisième équation doit être définie. Pour cette troisième équation, le modèle LW proposé par Lighthill et Whitham [Lighthill and Whitham1955a] est un des plus connus et part du principe que la vitesse peut être décrite comme une fonction de la densité du traffic ce qui amène à la relation [Lighthill and Whitham1955b]. Il existe d'autres modèles comme le modèle de Payne [Isaksen and Payne1973] qui est dérivé des modèles de poursuite et donne :

— $\frac{\delta v}{\delta t} + v\frac{\delta v}{\delta x} = \frac{1}{T}(V_e(\rho) - v) - \frac{c^2(\rho)}{\rho}\frac{\delta \rho}{\delta x}$

Avec T le temps de réaction, V_e la vélocité à l'équilibre en fonction de ρ et c la fonction d'anticipation liée à la densité [Payne1971].

Les modèles microscopiques sont bien plus détaillés et considèrent les voyageurs individuellement. Ces modèles prennent en compte la position individuelle de chaque voyageur, sa vitesse, son accélération et parfois ses interactions avec les autres voyageurs. Comme pour les modèles macroscopiques, on trouve plusieurs types de modèles microscopiques, certains avec une représentation discrète du temps et de l'espace [Benjaafar *et al.*1997] et d'autres avec des représentations continues [Tordeux2010]. Dans les modèles de poursuite communément utilisés, la vitesse d'un conducteur dépend de celle du conducteur précédent, de l'espace qui les sépare et du temps de réaction des conducteurs. Les auteurs dans [Treiber and Kesting2013] présente par exemple le modèle de Gipp ou le modèle de conducteur intelligent (IDM).

Ainsi, chaque échelle de simulation possède des avantages et des inconvénients et plusieurs modèles différents existent pour chaque échelle. Le choix de l'utilisation d'un modèle plutôt qu'un autre dépend du but de la simulation, de ses contraintes, ainsi que des données disponibles. Tous ces modèles de simulation possèdent des limites directement liés à leurs échelles. Les modèles mésoscopiques et macroscopiques sont plus simples à calibrer que les modèles microscopiques grâce à leurs faible nombre de paramètres qui sont plus facilement mesurables. Cependant ces modèles ne peuvent pas être utilisés lorsque l'interaction entre les entités est cruciale pour les résultats de la simulation. Par exemple, pour la modélisation et la simulation du contrôle adaptatif des signaux de trafic routier sur une zone réduite il faut connaître la position précise des véhicules. En revanche les modèles microscopiques possèdent deux limites majeures, leur temps de calcul et leur paramétrage. Le temps de calcul vient de la modélisation des voyageurs qui sont représentés individuellement et pour lesquels de nombreuses interactions peuvent être calculées. La calibration des modèles microscopiques peut être faite manuellement ou à l'aide d'algorithmes d'optimisation, mais dans tous les cas c'est une tâche qui prend du temps et qui

nécessite la collecte de nombreuses données qui sont comparées aux résultats de la simulation. Les simulations multi-échelles cherchent en général à palier ces limites en apportant les avantages des différentes échelles.

2.2 Simulations multi-échelles

Il existe différentes manières d'utiliser deux simulateurs d'échelles différentes pour effectuer une analyse multi-échelles. Les auteurs dans [Holmgren *et al.*2014] présentent les trois méthodes suivantes (cf Figure 2.1 [Holmgren *et al.*2014]).

— Les deux simulateurs peuvent être utilisés séparément puis l'utilisateur effectue une analyse globale des différents résultats. Il ne s'agit pas à proprement parler de simulations multi-échelles puisque les simulations informatiques n'interagissent pas, c'est à l'utilisateur humain de combiner les résultats et d'en tirer les informations pertinentes.

— Un simulateur, en général celui suivant le modèle macroscopique, peut être utilisé seul dans un premier temps, et ses résultats peuvent ensuite servir de données pour le second simulateur. Éventuellement, le premier simulateur peut par la suite être exécuté à nouveau avec les nouvelles données de sortie du second simulateur. Il s'agit de simulations en séquence où les résultats d'un simulateur servent de données d'entrée à l'autre simulateur. Un exemple courant est la génération de la demande du simulateur microscopique grâce aux résultats d'un simulateur macroscopique dont la zone simulée contient la zone du simulateur microscopique.

— Finalement, les deux simulateurs peuvent être utilisés d'une manière intégrée pour former une seule simulation. Il s'agit du cas le plus complexe et se différencie du cas précédent par le fait que les deux simulateurs échangent des données pendant leurs simulations et pas seulement à la fin de l'une d'entre elles.

La première approche n'est pas une simulation multi-échelles et ne sera donc pas discutée ici. La seconde approche est souvent utilisée pour combiner des modèles d'affectation statiques avec des simulations. Cette approche ne représente pas exactement le centre d'intérêt de cette thèse, puisqu'il n'y pas véritablement d'interaction entre les deux composants lors de leur exécution et donc une partie des problématiques abordées plus tard ne sont pas rencontrés dans ce cas. Par exemple, Vissim est le modèle de simulation microscopique de PTV et peut utiliser la topologie du réseau de transport et les matrices origine-destination exportée de Visum, qui est le modèle d'affectation macroscopique statique de PTV. Le modèle macroscopique est ici utilisé pour proposer un réseau et une demande

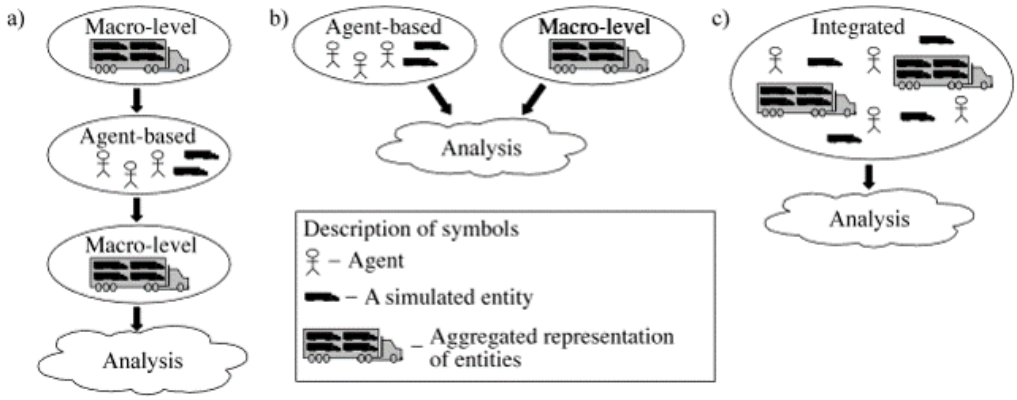

FIGURE 2.1 – Trois types d'interaction entre simulateurs

pour le modèle microscopique. On retrouve le même principe dans [Montero *et al.*1998] ou le modèle d'affectation est EMME et le simulateur microscopique est AIMSUN. Dans ces deux cas, il n'y a pas de retour des résultats du modèle microscopique vers le modèle macroscopique ce qui signifie que la demande générée et envoyée au modèle microscopique ne dépend pas des résultats de celui-ci. Enfin, dans [M.D. Hall1998], la sortie du modèle macroscopique statique sur un réseau est utilisée dans la simulation d'une partie de réseau de manière microscopique.

Ces simulations multi-échelles séquentielles peuvent être suffisantes si les résultats du second simulateur n'influencent pas la première simulation. En revanche, si les deux simulations s'influencent mutuellement ou si la simulation doit être connectée en temps réel alors l'interaction dynamique est nécessaire et cela nous amène à la troisième catégorie ; une simulation multi-échelles interactive où les deux simulateurs sont connectés ensemble. On trouve de nombreux exemples de ce type dans la littérature.

Par exemple, dans [Poschinger *et al.*2002], l'auteur réalise le couplage d'un modèle microscopique de poursuite - IDM [Derbel *et al.*2013] avec le modèle macroscopique de second ordre de Payne [Payne1971]. Pour assurer la communication entre les deux modèles, une zone de transition est prévue dans laquelle la transition micro-macro se fait par agrégation des données et la transition macro-micro se fait grâce à un ajout d'informations à l'aide d'un processus stochastiques.

On trouve également le modèle Hystra [Bourrel and Henn2002, Bourrel and Lesort2003], qui est un modèle hybride combinant un modèle microscopique et un modèle macroscopique. Le modèle macroscopique suit les équations du premier ordre de LWR (Lighthill-Whitham-Richards) [Lighthill and Whitham1955a] et le modèle microscopique est le mo-

dèle de vélocité optimale qui est également fondé sur le modèle LWR. Ainsi, Bourrel cherche à rendre les résultats des modèles cohérents en définissant deux contraintes :

— On doit avoir la conservation des flux de véhicules aux interfaces limites des modèles ;
— dans des conditions stationnaires, les deux modèles doivent fournir le même résultat.

Le modèle proposé par [Mammar *et al.*2006] ressemble à celui de Bourrel dans le couplage mais ne se fonde pas sur le même modèle macroscopique. Le modèle macroscopique est cette fois le modèle de second ordre ARZ [Lebacque *et al.*2007] et le modèle microscopique reste le modèle de vélocité optimale. Une fois encore, les modèles sont choisis et retravaillés afin d'assurer leur cohérence.

AIMSUN next, qui est une solution de prévision du trafic fondée sur la simulation, possède un modèle microscopique ainsi qu'un modèle mésoscopique. Ces deux modèles peuvent être utilisés en interaction lorsque la zone simulée est trop large pour le modèle microscopique, dans le cadre d'une simulation hybride. La simulation hybride permet de simuler une zone définie comme microscopique et de simuler le reste du réseau de manière mésoscopique. Pour ce faire, la simulation est dans un premier temps calibrée avec uniquement le modèle mésoscopique et lorsque l'affectation est satisfaisante, le modèle microscopique est ajouté en prenant en compte cette affectation. Il est à noter qu'AIMSUN propose également un modèle macroscopique statique qui peut être utilisé en amont de l'un ou l'autre des deux modèles précités, ce qui correspond à l'approche séquentielle.

Paramics [Smith *et al.*1995] est un modèle de simulation microscopique du trafic qui a été combiné avec le modèle mésoscopique Dynasmart [Jayakrishnan *et al.*1994]. Le modèle mésoscopique permet d'améliorer l'affectation du modèle microscopique sur des réseaux de taille importante ou la représentation microscopique, et donc détaillée, des routes implique un temps de calcul trop long [Jayakrishnan *et al.*2001].

Simmobility [sim, Adnan *et al.*2016] est une plateforme de simulation fondée sur les activités, multimodale et surtout multi-échelles. En effet Simmobility comporte un module long terme, un module moyen terme [Lu *et al.*2015] et finalement un module court terme [Azevedo *et al.*2017]. Le module court terme sert à simuler le déplacement des voyageurs de manière microscopique et l'impact des dispositifs de communication sur ces comportements, le module moyen terme sert à simuler les activités avec des agrégations de véhicules. Finalement le modèle macroscopique fonctionne d'année en année et détermine l'utilisation des territoires et les activités économiques.

2.3 Conditions de cohérence des simulations multi-échelles

Dans [Burghout2004, Burghout and Koutsopoulos2006], les auteurs proposent une simulation hybride avec le modèle microscopique MITSIMLab fonctionnant au avec le modèle mésoscopique Mezzo. En plus du couplage de ces deux modèles, les auteurs dans [Burghout2004] présentent les conditions nécessaires suivantes pour que les deux modèles soient cohérents entre les différents niveaux de détails :

1. La cohérence dans le choix du chemin (affectation) et la représentation du réseau routier. L'affectation des voyageurs ne doit pas dépendre du modèle avec lequel ils sont simulés, pour cela la représentation entre les deux modèles doit être cohérente et les temps de parcours doivent être les mêmes.
2. La cohérence des dynamiques de trafic aux frontières micro-méso. Par exemple si une file d'attente se forme dans un modèle et atteint la limite de la zone du modèle, elle doit se former également dans l'autre modèle.
3. La cohérence entre les résultats des deux modèles sur une même zone pour la même demande. Pour cela il faut calibrer les deux modèles ensemble.
4. La minimisation des échanges entre les deux modèles grâce à un paradigme de communication et de synchronisation efficace. Sans cela, la communication prend un temps trop important dans la simulation.

Ces différents modèles hybrides combinant à chaque fois deux modèles d'échelles différentes mettent en avant plusieurs besoins et contraintes. Tout d'abord, on doit définir des critères de cohérence pour que les résultats des simulations soient les mêmes, c'est ce que nous faisons dans le chapitre 4. Ensuite, il faut faire en sorte que les deux modèles utilisés soient définis et calibrés de manière à être cohérents. Dans cette thèse, nous proposons une solution générique et ne choisissons donc pas les modèles ou leur paramétrage. C'est la raison pour laquelle nous forçons la cohérence en permettant aux simulateurs de se corriger mutuellement. Nous voyons aussi l'importance de la synchronisation des simulateurs, de la représentation du réseau de transport et des différentes manières de représenter les voyageurs, cela annonce notre définition des échelles de simulations dans le chapitre 4. Finalement, le passage entre les différents modèles doit se faire à l'aide de fonctions d'agrégation et de désagrégation. La désagrégation consiste à créer de nouvelles données et se fait donc à l'aide de processus stochastiques.

2.4 Simulations multi-niveaux

En plus des simulations multi-échelles, on retrouve dans la littérature des simulations dites « multi-niveaux ». Ces simulations intègrent plusieurs échelles au sein du même simulateur disposant de plusieurs modèles entre lesquels il peut choisir pour représenter les voyageurs. Il s'agit donc de simulateurs créés dès l'origine dans l'objectif de simuler plusieurs échelles simultanément. Un des avantages principaux de ces simulateurs multi-niveaux est qu'en général ils permettent d'adapter dynamiquement les échelles lors de l'exécution de la simulation pour obtenir la meilleure qualité de détails en fonction de l'état de la simulation. Dans [Sewall *et al.*2011], les auteurs présentent une simulation hybride entre un modèle multi-agents et un modèle de flux macroscopiques. N'importe quelle zone de la simulation peut passer d'un modèle à l'autre pendant l'exécution et l'article explique comment le zoom est effectué, comment transformer des agents en flux et réciproquement. Dans [Navarro *et al.*2011] et [Navarro *et al.*2013], l'auteur travaille également sur la représentation des entités dans le cadre de la simulation multi-niveaux de piétons. Il propose une méthode pour agréger et désagréger les agents de la simulation en groupes grâce à des fonctions de distance entre ces agents. Là encore cette agrégation est dynamique et peut être modifiée durant la simulation, la taille des groupes et le niveau d'agrégation peut varier selon l'importance des évènements de la simulation.

Dans un autre domaine que la simulation de transport, les auteurs dans[Sharpanskykh and Treur2011] s'intéressent également à l'agrégation d'agents et présentent deux méthodes :

— Une méthode de moyennes pondérées où l'état d'un groupe d'agents est estimé en faisant la moyenne des états des agents le composant, pondérée par l'importance de l'agent.
— Une méthode d'abstraction fondée sur un invariant qui consiste à déterminer une propriété qui ne varie pas au cours du temps et à s'en servir comme loi de conservation.

Pour revenir aux transports, les auteurs dans [Gaud *et al.*2008] proposent d'aller encore plus loin dans la flexibilité du niveau d'agrégation pendant la simulation et considèrent que les échelles de simulation ne devraient pas être fixées *a priori*. Ainsi, le modèle proposé ne possède pas deux échelles fixes mais définit une hiérarchie entre les éléments qui leur permet d'être agrégés au besoin et de créer la meilleure échelle en fonction du contexte. Le travail de [Haman *et al.*2017] est fondé sur la même idée d'organisation ho-

lonique des éléments mais adapte ce concept aux simulations multi-agents de véhicules.

Dans [Abouaïssa *et al.*2014], les auteurs proposent un simulateur multi-agents multi-niveaux de flux de trafic routier nommé JAM-FREE considérant que les simulations les plus larges ont besoin de plusieurs modèles d'échelles différentes, macroscopiques pour les autoroutes et microscopiques pour les sections urbaines. Afin de pouvoir simuler de grandes zones ils proposent un simulateur multi-niveaux. Le simulateur est basé sur SIMILAR (SImulations with MultI-Level Agents and Reactions). Chaque route est divisée en sous-ensembles qui peuvent être soir simulés d'une manière macroscopique soit avec le modèle microscopique du conducteur intelligent - IDM - qui permet de modéliser l'accélération des véhicules et de gérer le changement de voies.

Dans [Soyez2013], le méta-modèle IRM4MLS propose une approche théorique et formelle d'un modèle multi-niveaux pour les systèmes de systèmes. Il traite l'influence et la perception des agents d'un niveau par rapport à un second niveau et présente les concepts centraux du méta-modèle que sont les agents, l'environnement et les niveaux. Finalement, il propose un modèle de graphe hiérarchique des niveaux permettant de montrer les différents niveaux de la simulation et leurs interactions grâce à des arcs représentant les différentes fonctions d'agrégation utilisées pour passer d'un niveau à l'autre.

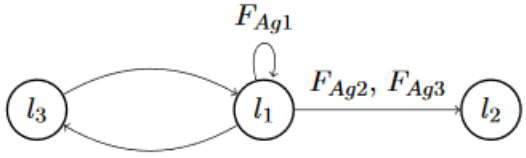

FIGURE 2.2 – Exemple de graphe des niveaux - image provenant de Conception et modélisation de systèmes de systèmes : une approche multi-agents multi-niveaux.

Les simulations multi-niveaux sont conçues et implémentées en couplage fort, et créé pour un besoin multi-échelles identifié originellement. L'objectif de cette thèse est d'éviter la création *ex nihilo* d'une simulation multi-niveaux à chaque fois que le besoin s'en ressent. Nous proposons donc un modèle permettant de coupler des simulations mono-échelle et mono-niveau existantes pour créer de nouvelles en les composant. Ce modèle propose en fait une interface générique pour le couplage de simulations d'échelles différentes. Ce sujet fait l'objet de la section suivante.

2.5 Interfaces génériques

Dans tous les exemples vus précédemment, qu'il s'agisse de simulations multi-échelles ou multi-niveaux, il s'agit de cas particuliers d'interactions entre différents modèles définis. Nous cherchons à présent des travaux génériques sur le couplage de modèles de simulation de mobilité.

Dans [Biedermann *et al.*2014], les auteurs proposent un framework pour coupler n'importe quelle paire de modèles (mésoscopique × microscopique) de déplacements de piétons. Pour ce faire il affirme que pour qu'un framework de couplage soit générique il doit utiliser des zones de transition afin que les piétons puissent se déplacer d'un modèle à l'autre sur toute la limite de la zone simulée et non seulement sur des points d'entrée et de sortie. Il met également en avant le fait que deux modèles de simulation peuvent avoir des pas de temps différents et qu'il est donc nécessaire de trouver quand envoyer les données et comment modifier ces données pour qu'elles correspondent à ce qu'attend le simulateur qui les reçoit grâce à des interpolations. Dans [Biedermann *et al.*2016], l'auteur ajoute l'échelle macroscopique qui correspond à l'arrivée des voyageurs sur le site d'un évènement public qu'il souhaite modéliser.

On trouve également le travail de [Joueiai *et al.*2014] pour le trafic de véhicules, qui propose un framework de modélisation multi-échelles permettant de coupler n'importe quel modèle macroscopique avec un modèle microscopique. Le cadre de l'article est la simulation de zones exclusivement microscopiques ou macroscopiques, on peut donc trouver, par exemple, une zone microscopique puis une zone macroscopique et enfin une nouvelle zone microscopique. Ils mettent en avant l'importance de deux formes de cohérence : la cohérence locale et la cohérence globale. La cohérence locale concerne la propagation des phénomènes comme la formation de files qui, si elles atteignent la limite d'un modèle doivent se propager dans le second. La cohérence globale concerne le passage des véhicules entre les modèles et la transformation de leurs caractéristiques. En effet, lorsqu'un véhicule passe d'un modèle microscopique à un modèle macroscopique, il perd des informations qu'il doit ensuite être capable de reconstruire lorsqu'il rentre à nouveau dans un modèle microscopique comme s'il ne l'avait jamais quitté.

2.6 Conclusion

Dans ce chapitre, nous avons défriché la question d'échelles dans les simulations de mobilité. Nous avons décrit les principales échelles considérées, leur intérêt, ainsi que les deux principales manières de les composer. Les simulations multi-échelles composent des simulations indépendantes travaillant sur des échelles différentes. Il existe de nombreux travaux sur les simulations multi-échelles et même si la plupart d'entre elles ne concernent que des outils spécifiques, on trouve des approches génériques comme dans [Biedermann *et al.*2014] ou encore [Joueiai *et al.*2014]. Cependant, ces études restent incomplètes et ne prennent pas en compte tous les aspects liés aux modèles multi-échelles comme détaillé dans ce chapitre. C'est la raison pour laquelle un des premiers objectifs de la thèse est de déterminer clairement et de manière approfondie ce qui caractérise les échelles de simulation et ainsi proposer une notion de cohérence pertinente en s'inspirant de [Burghout2004] comme on le verra dans nos contributions.

Les simulations multi-niveaux créent des simulations dédiées représentant simultanément plusieurs échelles. Pour éviter de créer un grand nombre de simulations multi-niveaux dédiées, notre thèse est qu'il est plus profitable de composer des simulations mono-échelles existantes. Pour ce faire, nous proposons un modèle d'intergiciel facilitant le couplage de telles simulations. La définition de l'intergiciel posent des questions, d'un point de vue purement informatique, qui font l'objet du chapitre suivant de l'état de l'art.

Chapitre 3

Architectures distribuées

Sommaire

3.1 Introduction

Pour limiter l'explosion de propositions de simulateurs multi-niveaux, nous faisons le choix dans cette thèse de proposer un intergiciel pour le couplage de simulations travaillant à des échelles différentes. Nous souhaitons proposer une solution générique réutilisable afin de coupler des simulateurs d'échelles différentes à l'aide d'un intergiciel. C'est la raison pour laquelle nous nous intéressons aux différents types d'intergiciels existants, afin d'en déterminer les principales caractéristiques. Dans un second temps, nous explorons

les architectures d'informatique distribuée permettant de faire communiquer plusieurs programmes. Finalement, nous décrivons l'utilisation des architectures orientées service dans le cas des simulations.

3.2 Moyens de communication entre logiciels et intergiciels

L'informatique distribuée est une branche de l'informatique où plusieurs parties de l'architecture peuvent se trouver à des endroits différents du réseau, sur des machines différentes. Faire fonctionner en interaction deux simulateurs rentre dans le cadre de l'informatique distribuée. Cette section s'intéresse donc aux différentes solutions qui existent pour faire communiquer deux programmes ensemble. En informatique distribuée, on utilise souvent l'architecture client/serveur où l'information se trouve sur un où plusieurs serveurs qui doivent la distribuer et la répartir à des programmes clients en fonction de leurs requêtes. On peut trouver plusieurs types d'architecture client-serveur :

— L'architecture pair à pair est une architecture clientserveur où chaque programme connecté est susceptible de jouer tour à tour le rôle de client et celui de serveur.
— L'architecture à 2 niveaux est une architecture clientserveur où le client demande une ressource au serveur qui la fournit à partir de ses propres ressources.
— L'architecture à 3 niveaux ajoute un niveau supplémentaire à l'architecture à 2 niveaux, permettant de spécialiser les serveurs dans une tâche précise, ce qui donne un avantage de flexibilité, de sécurité et de performance. Un client qui demande une ressource via une interface utilisateur (généralement un navigateur web) chargée de la présentation de la ressource ; un serveur d'application qui fournit la ressource, mais en faisant appel aux ressources d'un autre serveur ; un serveur de données qui fournit au serveur d'application les ressources requises pour répondre au client.
— L'architecture à N niveaux n'ajoute pas de niveaux supplémentaires à l'architecture à 3 niveaux, mais introduit la notion d'objet qui offre la possibilité de distribuer les services entre les 3 niveaux selon N couches, permettant ainsi de spécialiser les serveurs davantage.

Un intergiciel est un composant logiciel intermédiaire qui se situe entre deux autres composants logiciels afin de les faire communiquer. Le but d'un intergiciel est de réduire les difficultés liées à l'hétérogénéité des différentes composantes de l'architecture que va ren-

contrer le développeur. Il s'agit donc, comme pour un framework, d'un ensemble de fonctions que le développeur utilise sans avoir à se soucier de certaines difficultés techniques de plus bas niveau. Les intergiciels sont utilisés dans de nombreux domaines (informatique embarquée, Internet des objets, simulations, jeux vidéos, etc.) et il en existe de nombreux types différents. Le travail de [Bishop and Karne2003] propose une classification des intergiciels précise qui se fonde sur la manière dont l'intergiciel assiste une application ou aide à l'intégration de plusieurs applications dans un système. Les intergiciels peuvent intervenir à divers niveaux : entre une application et le système d'exploitation, entre deux applications, etc. La définition d'un intergiciel proposée dans l'article est donc la suivante : « Un intergiciel est un logiciel qui aide une application à interagir ou communiquer avec d'autres applications, un réseau, du matériel et/ou un système d'exploitation. Ce logiciel aide les développeurs en leur évitant des complexités liées à un système distribué ». Les auteurs proposent ensuite la classification reportée dans la figure 3.1 [Bishop and Karne2003].

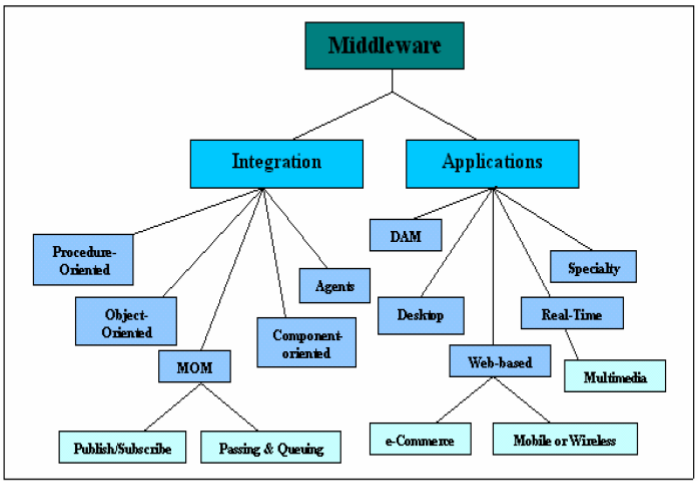

FIGURE 3.1 – Catégories d'intergiciels

Les intergiciels sont dans un premier temps séparés entre les intergiciels d'intégration et les intergiciels d'application. Alors que les intergiciels d'intégration doivent être intégrés dans un système distribué hétérogène à l'aide de protocoles de communication, les intergiciels d'application fonctionnent au sein d'une application pour lui permettre d'accéder plus facilement à des fonctionnalités comme un accès à des données, au web , etc. Il y a donc une différence entre des intergiciels entre applications et des intergiciels entre couches d'une même application. Dans les intergiciels d'intégration on retrouve :

— Les intergiciels orientés message (MOM) récupèrent les messages envoyés par des applications sous forme de files de messages, auxquelles peuvent accéder plusieurs processus d'un même réseau. Cette approche est fondée sur le concept de message

et de canal de communication qui peut être de « un vers un », de « un vers plusieurs », de « plusieurs vers un », ou de « plusieurs vers plusieurs ». L'intergiciel sert à définir quelles applications ont accès à quel message, ainsi que l'ordre dans lequel les messages sont distribués. Un *Message broker* est un programme permettant de traduire un message depuis l'émetteur vers le récepteur. Parmi ses implémentations les plus utilisées, nous retrouvons : Apache ActiveMQ, Fuse, RabbitMQ, ainsi que le protocole AMQP (*Advanced Message Queuing Protocol*) pour traiter les problématiques d'interopérabilité.

— Les intergiciels RPC (*Remote Procedure Call*) : Dans les systèmes distribués, le modèle RPC permet la communication entre processus distants. Il permet à une application d'exécuter une procédure offerte par un serveur distant. Le serveur est vu comme un ensemble de procédures exécutables via des appels par un client distant afin de réaliser une tâche spécifique. Le client convertit les paramètres de la procédure en un message (*marshalling*) qui sera récupéré par le serveur qui transformera ce message en paramètres. L'inverse se produit lorsque le serveur envoie le retour de la procédure au client.

— Les intergiciels RMI (*Remote Method Invocation*), et les intergiciels orientés objet en général, sont des intergiciels RPC qui permettent d'appeler les méthodes des objets d'un autre programme. Il s'agit donc d'intergiciels RPC spécifiques à la programmation orientée-objet (et à Java en particulier pour RMI). Cette approche se base sur l'utilisation d'un ORB (*Object Request Broker*) afin de permettre l'interaction distante avec des objets déployés dans un même espace mémoire local. Ces objets communiquent entre eux par l'envoi et la réception de requêtes et de réponses, de manière transparente et portable. Cette approche représente ainsi une extension de l'approche RPC, spécifique aux objets.

— Les intergiciels réflectifs (ou fondés sur les composants) se caractérisent par leur flexibilité et leur capacité à utiliser de nouveaux composants ajoutés lors de l'exécution du programme et non pas seulement à la compilation. Ces intergiciels se basent sur la réflexion et sur le protocole méta-objet de Gregory Kiczales [Kon *et al.*2001].

Parmi les intergiciels d'application, on trouve :

— Les intergiciels orientés base de données (DAM pour *Data Access Middleware*) facilitent les communications entre une application et une base de données, ou entre bases de données. Ces intergiciels permettent par exemple de voir les relations entre les données de la base comme des objets (au sens de la programmation orientée-objets) ce qui permettra d'extraire les données de la base de données et de les

utiliser directement en tant qu'objets. On trouve parmi ces intergiciels JDBC ou encore MySQL.

— Les intergiciels orientés web aident l'utilisateur à naviguer sur internet, utilisent des interfaces qui recherchent les pages susceptibles de l'intéresser et peuvent déterminer les changements d'intérêts de l'utilisateur grâce à son historique de navigation. Les intergiciels orientés web fournissent des fonctionnalités utilisées par de nombreuses applications comme l'authentification, la communication entre processus indépendamment des systèmes d'exploitation ou des protocoles réseaux.

— Les intergiciels bureau permettent de contrôler et d'assister des applications selon le besoin de l'utilisateur. Ils proposent des fonctions d'arrière-plan sans modifier le comportement standard de l'application assistée. On trouve parmi ces intergiciels des fonctionnalités comme la surveillance et le nettoyage de la mémoire, des informations sur l'application, la gestion de fichiers, l'ajout de notifications, etc.

— Les intergiciels temps-réel qui sont principalement utilisés en informatique embarquée. Ces intergiciels prennent en compte la composante temporelle des requêtes qu'ils traitent. Ils peuvent aider le système d'exploitation à répartir les ressources pour permettre aux applications temps réel d'en avoir suffisamment.

— Intergiciels spécialisés. Certains intergiciels ne font partie d'aucune des catégories présentées ci-dessus et répondent à des besoins *ad hoc* très spécifiques.

L'intergiciel que nous présentons se trouve dans la première catégorie, les intergiciels d'intégration puisque notre objectif est de faire communiquer deux applications, en l'occurrence deux simulateurs. Cet intergiciel fait partie de la catégorie des intergiciels orientés objets (type RMI), puisque de nombreux simulateurs microscopiques/mésoscopiques utilisent le paradigme multi-agents et la programmation orientée objet.

Nous nous intéressons plus précisément aux intergiciels orientés objets en étudiant leurs points communs et en détaillant le fonctionnement des trois principaux. Dans [Emmerich and Ellmer2019], les auteurs définissent deux fonctionnalités principales que doivent posséder des intergiciels orientés objets :

— Les difficultés liées à une architecture distribuée doivent être gérées par l'intergiciel et le développeur des applications ne doit pas avoir à s'en occuper, elles doivent donc être transparentes pour ce dernier. L'auteur classe ces types de transparence de la manière suivante :

  i transparence d'accès : l'interface pour envoyer une requête à un service est la même que le service, soit sur le même hôte ou un hôte différent

ii transparence de localisation : on doit pouvoir identifier un service sans connaître son adresse

iii transparence de migration : conséquence de la précédente, si on déplace un service le client ne doit pas avoir besoin de modifier la manière dont il y accède

iv transparence de concurrence d'accès : si plusieurs clients veulent accéder à un service, ce n'est pas au client de gérer cette concurrence

v transparence de la gestion des erreurs : s'il y a des erreurs du côté de l'intergiciel ou du service il ne faut pas que cela impacte le fonctionnement du client

vi transparence de la performance et du passage à l'échelle.

— Les fonctionnalités essentielles. Un intergiciel doit être fondé sur un modèle de composants indépendant des langages de programmation des différents clients ou services. Les objets sont mappés en composants de ce modèle puis traduits en objets du destinataire. Les développeurs doivent être capables de faire des interfaces de ces composants sur des concepts définis dans le modèle. L'intergiciel doit donc avoir un langage dans lequel exprimer ces interfaces et des objets facilement. C'est pourquoi il doit y avoir un IDL (*Interface Description Language*) pour l'intergiciel. L'intergiciel doit également fournir des primitives de communication entre les différents composants et gérer les transmissions synchrones ou asynchrones.

Maintenant que nous avons vu plusieurs classifications d'intergiciels, nous décrivons en détails les trois principaux standards d'intergiciel de communication : CORBA, Java RMI et DCOM.

3.2.1 CORBA

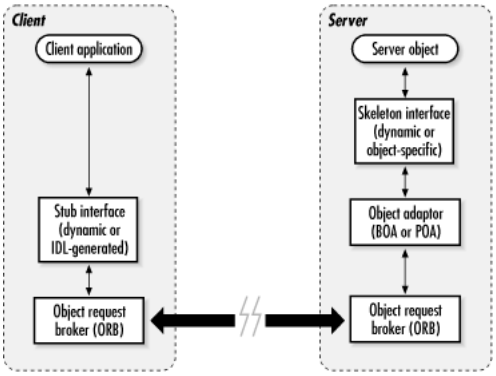

FIGURE 3.2 – Architecture d'une application CORBA

La technologie la plus connue permettant de faire interagir deux parties d'une architecture distribuée est CORBA (*Common Object Request Broker Architecture*) qui a été proposée par l'OMG (*Object Management Group*) réunissant de nombreux industriels. l'ISO (*International Organization for Standardization*) a défini CORBA comme un standard pour les architectures distribuées objet. L'OMG ne propose pas d'implémentation de CORBA mais uniquement des spécifications très précises qui garantissent la standardisation de différents logiciels d'éditeurs différents. Pour qu'une application permette l'accès à certains de ses objets, il faut que ces objets aient une interface utilisable par les autres composants de l'architecture. L'IDL (*Interface description language*) est le langage informatique dans lequel ces interfaces doivent être décrites. Ce langage commun est ce qui permet l'interopérabilité entre différents composants écrits dans des langages différents. L'interface de l'objet contient ses attributs et méthodes et correspond à la partie visible de l'objet. Le serveur peut ensuite l'implémenter dans un langage et le client l'utiliser dans un autre.

CORBA utilise une architecture client/serveur. Le serveur est un programme qui présente une interface avec une liste de méthodes qui peuvent être utilisées par d'autres programmes, ces méthodes sont implémentées sur le serveur et les clients n'ont pas besoin d'en connaître l'implémentation mais uniquement les arguments et le type de retour. Les clients sont des programmes qui appellent des méthodes du serveur.

3.2.2 DCOM

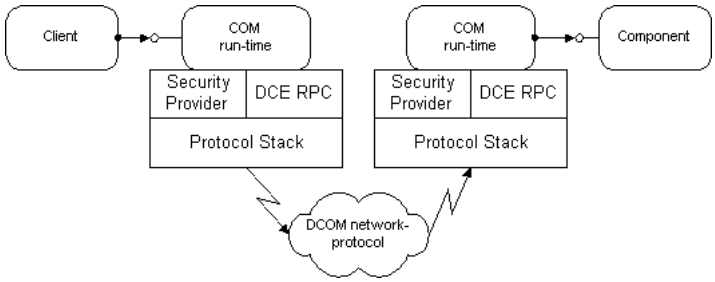

FIGURE 3.3 – Architecture d'une application DCOM

DCOM (*Distributed Component Object Model*) est un équivalent de CORBA développé par Microsoft. Il s'agit donc d'un ensemble de concepts et d'interfaces permettant de faire communiquer plusieurs composants d'une architecture distribuée. Il est fondé sur COM (*Component Object Model*) et permet à ces objets COM d'interagir via un réseau. DCOM a donc été développé après COM pour apporter de nouvelles fonctionnalités permettant la

communication entre applications distribuées et a du pour cela apporter les fonctionnalités suivantes :

— *Marshalling* : sérialiser et dé-sérialiser (transformer un objet en une suite d'octets que l'on peut écrire dans un fichier ou envoyer via un réseau) les arguments et les valeurs de retour d'appels de méthodes. Le *Marshalling* résout le besoin de passer des données d'une instance d'un objet COM à un autre ordinateur.
— La gestion de la mémoire dans un environnement distribué : il s'agit de savoir quand sont effacées les références d'objets utilisés par un programme client si le client perd la connexion au serveur. La gestion de la mémoire est d'autant plus importante dans ce genre d'architectures que les appels au serveur peuvent être nombreux, ce qui résulte en la création d'un grand nombre d'objets. Il faut également s'assurer de communiquer avec les autres serveurs dans la chaîne de transactions pour leur permettre d'effacer ces objets.
— L'envoi d'un grand nombre d'objets au client dans une seule transmission afin de minimiser la bande passante utilisée.

Comme pour CORBA, DCOM est fondé sur l'interaction entre un client et un composant dont certaines méthodes sont publiques et définies dans une interface (cf. figure 3.3). La publication d'interface pour les objets COM se fait à l'aide du langage MIDL fournit par Microsoft. Encore une fois, comme pour CORBA, l'utilisation d'un langage standard permet à des applications ayant été développées dans des langages différents de communiquer. Ainsi, les objets DCOM peuvent être implémentés en C++, VB, etc.

3.2.3 Java RMI

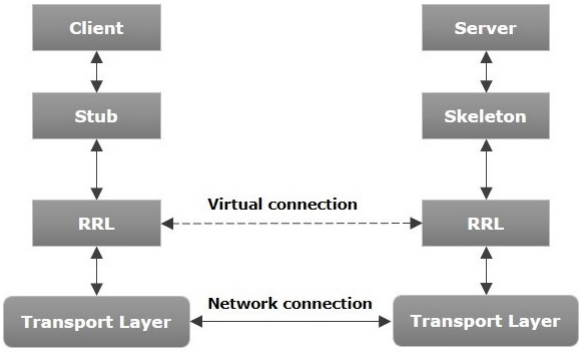

FIGURE 3.4 – Architecture d'une application RMI

Java RMI (*Remote Method Invocation*) est une API pour faire interagir des applica-

tions Java en permettant d'appeler les méthodes de certains objets. Pour interagir avec ses objets, on utilise leurs interface Java. Java RMI est donc spécifique au langage Java contrairement à CORBA qui permet d'échanger entre différents langages. CORBA utilise IDL alors que Java RMI utilise les interfaces Java. Bien qu'on retrouve le concept de *marshalling* commun à tous les intergiciels présentés ici, avec Java RMI il n'y a pas besoin de mapper les données dans une représentation commune aux différents composants. On évite donc des difficultés rencontrées avec CORBA qui, par exemple, transforme un `long` IDL en `int` java, de plus RMI permet de télécharger des classes d'autres programmes java (chargement dynamique de classes) alors que CORBA ne propose pas ce mécanisme de code partagé. Les erreurs et exceptions sont gérées à l'aide du mécanisme de gestion d'exceptions de Java.

3.3 Utilisation d'architectures distribuées pour des simulations multi-échelle

Une architecture distribuée est un ensemble de composants logiciels qui travaillent ensemble sans se trouver sur le même composant physique. Les différents logiciels et les différentes ressources d'une telle architecture sont séparés entre plusieurs machines qu'il faut faire communiquer ensemble. Puisque dans cette thèse nous considérons l'utilisation de plusieurs simulateurs reliés par un intergiciel, nous considérons également la possibilité pour ces simulateurs de se trouver sur des machines différentes. Les simulations de mobilité, par exemple, peuvent se révéler très coûteuse en puissance de calcul et en temps d'exécution. L'informatique distribuée est une solution connue pour assurer la scalabilité d'une application : l'utilisation de plusieurs ordinateurs permettant de stocker plus de données et d'utiliser plus de puissance de calcul. Dans cette section, nous commençons par discuter des standards d'architecture en informatique distribuée puis nous nous intéressons plus précisément aux travaux existants sur les architectures distribuées pensées pour des simulations multi-échelles, sans nous limiter aux simulations de mobilité. Les architectures distribuées sont directement liées aux intergiciels qui peuvent intervenir dans ces dites architectures comme intermédiaires entre les différents composants.

3.3.1 Standards d'architecture distribuées

L'architecture de haut niveau (*High Level Architecture* - HLA) est un standard pour les simulations distribuées dans lequel on retrouve plusieurs composants :

— Les composants fédérés, qui sont les différentes applications qui doivent être couplées et utilisées ensemble. Dans notre cas, il s'agit des simulateurs de mobilité d'échelles différentes.
— L'infrastructure d'exécution (*Run-Time Infrastructure* - RTI) qui permet l'échange d'informations, la synchronisation des composants fédérés ainsi qu'un contrôle plus haut niveau sur le système distribué. Cela correspond à notre intergiciel.
— Le modèle d'objets fédérés (*Federate Object Model* - FOM) qui est un ensemble d'interfaces et de spécifications régissant les interactions entre les différents composants de l'architecture.

Ces trois composants forment une fédération et sont soumis à 10 règles [Setola *et al.*2016] :

1. Chaque fédération doit avoir un FOM, documenté selon *l'Object Model template* (OMT) HLA.
2. Dans une fédération, toutes les instances d'objets de simulation doivent se trouver dans les composants fédérés et non dans la RTI.
3. Durant l'exécution d'une fédération, tous les échanges de données décrits par le FOM entre les composants fédérés doivent se faire via la RTI.
4. Durant l'exécution d'une fédération, les composants fédérés doivent interagir avec la RTI en accord avec les spécifications du FOM.
5. Durant l'exécution d'une fédération, une instance d'objet ne peut être présente que dans un composant fédéré à la fois.
6. Chaque composant fédéré d'une fédération doit avoir un SOM (*Simulation Object Model*), documenté selon l'OMT HLA.
7. Les composants fédérés doivent être capables de mettre à jour ou de renvoyer les attributs d'une instance, tel que spécifié dans leurs SOM.
8. Les composants fédérés doivent être capables de transférer ou d'accepter une instance d'objet de manière dynamique pendant la simulation, tel que spécifié dans leurs SOM.
9. Les composants fédérés doivent pouvoir modifier les conditions sous lesquelles ils mettent à jour leurs attributs , tel que spécifié dans leurs SOM.

10. Les composants fédérés doivent être capables de connaître le temps local de manière à pouvoir coordonner les échanges de données avec les autres composants de la fédération.

HLA n'est pas le seul standard pour les simulations distribuées. *Distributed Interactive Simulation* (DIS), contrairement à HLA n'utilise pas de *middleware* central reliant tous les composants de l'architecture. DIS est plus axé sur la forme des données envoyées entre les différents composants grâce à un protocole d'échange standard. Les simulations communiquent en envoyant des PDU (*Protocol Data Units*) au réseau d'ordinateurs connectés. Le format d'un PDU est spécifié dans les standards du protocole DIS. Chaque PDU possède une entête qui identifie la simulation, la machine et le temps auquel s'est produit tel ou tel évènement. Le corps d'un PDU contient les informations à propos de l'évènement qui s'est produit. Un PDU peut représenter n'importe quel évènement pertinent pour les simulations et doit contenir suffisamment d'informations pour que les autres modèles de simulation puissent traduire cet évènement selon leur modèle et ainsi répliquer cet évènement. Afin d'exécuter un exercice - une simulation - DIS, les modèles sont distribués sur plusieurs machines reliées par un réseau (LAN ou WAN) et ces modèles s'envoient des PDU en utilisant les protocoles UDP/TCP. Les PDU sont envoyés soit pour signaler un évènement important aux autres modèles, soit en temps que « battement de coeur »(*heartbeat*). Ces battements de coeur surviennent à une fréquence constante déterminée avant l'exécution de la simulation, en général toutes les 4 ou 5 secondes. Ces battements permettent d'assurer que tous les modèles dans la simulation sont synchronisés.

3.3.2 Les architectures distribuées appliquées aux simulations multi-échelles

A présent que nous avons vu les bases de l'informatique distribuée, nous allons présenter ici les travaux spécifiques qui se trouvent dans la littérature concernant notre problématique : le couplage de modèles d'échelles différentes dans une architecture distribuée.

Les solutions de couplage présentées ici se fondent souvent sur le MML (*multiscale modeling language*) [Falcone *et al.*2010] qui est un langage de description facilement lisible par l'humain et fondé sur le langage XML. MML permet de décrire l'architecture d'un système de simulation multi-échelles. Ce langage permet entre autres de spécifier la liste des différents modèles, la manière de les coupler, le type de couplage, ou les entrées et sorties de chaque modèle. MML est limité au couplage de modèles d'automates

cellulaires [Falcone *et al.*2010] et de Lattice-Boltzmann [Noël *et al.*2018]. Dans ces types de modèles, la description en MML peut être utilisée pour générer automatiquement le squelette de l'application multi-échelles.

L'idée du langage est que les modèles peuvent être exprimés à l'aide d'une suite d'opérateurs et que le couplage peut donc être exprimé comme des flux de données entre ces opérateurs. Ces données doivent parfois être modifiées d'un modèle à l'autre (changement d'échelle, interpolation, moyenne, etc.). Lorsque plus de deux modèles doivent être couplés alors un nouveau composant est nécessaire, il s'agit d'un élément central faisant le lien entre tous ces modèles.

MUSCLE2 (*Multiscale Coupling Library and Environment*) [Borgdorff *et al.*2014] est un framework permettant d'implémenter le couplage effectif de différents modèles. MUSCLE2 est implémenté en Java mais supporte également d'autres langages comme le C, C++, Python ou Fortran. MUSCLE2 propose une approche modulaire et donc l'implémentation d'un modèle ne modifie en rien la manière dont celui-ci est couplé avec un autre modèle, chaque modèle envoie et reçoit des messages. MUSCLE2 est composé d'une API que les différents modèles utilisent, un environnement de script de couplage qui spécifie comment les modèles vont être couplés et un environnement d'exécution qui exécute les différents modèles sur les différents machines. L'API est indépendante du couplage qui est lui même indépendant de l'environnement d'exécution. MUSCLE2 gère également la communication entre les différentes machines hébergeant les simulations en définissant les ports et les protocoles de communication.

MAPPER [Belgacem *et al.*2013] est un projet européen visant à proposer un formalisme et un framework fondé sur MML et MUSLCE2 afin de pouvoir décrire, implémenter et exécuter des simulations multi-échelles sur des architectures distribuées. L'approche de MAPPER est décomposée en 4 étapes :

1. modélisation et description. La modélisation consiste à décomposer les phénomènes physiques en modèles d'échelles différentes puis à décrire ces différents modèles à l'aide de MML. Afin de faciliter le découpage du problème en échelles, on peut s'aider de l'outil SSM (*Scale Separation Map*). Les différents modèles ne voient pas les échelles des autres modèles et sont informés uniquement par l'arrivée de messages provenant du framework.

2. compilation des descriptions. La compilation des descriptions est l'implémentation des spécifications MML grâce aux outils MaMe (*Mapper Memory*) et MAD (*Mapper*

Application Designer). Cette étape permet de générer les fichiers de configuration et les graphes de workflows.

3. implémentation. L'implémentation de la simulation multi-échelles inclue l'implémentation des modèles eux-mêmes ainsi que l'utilisation de MUSCLE pour le couplage qui s'occupe des communications.

4. exécution. L'exécution est également réalisée grâce à MUSCLE qui lance l'exécution des différents modèles sur leurs machines respectives selon les fichiers de description et qui transmet les données entre les modèles tout en les modifiant pour qu'elles soient pertinentes et utilisables.

3.4 Intergiciels orientés service

L'architecture orientée service (SOA) est un paradigme de programmation informatique ressemblant aux architectures distribuées. Différents composants applicatifs, appelés fournisseurs de services, mettent à disposition un ensemble de services qui peuvent être utilisés par d'autres applications qui sont les consommateurs de services. Cela permet à ces consommateurs de service qui ont besoin de fonctionnalités applicatives d'utiliser des fonctionnalités sécurisées et déjà testées sans avoir à les refaire soi même. La communication entre fournisseurs et consommateurs de ces services est faite à l'aide de requêtes et de réponses qui sont définies par des interfaces de programmation en utilisant le langage WSDL (*Web Service Description Language*).

Dans [Paul *et al.*2005], les auteurs mettent en avant les différences entre la HLA présentée précédemment et les SOA et propose d'utiliser des BOM (*Base Object Model*) à la place des FOM (*Federate Object Model*) afin de faire un premier pas vers une SOA adaptée aux simulations distribuées. Les SOA permettent de réutiliser facilement des applications existantes, essentiellement grâce à un couplage faible des applications, ce qui diffère des architectures comme HLA dans laquelle le FOM décrit très précisément les interactions entre composants et doit être scrupuleusement respecté. En revanche les BOM [Gong *et al.*2006] décrivent des composants réutilisables en différenciant totalement l'interface de l'implémentation et du fonctionnement. On peut trouver plusieurs frameworks orientés services pour l'informatique distribuée.

DDSOS (*Dynamic Distributed Service-Oriented Simulation*) [Tsai *et al.*2006] est un framework multi-agents distribué orienté service. Ce framework propose plusieurs fonc-

tionnalités telles que le gestion de la configuration dynamique d'une fédération de simulations, la génération de code automatique, le déploiement de code automatique, la reconfiguration dynamique de simulations multi-agents ou l'analyse de la simulation. DDSOS est fondé sur le PSML-S (*Process Specification and Modeling Language for Services*). Les propriétés dynamiques de DDSOS sont possibles car il est fondé sur la MDA (*Model Driven Architecture*). Le code de simulation peut être automatiquement généré, déployé et exécuté une fois le modèle PSML-S créé et configuré.

Dans [Shekhar *et al.*2016], les auteurs présentent SIMaaS, un intergiciel orienté service qui considère les différents simulateurs comme des services et gère les différentes données et ressources des simulations qui s'exécutent en parallèle. On peut également trouver DUNIP (*DEVS unified process framework*) présenté dans [Mittal2007] pour le développement et le test d'architectures orientées services. Les auteurs proposent d'utiliser les DEVS (*Discrete EVent systems Specification*) comme seules spécifications et des méthodes pour générer des modèles DEVS depuis plusieurs autres formalismes. De plus, ils proposent une plateforme de services de simulation pour résoudre les problématiques de compatibilité entre DEVS et C++, DEVS et Java, DEVS et RMI. Dans [Sarjoughian *et al.*2008], les auteurs présentent un framework de SOA compatible avec les DEVS (SOAD) qui est repris ensuite et amélioré par [Muqsith *et al.*2011].

Après ces différents exemples de frameworks ou d'intergiciels pour les architectures orientées services pour l'informatique distribuée, il convient de s'intéresser aux points communs et aux caractéristiques nécessaires aux intergiciels orientés services. Dans [Al-Jaroodi and Mohamed2012a], les auteurs proposent un état de l'art de ces intergiciels et mettent en avant quelques points importants. On y retrouve évidemment l'interface commune qui doit être utilisée et implémentée par les différents composants de l'architecture, mais également l'idée qu'une API doit faciliter l'implémentation de cette interface. L'API est composée de différentes bibliothèques permettant aux consommateurs et aux fournisseurs de services d'utiliser les mêmes données. Une fois que les interfaces et les interactions entre les fournisseurs et services sont définies, les nouveaux services doivent être automatiquement vérifiés par l'intergiciel en étant comparés aux modèles. C'est le rôle de l'intergiciel de s'assurer que chaque nouveau service est facilement utilisable par les consommateurs. Dans [Wang *et al.*2011], on retrouve également l'idée que l'intergiciel a pour rôle de rechercher les services adaptés au consommateur selon les paramètres fournis par ce dernier, puis de les classer de manière à proposer un classement de services selon leur pertinence.

Les auteurs dans [Al-Jaroodi and Mohamed2012b] décrivent 9 principales fonctionnalités requises d'un intergiciel orienté service :

1. L'intergiciel doit proposer aux fournisseurs de services une API standard afin de construire et de publier leurs services. Cette API sert à palier l'hétérogénéité des différents composants. On retrouve cette contrainte chez les intergiciels classiques (IDL, interfaces java, etc.)
2. L'intergiciel doit fournir des outils adaptés à la publication des services par les fournisseurs ainsi que des outils pour mesurer l'accessibilité et la fiabilité des services ainsi déployés.
3. L'intergiciel doit fournir aux clients des outils pour explorer l'ensemble des différents services disponibles et pour trouver les plus adaptés à leurs besoins. Il doit également s'assurer que l'intégration des services par le client se fait facilement.
4. L'intergiciel doit permettre de faire communiquer ensemble des services et des clients hétérogènes. Écrits avec des langages différents, dans des systèmes d'exploitation différents.
5. L'intergiciel doit s'assurer de la transparence lors de l'intégration d'un service par le client. Idéalement, le client ne doit voir qu'un ensemble de fonctionnalités sans avoir besoin de les paramétrer et en obtenant le résultat qui lui convient le mieux.
6. L'intergiciel doit être capable de découvrir et d'intégrer automatiquement et de manière adaptative des services pertinents pour le client en cas de service qui n'est plus disponible ou qui a une erreur. Le client doit en permanence être assuré de la disponibilité des services qu'il utilise et l'intergiciel doit être capable de remplacer efficacement ceux défectueux.
7. L'intergiciel doit être capable de gérer de grands nombres de connexions et d'important flux de données.
8. L'intergiciel doit permettre des communications et des transactions sécurisées. En effet, les applications clientes font appel à d'autres applications qui agissent comme des boites noires. Il faut donc que l'intergiciel assure que ces boites noires sont bien dignes de confiance. Il y a donc des contraintes de sécurité autant au niveau de la communication qu'au niveau fonctionnel.
9. L'intergiciel doit pouvoir assurer une qualité de service (QoS) si certaines applications clientes ont des besoins particuliers. Ainsi, comme dans toute solution pour une architecture distribuée, l'intergiciel doit proposer dans son API de définir des niveaux de qualité de service.

Après avoir défini les fonctionnalités attendues d'un intergiciel orienté service, les auteurs dans [Al-Jaroodi and Mohamed2012b] décrivent brièvement 14 intergiciels et vérifient s'ils possèdent ces fonctionnalités. La plupart des intergiciels vérifient les prérequis 2,3 et 4. Les autres sont à peu près également répartis (30 à 40 % des *middlewares* vérifiés). L'auteur considère que les trois premières fonctionnalités sont importantes et que les suivantes sont certes des plus values, mais non nécessaires. Finalement, l'auteur conclue que certaines de ces fonctionnalités sont contradictoires entre elles. L'idée est donc qu'il trouver le juste équilibre entre toutes ces fonctionnalités en les prenant toutes en compte.

3.5 Conclusion

Puisque nous proposons une solution logicielle sous forme d'intergiciel pour les simulations multi-échelles, ce chapitre a étudié cet aspect de la problématique. Les caractéristiques attendues des architectures distribuées et des intergiciels, énumérées dans ce chapitre seront reprises dans notre solution. Des caractéristiques telles que la synchronisation des simulations, la possibilité de faire fonctionner ensemble des simulations hétérogènes ou l'utilisation facilitée par une interface utilisable par n'importe quel simulateur, sont toutes prises en compte dans nos contributions, détaillées dans la partie suivante de cette thèse.

Deuxième partie

Contributions

Chapitre 4

Modèle d'intergiciel pour les simulations multi-échelles

Sommaire

4.1 Introduction

Cette partie entame la présentation des contributions de cette thèse, en se fondant sur les conclusions de l'étude de la littérature sur les différents sujets abordés. Dans l'état de l'art, nous avons identifié un manque de clarté et de rigueur dans la définition des échelles dans les simulations de mobilité. Dans ce chapitre, nous commençons donc par présenter une méthodologie pour classifier les simulateurs selon leurs caractéristiques et déterminer le type de couplage entre ces simulateurs. Nous discutons ensuite de la question de la validation des simulateurs de mobilité, notion nécessaire pour la correction des simulateurs les uns les autres. Puis, nous présentons le modèle d'intergiciel pour le couplage de simulateurs, en nous focalisant successivement sur les aspects qui doivent être traités. Le modèle servira à l'implémentation d'un outil informatique avec lequel deux simulateurs peuvent être connectés pour obtenir une simulation multi-échelles.

Les objectifs de cet intergiciel sont de :

1. Permettre à deux simulateurs de mobilité d'échelles quelconques de communiquer et d'interagir en transmettant les données d'un simulateur à l'autre et en les traduisant d'une échelle à l'autre.
2. Assurer la cohérence de la simulation multi-échelles en permettant aux deux simulateurs de se corriger mutuellement à chaque pas de temps.
3. Faciliter l'intégration des simulateurs à l'architecture proposée, en étant le moins intrusif possible dans les simulateurs pré-existants.

Afin d'atteindre ces objectifs, comme pour toute simulation il faut tout d'abord définir le scénario d'utilisation de cet outil. Lors d'un couplage de simulateurs de mobilité, il faut définir les zones géographiques simulées par chaque simulateur ; les simulateurs peuvent simuler la même zone, deux zones distinctes en interaction, etc.

4.2 Vue d'ensemble

Un modèle de mobilité est une représentation abstraite d'un système de mobilité. Une simulation de mobilité est une application d'un modèle de mobilité particulier pour visualiser son comportement sur une période donnée. Un simulateur est un programme informatique qui effectue des simulations [Lok and Brent2005]. Un simulateur peut être

décrit comme une fonction $y = f(u)$ avec :

— $u = \{M_{od}, G\}$ les paramètres d'entrée, avec M_{od} la matrice origine-destination et G le graphe de transport.
— y les données de sortie du simulateur $= (G, t_i)_{i=0...n}$, qui montre les états successifs de G à travers le temps.

L'intergiciel dont nous détaillons les fonctionnalités dans ce chapitre s'intercale entre deux simulateurs, et est décrit par une image de f, prenant en entrée les sorties des simulateurs $(G, t_i)_{i=0...n}$ et fournissant aux simulateurs de nouvelles origines-destinations et une nouvelle définition de G. Les seuls leviers pour l'intergiciel pour influencer le comportement des simulateurs à composer sont les définitions des graphes et les voyageurs à déplacer (quelle que soit leur représentation).

Nous proposons une description abstraite des caractéristiques d'un simulateur sous la forme suivante :

$$\mathcal{S} = \langle \omega, \sigma, \theta, \mathcal{P} \rangle$$

avec ω la représentation interne des entités (flux, groupes ou individus), σ la représentation spatiale de G (détaillée ou agrégée) et θ l'échelle temporelle (secondes, minutes, heures) et qui détermine l'écart entre chaque couple t_i, t_{i+1} dans u. La section 4.4 détaille notre définition des échelles. $\mathcal{P}$ représente le processus du simulateur, qui est décrit ainsi :

$$\mathcal{P} = \{\texttt{Demand}, \text{Assign}, \text{Move}\}$$

`Demand` spécifie la manière avec laquelle la demande dans M_{od} est traitée, `Assign` spécifie la manière avec laquelle la demande est affecté sur G et `Move` définit la manière avec laquelle les entités se déplacent sur G (principalement leur vitesse dynamique).

Soit l'opérateur $\bigotimes$ permettant de coupler deux simulateurs. Il est défini ainsi :

$$\mathcal{S}_1 \bigotimes \mathcal{S}_2 = \langle \omega_1 \circ \omega_2, \sigma_1 \bullet \sigma_2, \theta_1 \diamond \theta_2, \mathcal{P}_1 \oplus \mathcal{P}_2 \rangle$$

Avec $\omega_i, \sigma_i, \theta_i, \mathcal{P}_i$ les propriétés et processus du simulateur i. Dans notre proposition, l'opérateur $\bigotimes$ est implémenté par un intergiciel. La suite de la présentation s'attachera à

la manière de définir les opérateurs $\circ$, permettant la conciliation des représentations des entités, $\bullet$ permettant de concilier les représentations spatiales, $\diamond$ permettant la conciliation des pas de temps des simulations et enfin $\oplus$, permettant la composition des processus des deux simulateurs. Chaque définition d'opérateur repose sur des hypothèses et a des contraintes propres, que nous définissons au fur et à mesure de la présentation. Notamment, l'opérateur $\oplus$ repose sur une connaissance *a priori* du simulateur qui corrige les comportements `Demand`, `Assign`, `Move` ; nous discutons de la validité des simulateurs dans la section 4.6.

4.3 Hypothèses principales

Les fonctionnalités de l'intergiciel, exprimées sous forme d'opérateurs, décrites dans ce chapitre repose sur quelques hypothèses décrites dans ce qui suit.

Hypothèse 1 (Entrées dynamiques) *Les simulateurs à composer permettent de fournir des données en cours d'exécution. Plus précisément, ils permettent de recevoir de nouveaux inputs à chaque pas de temps simulé.*

Hypothèse 2 (Contrôle de simulateur) *Certaines fonctionnalités de l'intergiciel sont applicables aux seuls simulateurs permettant un contrôle des simulateurs. En effet, certaines corrections ne sont possibles que si nous pouvons arrêter momentanément une simulation, ou relancer un pas de temps plusieurs fois.*

La première hypothèse est nécessaire, car si les simulateurs ne permettent pas d'intervention pendant l'exécution, l'intergiciel n'a aucun levier pour influencer leur comportement. La seule utilisation conjointe des simulateurs dans ce cas revient à les utiliser en séquence, ce qui n'a pas beaucoup d'intérêt dans le cadre de cette thèse. La deuxième hypothèse est nécessaire si les différences entre les représentations des simulations ne peuvent être conciliées qu'en contrôlant les simulations. Heureusement, un grand nombre de simulateurs sont *open source*. Ainsi, même si certains ne permettent pas ces accès et contrôle par conception, ils peuvent être modifiés pour l'être. Ces modifications concernent les interactions avec les simulateurs et ne modifient pas le cœur de leur fonctionnement.

4.4 Échelles de simulation : ω, σ et θ

Comme nous l'avons vu dans l'état de l'art, les échelles les plus utilisées pour décrire les simulateurs de déplacement sont les échelles microscopique, mésoscopique et macroscopique qui sont respectivement l'échelle la plus détaillée, l'échelle intermédiaire et l'échelle la moins détaillée. Ces échelles ne sont pas rigoureusement définies et la classification peut varier d'un article à un autre, selon le contexte. Un simulateur noté comme microscopique dans un travail peut être considéré comme mésoscopique dans un autre. Dans cette section, nous proposons de voir comment ces échelles peuvent être définies et si elles sont pertinentes pour décrire tous les simulateurs de mobilité. Notre objectif est de réussir à faire ressortir les principales différences entre deux simulateurs selon leur échelle et ce pour déterminer la manière avec laquelle l'intergiciel devra traduire les données pour passer d'un simulateur à l'autre.

Dans [Ni2011], les auteurs donnent une définition des échelles pour les simulations de mobilité, qui donnent quatre modalités possibles (les trois échelles évoquées plus haut et une quatrième : l'échelle picoscopique). Elles sont décrites ainsi :

- L'échelle macroscopique : cette échelle utilise des flux de trafic et ne se sert que de variables agrégées comme la densité des véhicules et leur vitesse moyenne sur un arc sur réseau. Les mouvements latéraux comme les changements de voie ne sont pas modélisés.
- L'échelle mésoscopique : cette échelle fonctionne également avec des flux de trafic mais utilise également des fonctions de probabilité pour déterminer la possibilité de l'existence d'un véhicule à une position x, un temps t et une vitesse v.
- L'échelle microscopique : cette échelle représente les véhicules individuellement avec leur propre trajectoire et vitesse. La position est déterminée sur la route mais le mouvement latéral n'est pris en compte que par la voie sur laquelle le véhicule se trouve. Le comportement des véhicules est modélisé par un modèle de poursuite ou par un modèle de transmission cellulaire.
- Picoscopique : Cette échelle représente les véhicules individuellement avec leur propre trajectoire et vitesse. Contrairement à l'échelle microscopique la vitesse et la trajectoire d'un véhicule sont en deux dimensions, sur l'axe de la route et latéralement. Dans les modèles les plus détaillés le véhicule et le conducteur sont même deux entités distinctes qui interagissent ensemble.

Dans [Bourrel and Lesort2003], l'auteur propose de ne pas utiliser ces échelles figées,

mais de considérer uniquement les simulateurs en fonction de certaines de leurs caractéristiques. Il considère que la représentation des entités peut être soit individuelle soit décrite comme un flux, il considère également que le modèle de déplacement utilisé peut être un modèle individuel ou un modèle de groupe et les simulateurs sont ensuite décrits en considérant ces deux caractéristiques. Cette classification est en fait critiquable puisque la représentation des voyageurs et leur modèle de comportement sont liés. En effet, même si un modèle de déplacement de groupe peut être utilisé pour le déplacement d'individus, un flux de voyageurs ne peut pas suivre de modèle individuel. Cependant, l'idée de classifier les simulateurs selon leurs caractéristiques est intéressante et c'est ce que nous nous proposons de faire en définissant trois axes d'agrégation qui sont trois dimensions d'analyse et qui permettent de définir l'échelle d'un simulateur : la représentation des voyageurs, le temps et l'espace.

1. La représentation des voyageurs (ω). Trois modalités principales sont définies pour cette échelle : microscopique (représentation individuelle), mésoscopique (représentation par groupes d'individus) et macroscopique (représentation par flux). Les modèles dits macroscopiques fonctionnent avec des flux de voyageurs en utilisant des variables agrégées alors que les simulateurs dites microscopiques représentent les voyageurs individuellement, en prenant en compte des variables individuelles comme la position, la vitesse ou l'accélération de chaque voyageur. Entre ces deux représentations on peut trouver les groupes de voyageurs agrégés, chaque agent de la simulation représente plusieurs voyageurs qui se déplacent ensemble.
2. L'espace (σ). Plusieurs modalités sont possibles pour cette dimension. On peut se référer aux niveaux de détails (LoD pour *Level of Detail*) du standard OGC CityGML [Gröger *et al.*2012]. Les auteurs dans [Beil and Kolbe2017] précisent le standard pour la modélisation dans le domaine des transports. Les auteurs proposent à bon escient de supprimer le LoD4 de CityGML, étant donné que le LoD4 est destiné à la représentation des structures intérieures dans bâtiments et son application pour les routes est absurde. Peu de simulateurs se réfèrent explicitement au standard CityGML, nous décidons donc de nous référer plutôt à deux modalités principales utilisées en simulation et en affectation du trafic : soit une représentation très détaillée des réseaux (y compris les restrictions directionnelles, i.e. sens-interdit, interdictions de tourner, etc.), soit une représentation agrégée, qui ne représentent que les principaux axes de circulation.
3. Le temps (θ). Nous identifions trois modalités principales correspondant aux ordres de grandeurs de la dimension temporelle représentée. L'ordre des secondes corres-

pond aux simulations de déplacement très détaillées, l'ordre des minutes correspond aux modèles d'affectation dynamique, et l'ordre des heures correspond à celui des modèles d'affectation statique (considérant généralement les heures de pointe).

Les « voyageurs »dans cette thèse correspondent à tous les véhicules (véhicules particuliers ou de transports en commun) et aux piétons, passagers ou conducteurs. Il s'agit donc de toutes les unités mobiles d'une simulation. Contrairement à [Bourrel and Lesort2003], nous avons décidé de ne pas prendre en compte le modèle de comportement en tant qu'axe d'agrégation car il dépend directement des trois axes ci-dessus. En effet, la représentation en flux empêche les comportements individuels, un pas de temps trop long n'est pas adapté à certains modèles car la prise en compte d'interaction entre les agents est par exemple impossible et un modèle de déplacement en deux dimensions est impossible si la représentation spatiale est en une dimension. Le modèle de déplacement est donc directement lié aux trois axes d'agrégation présentés ci dessus.

Au lieu d'avoir trois ou quatre classes de simulateurs, nous avons donc trois dimensions avec plusieurs modalités possibles. Il est donc virtuellement possible de créer 18 classes de simulateurs possibles ($3 \times 2 \times 3$) sur la base de ces dimensions, qui correspondent à toutes les combinaisons possibles de ces dimensions.

Cette classification selon trois caractéristiques permet donc de décrire le fonctionnement d'un simulateur et met en avant la représentation des données dans celui-ci. Ainsi en plus d'éviter toute ambiguïté lors de la description, elle est la première étape de la méthodologie proposée qui va permettre à l'intergiciel de faire le lien entre deux simulateurs en traduisant les données d'une représentation à une autre.

4.5 Les processus : $\mathcal{P}$

Nous proposons une abstraction des simulateurs de mobilité, fondée sur trois principales fonctions, inspirées des modèles à quatre étapes. L'étape 1 (génération des déplacements), l'étape 2 (distribution des déplacements) et l'étape 3 (choix modal) sont fusionnés dans une seule fonction `Demand`, puisqu'un simulateur dispose généralement de matrices origines-destinations en entrée. La seconde fonction est l'affectation `Assign` (quel chemin empruntent les voyageurs). La troisième fonction est le déplacement `Move`. Lors de la composition d'un couple de simulateur par l'opérateur $\oplus$, les trois fonctions doivent être composées.

— La gestion de la demande (fonction `Demand`). Il s'agit de la manière avec laquelle les origines et destinations des voyageurs dans M_{od} sont gérés. Lors du couplage de simulateurs $\mathcal{S}_1$ et $\mathcal{S}_2$, les deux fonctions $\texttt{Demand}_1$ et $\texttt{Demand}_2$ doivent être conciliées. Dans notre exemple, le simulateur régional alimente la zone d'intérêt avec des voyageurs qui viennent de toute la région et qui passent par cette zone. Les voyageurs et le moment de leur arrivée sur le quartier pourraient être le résultat d'une affectation effectuée par le simulateur régional. De son côté, le simulateur local pourrait avoir sa propre définition de la demande, issue par exemple d'une enquête effectuée uniquement sur la zone d'intérêt. Nous avons donc deux demandes pour la zone d'intérêt, les origines-destinations propres au simulateur local et les voyageurs qui passent par cette zone dans le simulateur régional. Ces deux demandes peuvent ne pas être identiques et il faudra déterminer laquelle est la plus pertinente l'un ou l'autre des simulateurs.

— L'affectation (fonction `Assign`). Étant donnée une demande, l'affectation consiste en la détermination du chemin emprunté par les voyageurs. Chaque simulateur peut avoir sa propre méthode d'affectation des voyageurs et les chemins déterminés peuvent être différents d'un simulateur à l'autre. L'affectation doit être cohérente entre les deux simulateurs.

— La vitesse des voyageurs (fonction `Move`). La vitesse renseigne le temps que mettent les voyageurs pour parcourir chaque arc de l'itinéraire auquel ils sont affectés. Les voyageurs doivent non seulement emprunter le même chemin sur les deux simulations, mais également le parcourir à la même vitesse pour se retrouver au même endroit au même moment dans les deux simulations, et ce quelques soient les modèles de déplacement utilisés par les simulateurs.

4.6 Validité des simulations

L'intergiciel que nous proposons pour coupler des simulations multi-échelles repose entre autres sur l'hypothèse d'une connaissance *a priori* concernant le simulateur qui aurait le comportement le plus « valide », i.e. celui dont les résultats sont plus corrects. Cette information est essentielle afin de résoudre les conflits de comportements entre simulateurs, et de savoir quel simulation corrige l'autre.

Cette information dépend en grande partie de la pertinence des données manipulées. Par exemple, un simulateur local, travaillant avec des données locales fines, est *a priori*

plus pertinent pour la représentation des comportements locaux qu'un simulateur régional, travaillant avec des données régionales. Mais la validité d'un simulateur dépend également de la validité du modèle sous-jacent. Si l'un des deux modèles n'a pas été correctement calibré et validé, il faudrait que les comportements induits par le simulateur correspondant soient considérés, quand bien même les données manipulées sont moins pertinentes.

La calibration des simulations des déplacements a reçu une attention variable dans la littérature, avec une exception notable avec l'action COST Européenne Multitude [1]. C'est pourtant un aspect d'une importance capitale dans les simulations. En effet, la propriété la plus importante d'un modèle de simulation est sa validité [Klügl2008]. Seul un modèle suffisamment valide est capable de produire des résultats fiables. Fondamentalement, la validité signifie que le bon modèle est utilisé [Balci1994]. Ce n'est que si le modèle est valide que les réponses dérivées de sa simulation peuvent être considérées comme des réponses à des questions adressées au système d'origine. La validation est alors généralement définie comme « le processus consistant à déterminer si un modèle de simulation est une représentation exacte du système, pour les objectifs particuliers de l'étude » [Law2005]. Il existe trois déclarations générales et essentielles sur la validité et la validation que l'on peut trouver dans de nombreux manuels sur la simulation :

1. Il n'y a pas de validité « absolue ». Cette affirmation est justifiable de deux points de vue. En pratique, il y a un compromis à faire pour garantir l'exactitude et les efforts investis dans la collecte de données, les tests, etc. Le rapport coût-efficacité est une question importante. Sur le plan conceptuel également, étant donné que l'abstraction et les simplifications constituent les principaux ingrédients du processus de modélisation, un modèle ne peut jamais être pleinement représentatif du système original. Cela est encore plus vrai pour les modèles de systèmes qui n'existent pas encore.
2. Un modèle de simulation doit être développé pour un ensemble particulier d'objectifs. La validité d'un modèle dépende de ces objectifs.
3. La validation doit être faite tout au long du cycle de vie complet de l'étude de simulation. Depuis les débuts de l'élaboration d'un modèle conceptuel jusqu'à l'analyse des résultats de la simulation, la validité de la représentation du modèle doit être assurée [Balci1994].

La validité d'un certain modèle n'est pas une propriété binaire, bien que les tests statistiques aboutissant à l'acceptation ou le rejet induisent à cette idée. La validation

1. Methods and tools for supporting the Use caLibration and validaTIon of Traffic simUlation moDEls

peut être une procédure coûteuse et peut être améliorée en investissant de plus en plus de temps pour tester et adapter le modèle. Ã un moment donné, un investissement supplémentaire n'entraînerait qu'une très faible amélioration de la validité. Ã ce stade, le modélisateur doit se demander si le niveau de validité atteint jusqu'à présent est suffisant pour l'objectif de l'étude de simulation. Le niveau de validité qui est possible en principe dépend clairement de la forme et de la technologie de validation utilisées. Il est évident que le niveau de validité le plus élevé n'est possible que si deux techniques de validation, formelle et informelle, ont été appliquées avec succès.

Vérifier la validité d'un modèle par rapport à une réalité observée correspond à l'équation :

$$Prob(|\text{Simulation} - \text{realité}|) \leq d > \alpha$$

Simulation correspond aux valeurs de sortie u de la simulation, realité correspond aux valeurs observées dans la réalité, d correspond à l'écart tolérable et α est le degré de confiance. Idéalement, les valeurs d et α ont été déterminées au préalable par l'analyste.

Les étapes pour valider un modèle de déplacements peuvent être résumées ainsi (adaptation de [Klügl2008]) :

1. ***Face validation*** : il s'agit d'une évaluation immersive, de l'animation et des résultats par un expert, pour juger de la crédibilité en première analyse.
2. Définir les paramètres P du modèle qui doivent être calibrés par **Analyse de sensibilité**. Il s'agit de détecter, parmi les paramètres du modèles, ceux qui ont un impact significatif sur les résultats, et qui nécessiteraient un calibrage.
3. Définir deux sous-ensembles (*training set* et *test set*) des données complètes d'entrée réelles ou observées.
4. Utiliser le *training set* pour calibrer les valeurs des paramètres d'entrée en P (**calibrage**)
5. **Validation statistique** en utilisant *test set*.

Seul un modèle pour lequel ces étapes ont été exécutées est considéré « valide ». Dans la suite de ce manuscrit, lorsqu'on suppose qu'une simulation corrige une autre, cela signifie :

— soit que le simulateur correcteur utilise des données plus pertinentes que le second simulateur,
— soit que le simulateur correcteur a été validé et que le second ne l'a pas été (ou l'a été avec un α inférieur).

4.7 Synchronisation des simulateurs : l'opérateur ∘

Deux simulateurs $\mathcal{S}_1$ et $\mathcal{S}_2$ travaillant en concurrence et à des échelles différentes θ_1 et θ_2 doivent être synchronisés. Le pas de temps d'un simulateur est le plus petit intervalle de temps entre deux états du simulateur ($t_i - t_{i-1}$ dans u). Par exemple, si un simulateur peut montrer l'état du réseau de transport à 9 :00 et que le prochain état de la simulation représente 9 :10 alors il a un pas de temps de 10mn. Un pas de temps est atomique et ne peut pas être divisé. Dans cet exemple, nous ne pouvons pas voir l'état du réseau de transport à 9 :05 si notre pas de temps est de 10mn.

Afin de pouvoir exécuter un pas de temps t_i, un simulateur a besoin de connaître :

i la demande (la part de M_{od} qui est en train d'être simulée à l'instant t_i),

ii l'affectation de ces voyageurs c'est à dire le chemin qu'ils vont emprunter de leur origine à leur destination.

Une fois ces données connues et que les nouveaux voyageurs ont été créés dans la simulation, le simulateur peut alors calculer la vitesse des voyageurs et les déplacer pour finalement arriver au nouvel état du réseau à la fin du pas de temps (G, t_{i+1}).

Soit $\mathcal{S}_1$ et $\mathcal{S}_2$ deux simulateurs de mobilité, avec $\theta_1 > \theta_2$. L'opérateur ∘ est implémenté par l'intergiciel en sauvegardant les données de sortie u_2 du simulateur $\mathcal{S}_2$ jusqu'à atteindre le prochain pas de temps de $\mathcal{S}_1$. A ce moment-là, les différentes sorties sauvegardées sont composées et fournies en entrée au simulateur $\mathcal{S}_1$. Les sorties u_1 sont également récupérées, et fournies à $\mathcal{S}_2$ au premier pas de temps possible à $\mathcal{S}_1$.

La manière de composer les u_2 reçues au fur et à mesure de la simulation dépend des différentes corrections à apporter aux trois fonctions `Demand`, `Assign`, `Move` et sera décrite lors de la description de l'opérateur $\oplus$ (cf. section 4.10).

Selon la volonté du modélisateur et les simulateurs particuliers considérés, la synchronisation peut aller au-delà du buffer que nous venons de décrire. En effet, lorsque le deux simulateurs se corrigent mutuellement (un simulateur corrigeant l'affectation et l'autre la demande par exemple), il peut s'avérer nécessaires d'exécuter l'un des simulateurs plusieurs fois. Dans ce cas, la synchronisation des simulateurs s'apparente plus à un ordonnancement de simulateurs. Ce cas sera également traité en section 4.10.

4.8 Composition de représentations de voyageurs : l'opérateur •

Soit deux simulateurs $\mathcal{S}_1$ et $\mathcal{S}_2$ avec deux représentations ω différentes. L'opérateur • transforme les deux représentations ω_1 et ω_2 les unes vers les autres.

Nous prenons en compte trois types de représentation : les flux de voyageurs, les agents représentant des voyageurs individuels et les agents représentant des groupes de voyageurs. Nous considérons deux transformations bidirectionnelles : conversion des agents en flux (et réciproquement) et (dés)agrégation des groupes d'agents - Figure 4.1

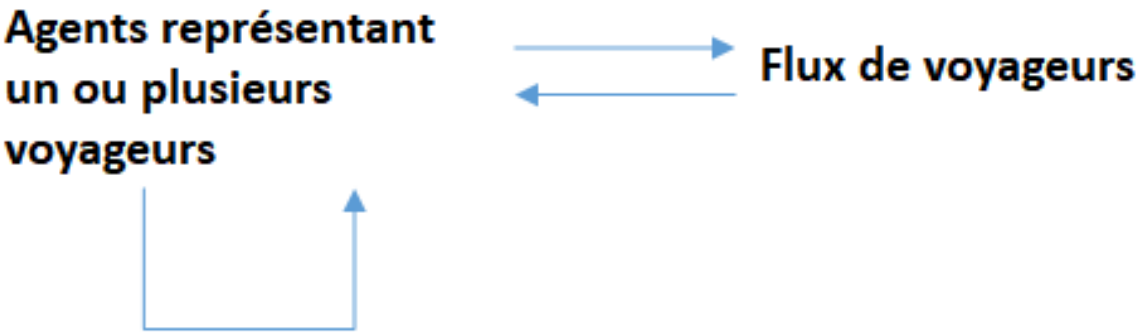

FIGURE 4.1 – Deux transformations bidirectionnelles

Pour la transformation entre flux et agents, dans [Sewall *et al.*2011] l'auteur propose de passer dynamiquement du flux aux agents dans un simulateur multi-niveaux. Cela ne correspond pas exactement à notre cas mais les mêmes transformations peuvent être utilisées pour obtenir la bonne représentation des voyageurs. La transformation la plus simple est le passage d'agents aux flux puisque cela correspond à calculer une densité et une vitesse moyenne. Ce calcul de moyennes est modifié si l'on prend ou non en compte la longueur des véhicules et si l'on souhaite calculer la moyenne pour chaque partie d'un arc ou sur l'arc tout entier. Nous reprenons l'approche de [Sewall *et al.*2011], qui traite le cas le plus complexe où la densité est calculée pour chaque partie de l'arc en prenant en compte la longueur des véhicules :

— Pour chaque véhicule, une fonction de x est définie : si le véhicule est présent à cette position x alors la fonction retourne une constante positive, sinon 0.
— Ensuite, toutes les fonctions définies précédemment sont sommées pour créer une nouvelle fonction. Cette nouvelle fonction retourne donc une constante positive pour chaque position x ou un véhicule est présent, 0 si aucun véhicule n'est présent. Cette fonction est appelée $D(x)$.

— Enfin, si les cellules du modèle sont espacées de Δx nous obtenons la densité du trafic sur les parties de l'arc, nécessaire au modèle macroscopique avec la fonction suivante :

$$\rho_k = \frac{1}{\Delta x} \int_{k\Delta x}^{(k+1)\Delta x} D(x)\mathrm{d}x$$

Pour le passage de la représentation agrégée à la représentation individuelle, il est nécessaire de créer de l'information pour la transformation des flux en agents. Un simulateur multi-agents utilise des données individuelles qui ne sont pas présentes dans un simulateur utilisant des flux, il faut alors générer ces données. Pour cela, les auteurs dans [Sewall *et al.*2011] proposent une méthode basée sur le processus de Poisson. Cela consiste à créer des agents à des positions déterminées par des probabilités. La création d'agents voyageurs depuis des flux se fait donc avec de la création d'information - comme la position des agents - à l'aide de fonctions probabilistes.

Le seconde transformation est l'agrégation d'agents en de nouveaux agents représentant des groupes de voyageurs. Les auteurs dans [Navarro *et al.*2013] proposent de déterminer quels agents agréger en définissant des fonctions de distance entre ces agents et d'agréger les plus proches. Ces fonctions de distance doivent prendre en compte deux types de paramètres : la distance physique qui est le chemin dans la simulation entre deux agents et la distance psychologique qui permet de déterminer le comportement futur des agents. Cette distance psychologique peut être calculée avec des facteurs comme le chemin que les agents vont emprunter, leur profil, leur but, leur réaction par rapport à un évènement, etc. Deux agents physiquement proches ne peuvent pas être agrégés si la probabilité qu'ils restent proches dans le futur est trop faible. Un autre point à prendre en compte est l'importance des voyageurs par rapport au scénario de la simulation. Si la simulation comporte des évènements particuliers comme la perturbation d'une ligne de transport alors les voyageurs les plus affectés par ces évènements doivent être moins agrégés que les autres afin de garder un maximum de détails sur les conséquences du scénario de simulation. Dans notre cas, les voyageurs passant par la zone d'intérêt seraient donc moins agrégés que ceux qui restent dans le reste de la région.

A présent que nous savons quels agents agréger, nous devons déterminer comment les agréger. Comme pour la transformation d'agents en flux, cela consiste, pour la plupart des paramètres, à en faire une moyenne. Cependant, les auteurs dans [Navarro *et al.*2011] soulignent le fait que tous les paramètres ne peuvent pas être agrégés de cette manière puisque certaines ressources, comme l'argent que possèdent les voyageurs par exemple,

doivent plutôt être sommées. La transformation inverse est la désagrégation de ces agents représentant des groupes de voyageurs en de plus petits groupes, voire en agents individuels. Tout comme pour la transformation de flux en agents, si la première transformation consistait à agréger l'information il faut cette fois en créer et donc se fonder sur des probabilités. L'opérateur • utilise également une fonction mémoire, qui permet de faciliter la désagrégation de groupes précédemment agrégés (comme dans [Navarro *et al.*2013]).

4.9 Composition de représentations spatiales : l'opérateur ◇

Toutes les simulations ne représentent pas de la même manière le réseau de transport (σ). On ne peut pas retrouver le même niveau de détail d'une simulation de quelques arcs routiers à une simulation de toute une région. Dans certaines simulations lorsque la zone représentée est importante, le réseau de transport doit être simplifié. Certains arcs du réseau peuvent être agrégés entre eux voire supprimés, comme expliqué dans [Connors and Watling2015], afin de réduire les temps de calcul et d'exécution. Cette différence de représentation du réseau pose problème pour la composition de simulateurs dans le cadre d'une simulation multi-échelles. Nous devons être capables de représenter une affectation des voyageurs équivalente dans les deux simulateurs, même si certains arcs des chemins des voyageurs sont différents. Il faut également être capables de traduire l'information de la vitesse moyenne d'un arc provenant d'un simulateur vers un autre simulateur, qui ne représente pas forcément cet arc.

La figure 4.2 est un exemple simplifié de deux représentations d'un même réseau de transport où les nœuds rouges sont présents dans les deux réseaux alors que les nœuds bleus ne se trouvent que dans la représentation détaillée.

Hypothèse 3 (co-existence des nœuds) *On suppose que chaque nœud appartenant à une représentation agrégée du graphe G existe dans la représentation détaillée du même graphe. De plus, pour chaque arc présent dans la représentation agrégée, il existe dans la représentation détaillée au moins un chemin entre l'origine et la destination de cet arc qui ne passe pas par un autre nœud commun aux deux représentations.*

En notant N un nœud, A un arc et C un chemin :

— $\forall N \in \sigma_{agr},\ N \in \sigma_{det}$

— $\forall A \in \sigma_{agr}$, $\exists C \in \sigma_{det}$ *tel que* $Origine(C) = Origine(A)$ *et* $Destination(C) = Destination(A)$.

Cette contrainte est nécessaire aux méthodes que nous présentons dans cette section et s'explique facilement. Si les nœuds correspondent à des arrêts ou des points de transition d'un réseau de transport à un autre, alors la représentation agrégée aura conservé ceux considérés comme les plus importants qu'on devrait donc retrouver dans la représentation détaillée. La deuxième contrainte qui porte sur les chemins signifie simplement que si on peut se rendre d'un point a à un point b sans passer par un point c dans la représentation agrégée, alors on peut faire la même chose dans la représentation détaillée.

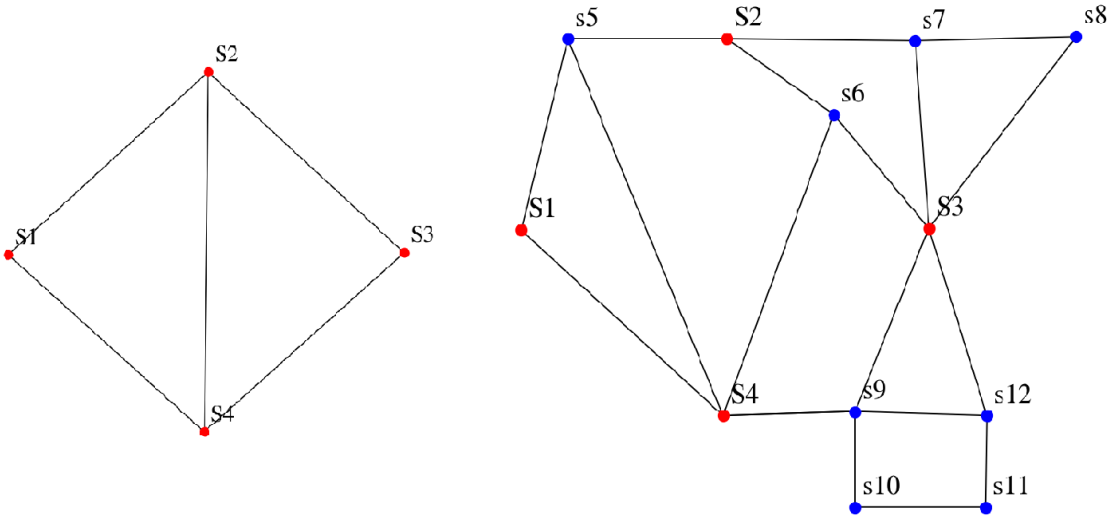

FIGURE 4.2 – Différentes représentation d'une même zone

4.10 Composition de processus : l'opérateur ⊕

Puisqu'il y a trois fonctions à composer, la première étape est de déterminer quel simulateur corrige quelle fonction de l'autre simulation (cf. section 4.6). Dans notre exemple, le simulateur régional pourrait par exemple corriger la demande alors que le simulateur local corrigerait l'affectation et la vitesse. Il faut noter que si un seul des deux simulateurs corrige seul les trois fonctions alors nous n'avons pas besoin de deux simulateurs synchrones, le simulateur correcteur peut être utilisé seul dans un premier temps et ses résultats peuvent alors être utilisés par le second simulateur, qui sera exécuté seul dans un

second temps. On serait alors dans un cas de simulation séquentielle et non pas interactive. Les 8 types de compositions sont énumérés dans le tableau 4.1.

Dans la suite de cette section, nous détaillons les corrections nécessaires aux trois fonctions de la simulation (`Demand`, Assign, Move). Les manières de traiter ces corrections sont autant de fonctionnalités à intégrer dans le modèle d'intergiciel.

Hypothèse 4 (Corriger les simulateurs avec modèle de poursuite) *Étant donné que l'intergiciel ne dispose que de deux leviers pour influencer le comportement des simulations (le graphe G et la matrice M_{od}), nous devons expliciter un cas particulier pour la composition de la fonction `Move`. Il s'agit du cas où l'on voudrait corriger le déplacement d'un simulateur qui s'appuie sur un modèle de poursuite. Dans ce cas, les temps de parcours sur les arcs ne sont pas considéré dans le déplacement des voyageurs. Ainsi, l'intergiciel peut transmettre le temps de parcours moyen par arc à $\mathcal{S}_2$ qui pourra ensuite modifier la vitesse de chaque véhicule proportionnellement pour que les temps de parcours correspondent.*

Dans la suite de cette section, nous détaillons les méthodes utilisées par l'intergiciel, afin de composer les trois fonctions essentielles des deux simulateurs. Le simulateur noté $\mathcal{S}_1$ est toujours celui qui corrige la fonction concernée.

4.10.1 Composition des vitesses - fonctions `Move`

En règle générale, afin de corriger la fonction `Move` dans un simulateur $\mathcal{S}_2$ par un autre simulateur $\mathcal{S}_1$, l'intergiciel modifie les vitesses sur les arcs empruntés par les voyageurs de $\mathcal{S}_2$ par les vitesses moyennes observées sur $\mathcal{S}_1$ (il modifie en fait les valuation des arcs dans G). Cela permet à $\mathcal{S}_2$, quelque soit son échelle, d'appliquer son modèle de déplacement avec les vitesses corrigées.

Dans le cas de ω (composées avec l'opérateur $\circ$) différents entre $\mathcal{S}_1$ et $\mathcal{S}_2$, l'intergiciel enverra à $\mathcal{S}_2$ un graphe G avec les temps de parcours moyens observés.

Dans le cas de θ différents (composées avec l'opérateur $\diamond$), l'intergiciel devra exécuter autant de pas de temps nécessaires de $\mathcal{S}_1$ jusqu'à couvrir un pas de temps complet de $\mathcal{S}_2$ avant d'exécuter $\mathcal{S}_2$ avec les bonnes valuations du graphe G. Cela est nécessaire pour exécuter $\mathcal{S}_2$ avec les bonnes vitesses (rappelons que le simulateur noté $\mathcal{S}_1$ dans notre description est toujours celui qui corrige la fonction considérée).

Lorsqu'on a une différence de représentation spatiale σ (composées avec l'opérateur $\bullet$), la composition des fonctions `Move` est effectué comme suit. Pour effectuer la composition de vitesse, nous souhaitons utiliser la vitesse moyenne d'un arc du réseau du simulateur $\mathcal{S}_1$ pour corriger la fonction `Move` de $\mathcal{S}_2$. Notre problématique est donc de déterminer la vitesse moyenne des arcs d'une représentation spatiale σ_1 en considérant que l'on connaît la vitesse des arcs dans l'autre représentation σ_2. Afin d'obtenir le même temps de parcours des voyageurs dans chaque simulateur, le temps de parcours moyen de chaque arc doit être le même que le temps de parcours moyen de ses chemins correspondants.

La Figure 4.3 présente le passage du réseau détaillé au réseau agrégé. Nous corrigeons le temps de parcours du lien agrégé, comme étant la moyenne des temps de parcours de ses chemins correspondants. Dans l'exemple de la Figure 4.3 nous obtenons donc $T(S1, S2)_{detailed} = 17.5s$. Une fois que cette opération est effectuée, la vitesse moyenne est déterminée en divisant le temps de parcours par la longueur de l'arc.

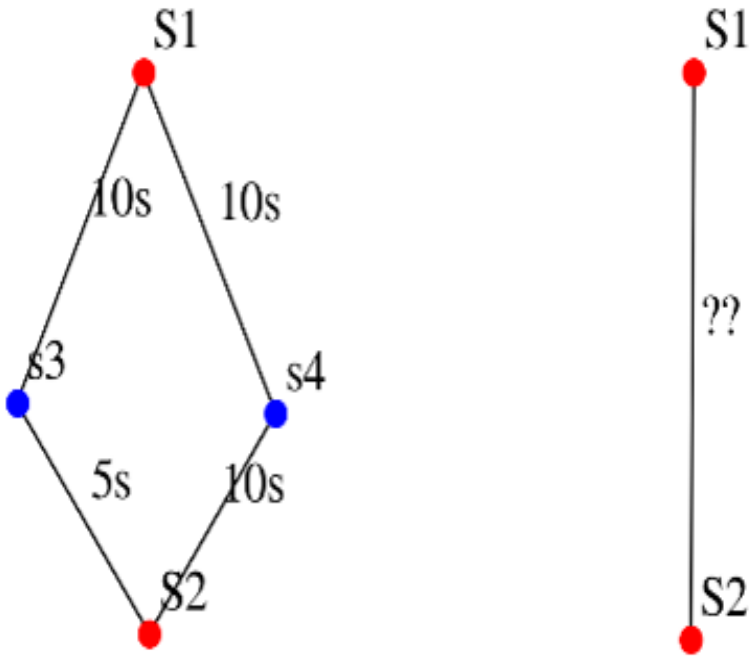

FIGURE 4.3 – Calcul de la vitesse moyenne d'un arc agrégé depuis la représentation détaillée

La Figure 4.4 présente le passage de la représentation agrégée à la représentation détaillée. Une fois encore, notre objectif est d'obtenir un temps de parcours des arcs agrégés égal à la moyenne des temps de parcours des chemins correspondants.

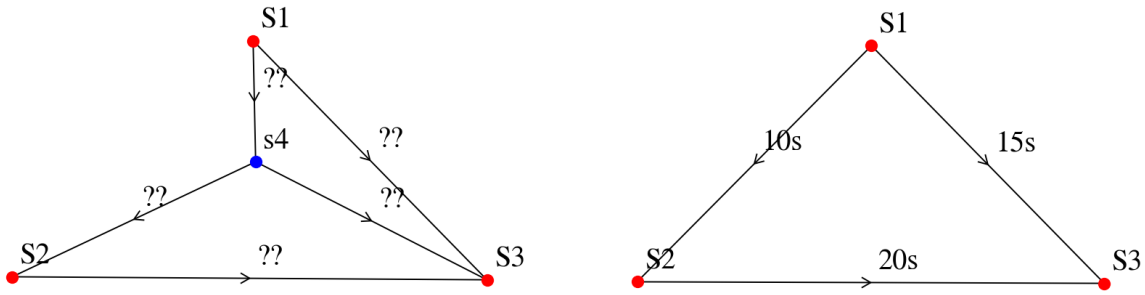

FIGURE 4.4 – Calcul de la vitesse d'un arc de la représentation détaillée depuis la représentation agrégée

On obtient alors le système d'équation linéaires suivant :

— $T(\overrightarrow{S_1S_2})_{agr} = T(\overrightarrow{S_1, s_4})_{det} + T(\overrightarrow{s_4, S_2})_{det}$
— $T(\overrightarrow{S_1, S_3})_{agr} = \frac{(T(\overrightarrow{S_1, s_4})_{det} + T(\overrightarrow{s_4, S_3})_{det}) + T(\overrightarrow{S_1, S_3})_{det}}{2}$
— $T(\overrightarrow{S_2, S_3})_{agr} = T(\overrightarrow{S_2, S_3})_{det}$

Il y a une infinité de solutions comme par exemple :

— $T(S_1, s_4)_{det} = 5s, T(s_4, S_2)_{det} = 5s, T(s_4, S_3)_{det} = 10s$
— $T(S_1, s_4)_{det} = 6s, T(s_4, S_2)_{det} = 4s, T(s_4, S_3)_{det} = 11s$
— ...

Comme à chaque passage de la représentation agrégée vers la représentation détaillée on retrouve un manque d'information que l'on doit créer. Pour cela, nous essayons de garder la proportionnalité entre les temps de parcours calculés par le $\mathcal{S}_1$. Ainsi, on garde les différences entre les multiples chemins, mises en évidence par le modèle de déplacement de $\mathcal{S}_1$, tout en assurant la cohérence des vitesses d'un simulateur à l'autre.

4.10.2 Composition de la demande - fonctions `Demand`

Chaque simulateur dispose de sa propre matrice M_{od}. Quelque soit le scénario et l'échelle considérés, dès que deux simulateurs sont en interaction, ils doivent échanger des voyageurs et des incohérences peuvent survenir. Cette différence entre les matrices origine-destination (OD) prises en compte dans les deux simulateurs (qui sont déterminées avant l'exécution) et la demande dynamique (qui est calculée ou obtenue à chaque pas de temps) doit être prise en compte lors de la composition des fonctions `Demand` de $\mathcal{S}_1$ et $\mathcal{S}_2$. Dans le cas général, l'intergiciel fournira M_{od_1} à $\mathcal{S}_2$ à chaque pas de temps, qu'il considérera en

lieu et place de sa propre matrice.

Dans le cas d'un σ différent avec σ_1 plus détaillé que σ_2, il faut potentiellement trouver de nouveaux nœuds d'origine et de destination pour les voyageurs dans M_{od2}, puisque les nœuds renseignés peuvent être inexistant dans σ_2. Dans ce cas, l'intergiciel crée les voyageurs considérés sur les nœuds appartenant aux deux représentations les plus proches de M_{od1}. Ce cas n'est pas rencontré si σ_2 est plus détaillé que σ_1 puisque tous les nœuds du simulateur détaillé existent dans le simulateur agrégé (cf. hypothèse 3).

Un autre cas particulier est à considérer, lorsque la zone couverte par M_{od1} est incluse dans la zone couverte par M_{od2}. Il faudrait idéalement que $\mathcal{S}_2$ redéfinisse M_{od2} à chaque pas de temps, de telle manière qu'elle soit cohérente avec M_{od1} (supposée plus correcte). Retrouver la matrice M_{od2} s'apparente à un problème de calibration de paramètres, et est laissé en dehors du champ de l'intergiciel. Cependant si cette calibration n'a pas été faite en amont de la simulation,l'intergiciel permet de forcer la cohérence en supprimant des voyageurs arrivant dans la zone couverte par M_{od1} depuis l'extérieur et en en créant de nouveaux en s'appuyant sur M_{od1}.

4.10.3 Composition de l'affectation - fonctions `Assign`

Dans un modèle classique à 4 étapes, l'affectation est exécutée une fois que l'on connaît le nombre de personnes voyageant entre chaque origine et chaque destination, i.e. une fois les matrices OD déterminées, et une fois que l'on connaît les modes de transport qu'ils utilisent. Il s'agit de les affecter à un chemin sur lequel la simulation les fera se déplacer.

Pour une simulation, on utilise en général un modèle d'affectation dynamique du trafic (DTA pour *Dynamic Traffic Assignment*) qui affecte les voyageurs en prenant en compte l'évolution des temps de parcours en fonction de l'état du réseau au cours du temps. L'affectation peut être faite de plusieurs manières, une des plus utilisées s'appelle l'équilibre utilisateur [Szeto and Wong2012] qui signifie qu'une fois affecté, plus aucun voyageur ne peut modifier son parcours afin d'en obtenir un plus avantageux. Un autre type d'affectation est l'optimum du système [Szeto and Wong2012] qui signifie que le temps de parcours total de l'ensemble des voyageurs a été minimisé même si certains voyageurs pourraient changer de chemin pour améliorer leur propre temps de trajet, en impactant négativement d'autres voyageurs. En général, ces équilibres ne sont pas précisément atteints et les simulations essaient de converger vers une solution acceptable à l'aide de multiples itérations de la même simulation. Ces itérations successives prennent du temps d'exécution et

c'est pourquoi des affectations plus simples existent. Par exemple, la méthode des K plus courts chemins consiste à trouver l'ensemble des chemins entre l'origine et la destination d'un voyageur et de l'envoyer sur un des K plus courts chemins sans chercher à atteindre un équilibre.

La Figure 4.5 montre comment la matrice OD est utilisée en tant que paramètre d'entrée pour alimenter la boucle entre le DTA et le simulateur qui exécutent des itérations pour trouver une solution acceptable. Une fois que le DTA reçoit la matrice OD, il propose une solution au simulateur qui l'utilise lors de l'exécution. Lorsque la simulation est terminée, il envoie ses résultats de temps de parcours au DTA qui propose alors une nouvelle solution. Cette boucle fermée entre le DTA et le simulation en interaction est appelée l'affectation basée sur la simulation. Dans ce cas, la simulation est exécutée à plusieurs reprises, c'est ainsi que l'équilibre utilisateur et l'optimum du système sont atteints.

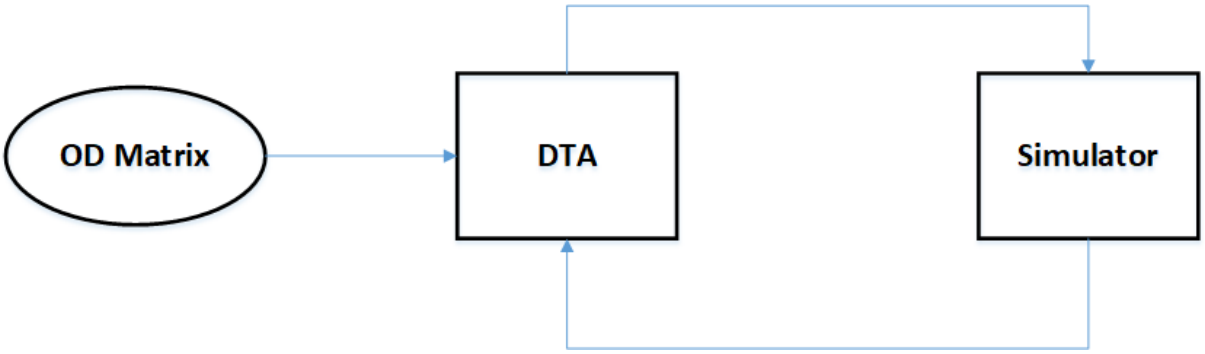

FIGURE 4.5 – Interaction entre un simulateur et un DTA qui cherchent à converger vers une solution d'équilibre

Dans le cas d'une affectation simple comme pour un K plus courts chemins, l'affectation est faite avant la simulation et ce dernier ne renvoie pas de résultats au DTA - Figure 4.6. Il s'agit donc d'une unique itération avec les matrices OD en paramètre du DTA et les voyageurs affectés à leurs chemins en paramètre du simulateur qui n'est exécuté qu'une fois.

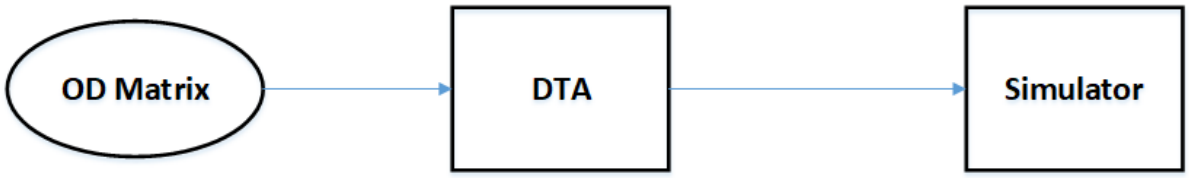

FIGURE 4.6 – Affectation d'un coup, sans itération

La correction de l'affectation est atteinte par l'intergiciel à travers la valuation des arcs dans G, qui font qu'un calcul de plus court chemin par $\mathcal{S}_{\in}$ suive le résultat de l'affectation

effectuée par $\mathcal{S}_1$ (rappelons que le simulateur noté $\mathcal{S}_1$ dans notre description est toujours celui qui corrige la fonction considérée).

Dans le cas de ω différents (composées avec l'opérateur $\circ$), l'intergiciel n'a rien de particulier à faire, puisque l'affectation concerne la valuation des arcs de G seulement.

Dans le cas de θ différents (composées avec l'opérateur $\diamond$), l'intergiciel devra exécuter autant de pas de temps nécessaires de $\mathcal{S}_1$ jusqu'à couvrir un pas de temps complet de $\mathcal{S}_2$ avant d'exécuter $\mathcal{S}_2$ avec les bonnes valuations du graphe G. Cela est nécessaire pour exécuter $\mathcal{S}_2$ avec les bonnes vitesses.

Dans le cas d'un σ différent (composées avec l'opérateur $\bullet$), certains chemins existant dans une des deux représentation n'existent pas dans l'autre. La première étape est de trouver les chemins du réseau détaillé qui correspondent à chaque arc du réseau agrégé. Nous définissons ces chemins de la manière suivante : un chemin dans la représentation détaillée correspond à un arc de la représentation agrégée si l'origine et la destination du chemin sont les mêmes que respectivement l'origine et la destination de l'arc et que ce chemin ne passe par aucun autre nœud présent dans la représentation agrégée. La figure 4.7 montre les chemins correspondants pour l'arc $(\overrightarrow{S_2S_4})$ de la représentation agrégée. L'hypothèse 3 nous assure qu'au moins un chemin correspond à chaque arc.

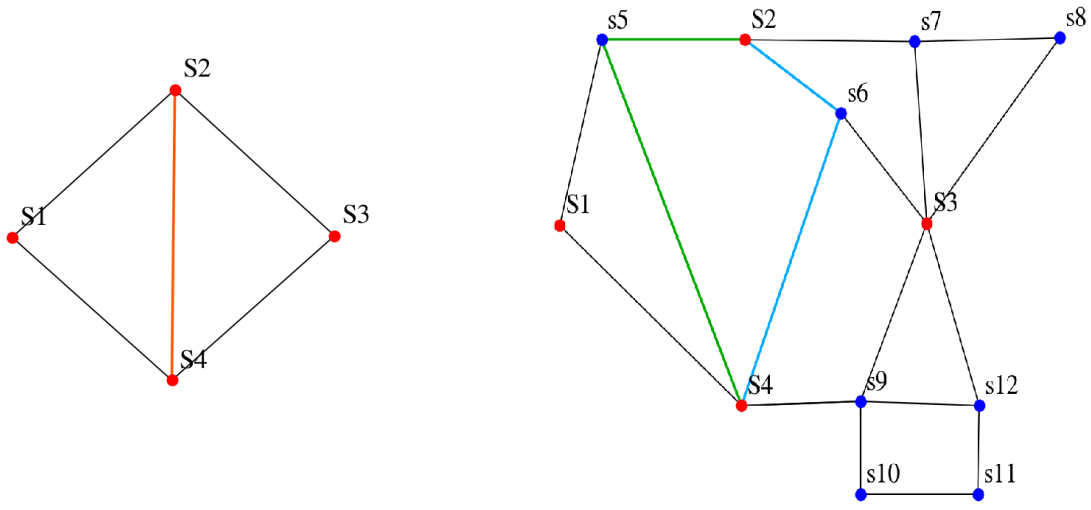

FIGURE 4.7 – Chemins correspondant à un arc de la représentation agrégée

Une fois les chemins correspondants trouvés, afin de corriger l'affectation :

— Pour obtenir le nouveau chemin dans la représentation agrégée depuis la représentation détaillée, nous devons supprimer les nœuds qui n'existent pas dans la représentation agrégée pour que le chemin soit valide dans celle-ci. Par exemple, le chemin $(\overrightarrow{S_1s_5}, \overrightarrow{s_5S_2}, \overrightarrow{S_2s_7}, \overrightarrow{s_7S_8}, \overrightarrow{s_8S_3})$ devient $(\overrightarrow{S_1S_2}, \overrightarrow{S_2S_3})$.

— Pour obtenir le nouveau chemin dans la représentation détaillée depuis la représentation agrégée, nous devons ajouter de nouveaux arcs appartenant à la représentation détaillée à un chemin existant. Même si le simulateur agrégé corrige l'affectation, seul le simulateur détaillé peut déterminer les arcs de la représentation détaillée à utiliser. Ainsi le simulateur détaillé définit une nouvelle affectation contrainte par l'affectation du simulateur agrégé précédemment faite. Le simulateur détaillé doit garder le chemin planifié par le simulateur avec la représentation agrégée et effectuer une nouvelle affectation partielle pour déterminer le chemin correspondant à chaque arc de la représentation agrégée. Le chemin $(\overrightarrow{S_1S_2}, \overrightarrow{S_2S_3})$ peut donc devenir $(\overrightarrow{S_1s_5}, \overrightarrow{s_5S_2}, \overrightarrow{S_2s_7}, \overrightarrow{s_7S_8}, \overrightarrow{s_8S_3})$ ou $(\overrightarrow{S_1s_5}, \overrightarrow{s_5S_2}, \overrightarrow{S_2s_7}, \overrightarrow{s_7S_3})$ ou encore $(\overrightarrow{S_1s_5}, \overrightarrow{s_5S_2}, \overrightarrow{S_2s_6}, \overrightarrow{s_6S_3})$. Les chemins correspondants aux arcs définis dans l'étape précédente sont utilisés pour cette affectation contrainte.

4.11 Correction croisée et ordonnancement de simulateurs

Nous avons décrit les opérations qui doivent être effectuées par un intergiciel pour composer des simulations de mobilité multi-échelles différentes. Cette section décrit un cas complexe, dans lequel les méthodes décrites pourraient s'avérer insuffisantes pour assurer une simulation multi-échelles cohérentes, ainsi que les méthodes nécessaires pour le résoudre. Nous commençons par instancier le scénario avec un cas réel, tel que nous l'avons rencontré au sein du projet MSM.

4.11.1 Instantiation du modèle : composition d'un simulateur local et d'un simulateur régional

Cette thèse s'inscrit dans le cadre du projet MSM (Modélisation de Solutions de Mobilité) de l'institut de recherche SystemX. Le projet MSM s'intéresse particulièrement aux problématiques de la meilleure connaissance et gestion de la mobilité des personnes à

l'échelle d'un « quartier gare »ou encore à l'échelle d'un bassin de vie. Dans ce contexte, nous imaginons dans cette thèse le poste de « gestionnaire de quartier », responsable de la gestion de la mobilité au niveau d'un quartier important donné (e.g. La Défense en région parisienne). Les modes de transport disponibles dans le quartier sont multimodaux, chacun géré par un opérateur différent. Le gestionnaire est intéressé par tous les déplacements à l'intérieur du quartier, et désire les représenter le plus finement possible, à l'échelle individuelle. Néanmoins, le quartier n'étant pas isolé du monde, les déplacements dans le quartier sont fortement dépendants des flux de voyageurs et des services de transport en amont et en aval du quartier. Si un évènement imprévu se produit à un endroit du réseau régional (e.g. Châtelet - les Halles), il risque d'y avoir un grand impact sur les flux traversant ou ayant pour destination le quartier en question. Il est donc nécessaire d'avoir une connaissance de la région dans laquelle le quartier se situe (e.g. l'Île de France pour la Défense). Il n'est cependant pas nécessaire ni utile de représenter les flux régionaux d'une manière aussi fine que le quartier.

Pour réaliser une simulation fidèle de ces deux « échelles », nous pouvons créer une nouvelle simulation, de type multiniveaux, qui représente conjointement les deux échelles souhaitées, et permettrait de passer de l'une à l'autre selon les besoins très précis du gestionnaire de quartier. Cependant, cette simulation multiniveaux ne pourra pas servir dans un autre contexte où on désirerait représenter conjointement la région Île-de-France avec l'ensemble des réseaux autoroutiers environnant, par exemple, et une nouvelle simulation multiniveaux s'avérerait nécessaire. C'est la raison pour laquelle nous optons pour une solution qui couple des simulateurs existants, dans le cadre d'une simulation multi-échelles.

Nous travaillons donc ici sur une zone d'intérêt faisant partie d'une zone plus large appelée la région. La région - qui couvre en autres la zone d'intérêt - est simulée entièrement par un simulateur appelé le simulateur régional. La zone d'intérêt est simulée par un second simulateur permettant d'observer plus en détails la dynamique du traffic en son sein comme illustré sur la Figure 4.8, ce simulateur est appelé le simulateur local.

Cette double simulation permet donc de modéliser précisément une zone d'étude tout en permettant de prendre en compte l'ensemble du réseau de transport avec lequel elle interagit et les éventuelles perturbations de ce réseau qui pourraient impacter cette zone d'étude.

Dans notre exemple, le simulateur S_1 (le simulateur régional) est décrit ainsi :

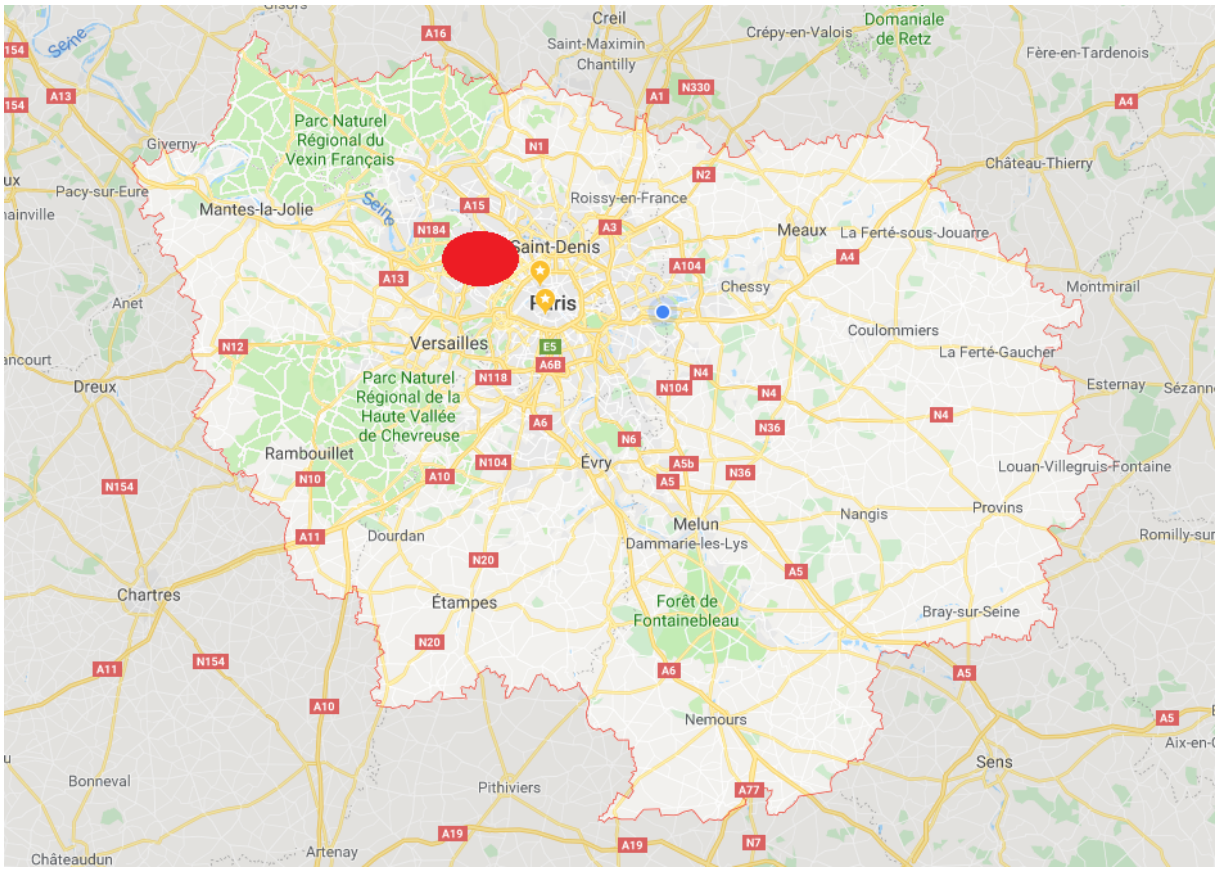

FIGURE 4.8 – Simulation multi-échelles de la région

$$\mathcal{S}_\infty = \langle \omega_1, \sigma_1, \theta_1, \mathcal{P}_\infty \rangle$$

avec :

— ω_1 est une représentation en flux des entités
— σ_1 est une représentation spatiale agrégée
— σ_1 est de l'ordre de 30 minutes
— $\mathcal{P}_1$ est fondé sur :
  — `Demand` qui se fonde sur une M_{od} régionale, à base d'enquêtes
  — `Assign` qui affecte les voyageurs selon l'équilibre utilisateur
  — `Move` qui fait évoluer les voyageurs selon un diagramme fondamental du trafic associé à G_1

$$\mathcal{S}_\in = \langle \omega_2, \sigma_2, \theta_2, \mathcal{P}_\in \rangle$$

avec :

— ω_2 est une représentation individuelle des entités
— σ_2 est une représentation spatiale détaillée
— σ_2 est de l'ordre de 15 secondes
— $\mathcal{P}_2$ est fondé sur :
  — `Demand` qui se fonde sur une M_{od} locale, à base d'enquêtes
  — `Assign` qui affecte les voyageurs selon les K plus chemins
  — `Move` qui fait évoluer les voyageurs selon un diagramme fondamental du trafic associé à G_2

De plus, la zone spatiale couverte par $\mathcal{S}_\in$ est complètement couverte par $\mathcal{S}_\in$.

Afin de pouvoir corriger la demande, l'affectation ou la vitesse de déplacement sur les différents arcs, l'intergiciel doit être capable de faire le lien entre les deux représentations différentes.

	Simulateur régional	Simulateur local	Doit être traité
1	Demand, Assign, Move	-	Non
2	-	Demand, Assign, Move	Non
3	Demand	Assign, Move	Oui
4	Assign	Demand, Move	Oui
5	Move	Demand, Assign	Non
6	Demand, Assign	Move	Oui
7	Demand, Move	Assign	Non
8	Assign, Move	Demand	Oui

TABLE 4.1 – Les différents types d'interactions pour notre exemple

Dans le tableau 4.1, la première colonne correspond aux corrections possibles par le simulateur régional sur le simulateur local, la seconde colonne correspond aux corrections possibles par le simulateur local sur le simulateur régional et finalement la troisième colonne indique si ce type d'interaction est traité ou non dans ce chapitre. Les cas 1 et 2 ne sont pas traités, puisqu'un seul simulateur corrige les trois aspects, pas besoin donc d'interaction entre simulateurs ni d'intergiciel. Les cas 5 et 7 ne sont pas traités non plus car nous considérons que si le simulateur local corrige l'affectation dans la zone d'intérêt alors il doit également corriger la vitesse.

Dans la suite de cet exemple, suivant le niveau de validité de chaque simulateur, nous considérons un complexe et réaliste (cas 6 dans le tableau) :

— $\mathcal{S}_\infty$corrige la demande de $\mathcal{S}_\in$
— $\mathcal{S}_\infty$corrige l'affectation de $\mathcal{S}_\in$
— $\mathcal{S}_\in$ corrige le déplacement de $\mathcal{S}_\infty$

4.11.2 Ordonnancement des simulateurs

Ce cas d'étude illustre des corrections croisées particulières entre simulateurs, qui nécessitent un ordonnancement particulier de leur exécution. Nous avons vu précédemment que le simulateur correcteur du mouvement devait être exécuté en premier afin de pouvoir corriger l'autre simulateur. La première solution proposée avec cette contrainte est donc la suivante, illustrée par les figures 4.9 et 4.10 :

1. Le simulateur régional corrige la demande et l'affectation.
2. Le simulateur local exécute autant de pas de temps que nécessaire pour avoir simulé un temps correspondant à celui d'un pas de temps du simulateur régional (3 fois dans la Figure 4.9).
3. Les vitesses moyennes des différents arcs sont stockées par l'intergiciel à chaque pas de temps du simulateur local et la moyenne de ces vitesses sur plusieurs pas de temps est calculée puis utilisée pour corriger la vitesse dans le simulateur régional.
4. le simulateur régional est exécuté.

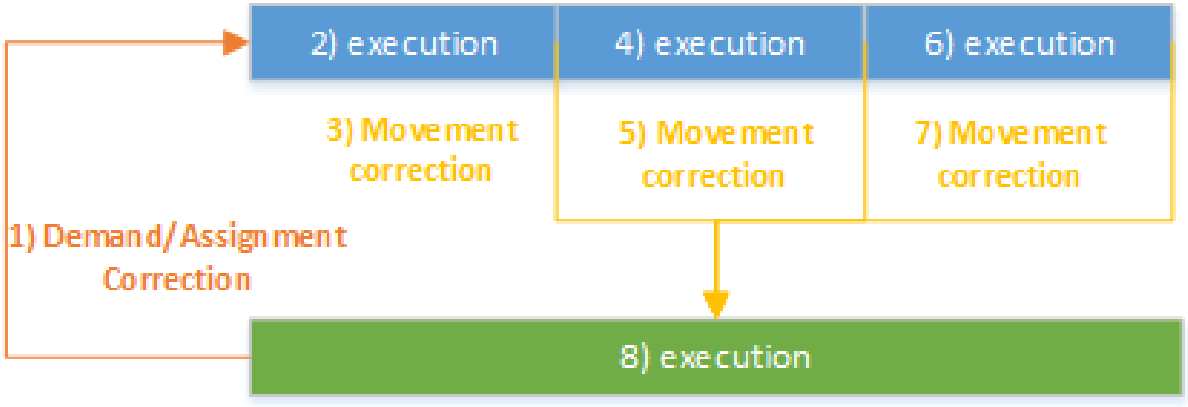

FIGURE 4.9 – Solution corrigeant les vitesses

Avec cette solution, nous avons la correction du mouvement sur tout le pas de temps du simulateur régional mais la demande n'est pas correctement corrigée. Pendant l'exécution du pas de temps du simulateur régional - vert -, certains voyageurs vont entrer dans la

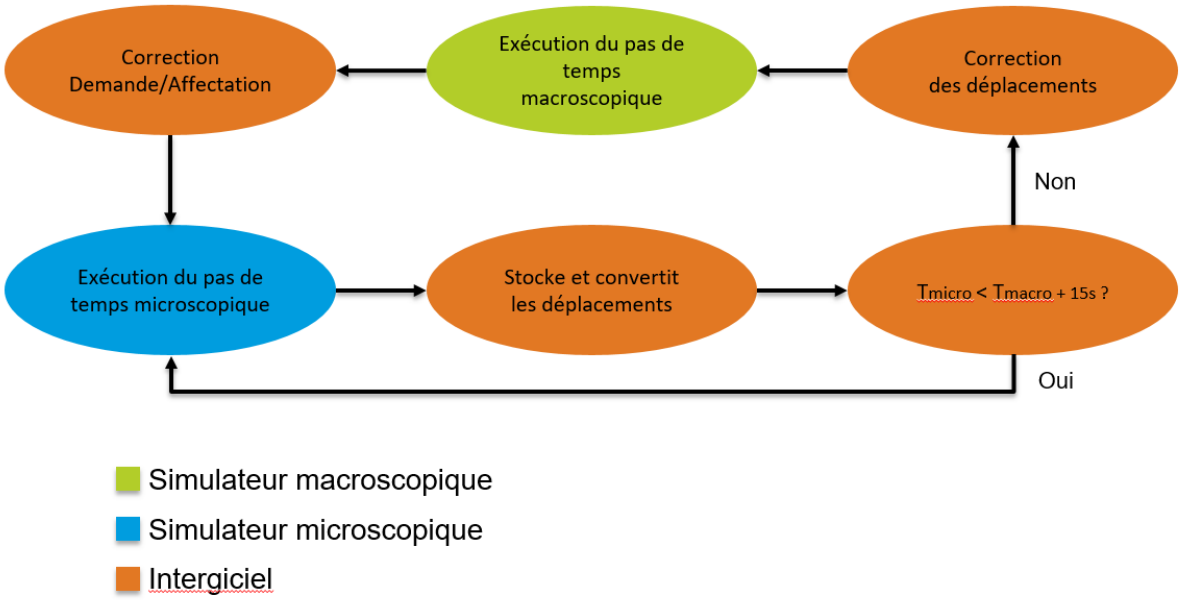

FIGURE 4.10 – Solution corrigeant les vitesses

zone d'intérêt et ces voyageurs vont devenir de nouveaux voyageurs pour le simulateur local. Ces voyageurs qui entrent dans la zone font donc partie de la demande que le simulateur régional va envoyer au simulateur local et devraient être pris en compte pendant le second et le troisième pas de temps du simulateur local. C'est ce que nous avons appelé dans 4.5 la demande dynamique. Puisqu'ils entrent dans la zone pendant le pas de temps du simulateur régional, ils doivent donc être divisés en trois groupes pour les trois corrections de pas de temps du simulateur local. Ceci nous amène à la seconde solution présentée dans les figure 4.11 et 4.12 :

1. Le simulateur régional corrige la demande et l'affectation.
2. Le simulateur local exécute son premier pas de temps.
3. Le simulateur local corrige le mouvement en fonction de la vitesse calculée dans le premier pas de temps.
4. le simulateur régional est exécuté.
5. Le simulateur régional corrige la demande et l'affectation pour le second et le troisième pas de temps du simulateur local.
6. Le second et le troisième pas de temps du simulateur local sont exécutés.

Contrairement à la solution précédent, dans celle ci le simulateur régional corrige la demande et l'affectation pour chaque pas de temps du simulateur local. En revanche cette fois la correction de la vitesse pour le simulateur régional n'est faite que grâce au

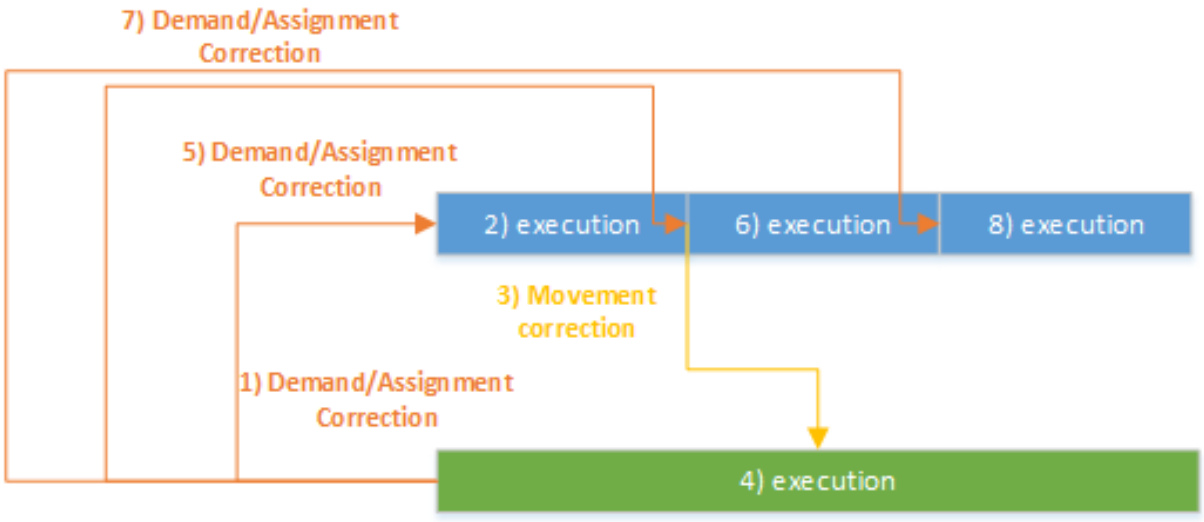

FIGURE 4.11 – Solution corrigeant la demande

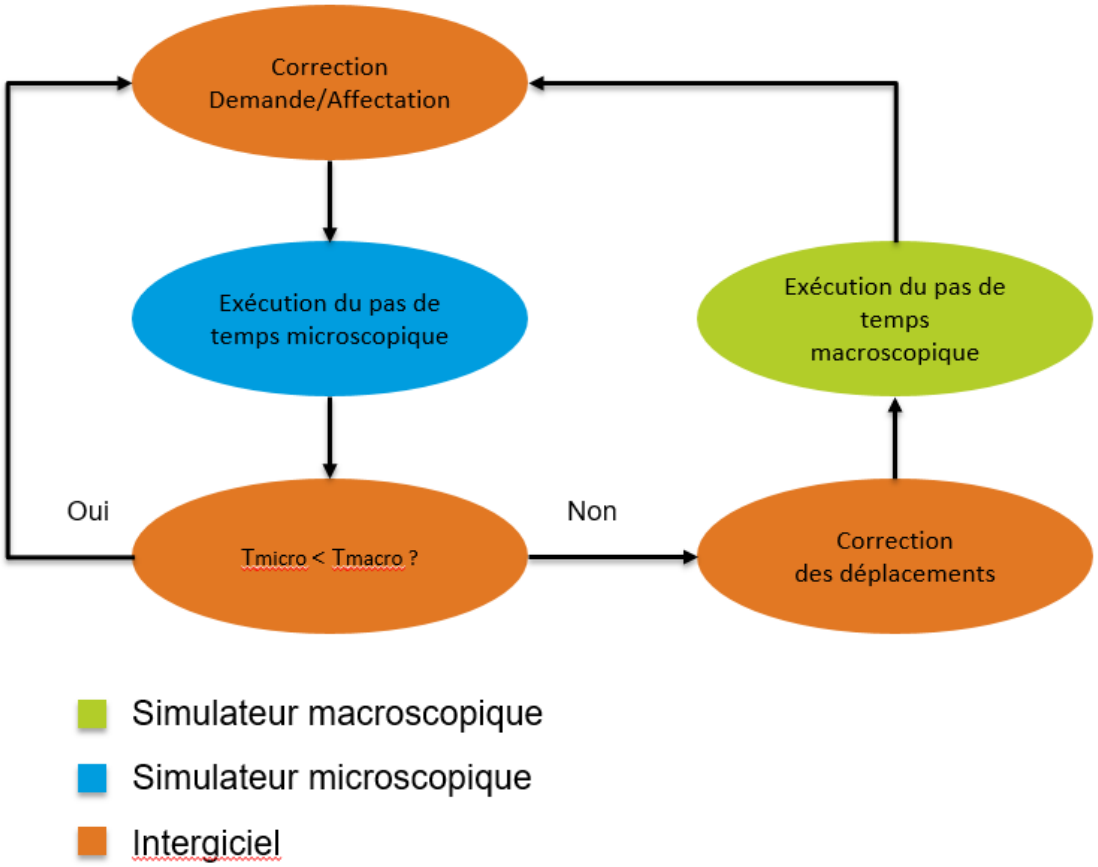

FIGURE 4.12 – Solution corrigeant la demande

premier pas de temps du simulateur local. Avec cette solution la vitesse n'est donc pas correctement corrigé et si la vitesse est modifiée lors du second ou du troisième pas de temps local, cela n'influencera pas le simulateur régional. La seule solution pour corriger correctement les trois critères est d'exécuter plusieurs fois le pas de temps du simulateur régional en faisant des retours en arrière à la fin du pas de temps comme le montre les figures 4.13 et 4.14 :

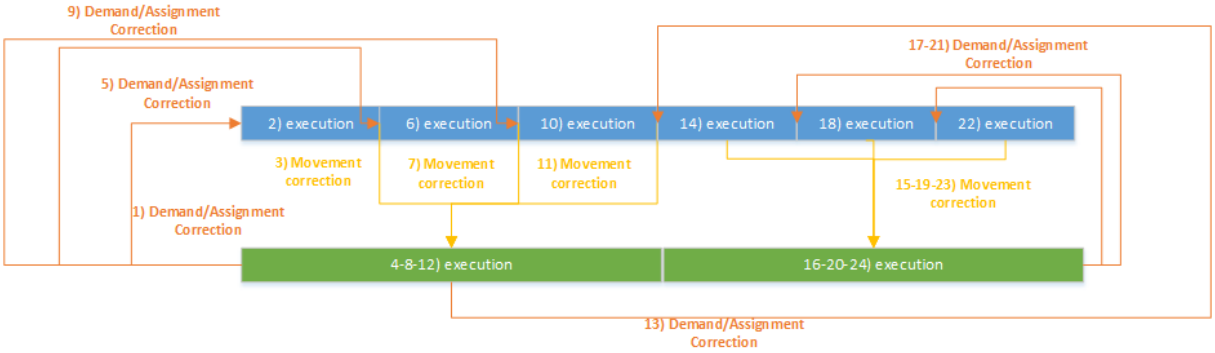

FIGURE 4.13 – Synchronisation avec retour en arrière

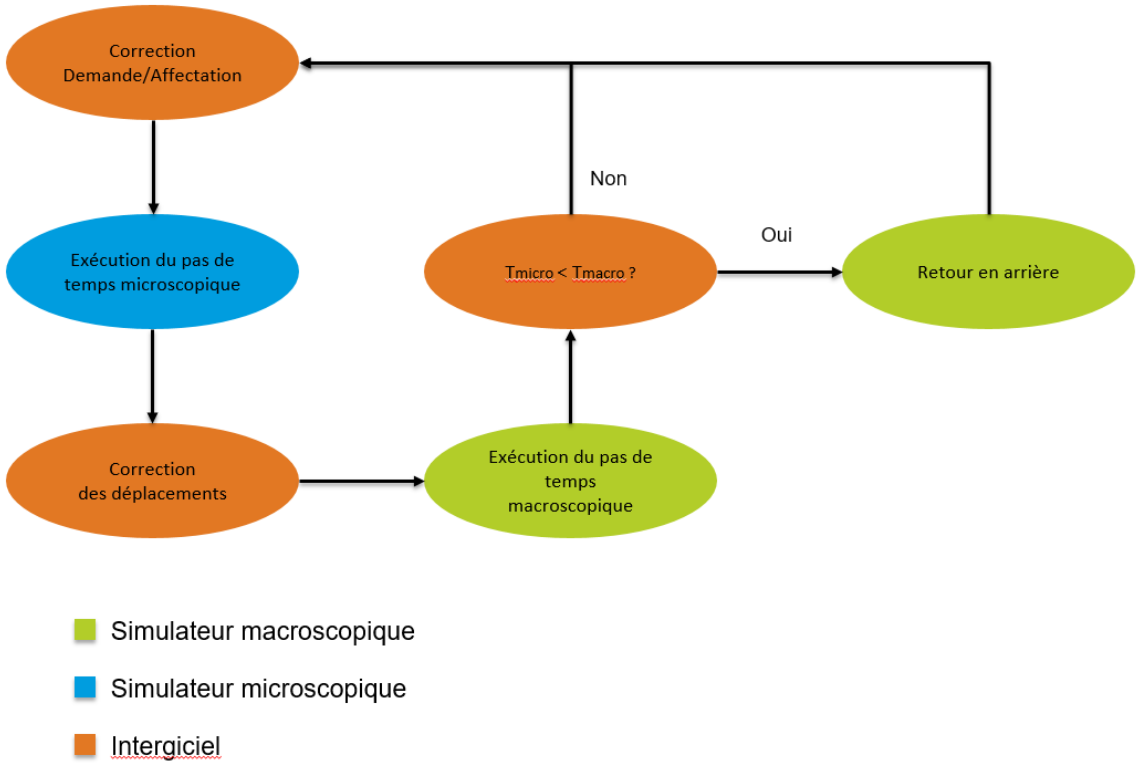

FIGURE 4.14 – Synchronisation avec retour en arrière

Dans cette dernière solution :

1. La demande et l'affectation sont corrigés par le simulateur régional pour le premier pas de temps du simulateur local.
2. Le simulateur local est exécuté.
3. La simulateur local corrige le mouvement du simulateur régional.
4. Le simulateur régional est exécuté une première fois afin de pouvoir corriger la demande pour le second pas de temps du simulateur local.
5. La demande et l'affectation du second pas de temps du simulateur local sont corrigées.
6. Le simulateur local est exécuté.
7. Le simulateur local corrige le mouvement du simulateur vert avec la vitesse moyenne calculée sur les premiers et second pas de temps bleus.
8. Le simulateur régional est exécuté une seconde fois, sur le même pas de temps, avec la nouvelle correction de mouvement, afin de fournir une nouvelle demande pour le troisième pas de temps du simulateur local..
9. Et cela avec autant de retours en arrière qu'il y a de pas de temps du simulateur local dans un pas de temps de simulateur régional.

Cette solution avec retour en arrière permet la correction de la demande, de l'affectation et de la vitesse. Évidemment c'est une solution coûteuse et le temps d'exécution de la simulation est bien plus important qu'avec les deux autres comme nous le verrons dans la partie 5.3.2. Des solutions intermédiaires peuvent être utilisées ou le retour en arrière ne se fait pas à chaque pas de temps du simulateur local. Par exemple si un pas de temps du simulateur régional correspond à 20 pas de temps du simulateur local alors le retour en arrière peut être fait tous les 5 pas de temps du simulateur local.

Finalement, un dernier cas est à prendre en compte. Nous avons considéré que le plus long pas de temps (30 minutes) est un multiple du plus petit (15 secondes). Dans [Biedermann *et al.*2014] l'auteur propose de résoudre le cas où aucun pas de temps n'est un multiple de l'autre dans le but de fournir un framework pour les simulations multi-échelles de piétons le plus générique possible. Ainsi la solution proposée pour corriger la vitesse et la position des piétons à un instant t_1 pour le simulateur 1 lorsque il n'y a des résultats sur le simulateur 2 qu'aux instants t_{2-} et t_{2+} avec $t_{2-} < t_1$ et $t_{2+} > t_1$ est d'effectuer une interpolation de ces résultats pour obtenir une estimation à l'instant t_1. Dans notre cas cela modifie trois points par rapport à la dernière solution proposée - Figure 4.15 :

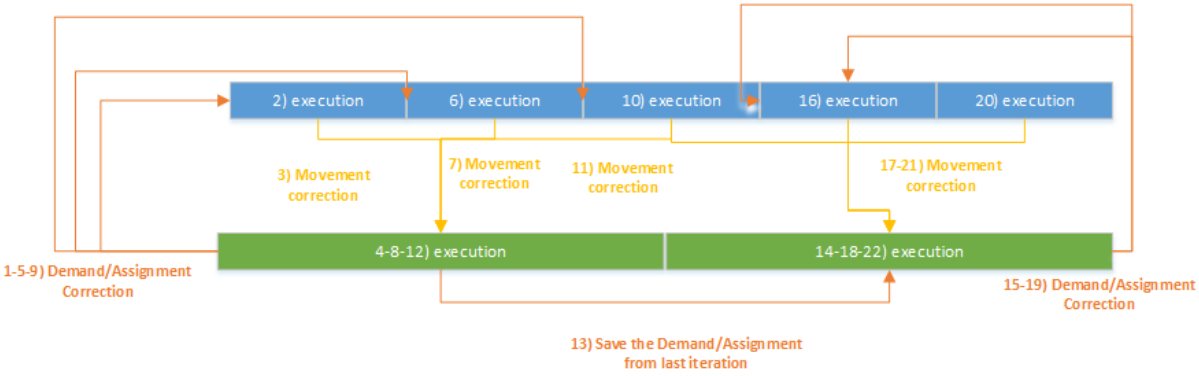

FIGURE 4.15 – Synchronisation avec *rollback*

— Ã la fin de la dernière itération du pas de temps du simulateur régional, on continue avec la première itération du second pas de temps du simulateur régional. Dans la solution précédente on continuait avec un nouveau pas de temps du simulateur local.
— Lors de la dernière itération du pas de temps du simulateur régional, la demande qui arrive dans la zone d'intérêt doit être sauvegardée et prise en compte lors de la prochaine correction de la demande.
— Le pas de temps du simulateur local situé entre les deux pas de temps du simulateur régional doit corriger le mouvement pour la fin du premier pas de temps et le début du second.

L'interpolation n'est pas nécessaire dans notre cas. Au niveau de la demande on sauvegarde simplement la demande qui est arrivée à la fin du premier pas de temps du simulateur régional et elle sera ajoutée à la demande calculée par la première itération du second pas de temps. Au niveau des vitesses nos corrections se font toujours sur les laps de temps correspondant, durant toute la fin du premier pas de temps du simulateur régional on a bien la vitesse calculée pendant le pas de temps du simulateur local.

Nous avons pris l'exemple d'une interaction avec un simulateur régional qui corrige la demande et l'affectation et un simulateur local corrigeant les vitesses. Nous avons vu dans le tableau 4.1 que d'autres interactions sont possibles, à savoir les cas (3), (4), (6) et (8) du tableau des types d'interaction présenté en début de chapitre. Il se trouve que le cas traité ici, le (6) est le cas le plus complexe et que les algorithmes proposés s'adaptent facilement aux autres différents cas comme nous allons le voir ci-dessous :

— Tout d'abord dans le cas (3) où le simulateur régional corrige la demande et où le simulateur local corrige l'affectation et les vitesses, les algorithmes présentés

précédemment n'ont presque pas besoin d'être modifiés. En effet, l'affectation au sein de la zone d'intérêt se fait simplement par le simulateur local après la correction de la demande venant du simulateur régional. Ce dernier est toujours exécuté après le simulateur local pour que ses vitesses soient corrigés et donc la correction de l'affectation, qui se fait à présent au même moment que celle des vitesses, est également faite avant l'exécution du simulateur régional. Le simulateur local, quant à lui, récupère la demande de la même manière que dans le cas (6) et l'affecte lui-même. La seule modification de l'algorithme est donc que dans ce cas, l'affectation est corrigée par le simulateur local et cela se fait au même moment que la correction des vitesses.

— Le cas (8) où le simulateur local ne corrige que la demande et où le simulateur régional corrige les autres est le plus simple. En effet, la correction de la demande par le simulateur local ne consiste qu'à modifier les matrices Origines-Destinations du simulateur régional en ajoutant les voyageurs prévus par le local et à ne pas prendre en compte la demande dynamique 4.10.2 qui apparaît lors de l'exécution. Cette modification des matrices OD se fait avant l'exécution de la simulation. Ensuite, pendant l'exécution, tout peut donc être corrigé par le simulateur régional et on tombe sur un cas trivial comme (1) ou (2). On peut donc tout à fait exécuter le simulateur régional seul puis le simulateur local seul avec les résultats du régional.

— Finalement dans le cas (4) où le simulateur régional ne corrige que l'affectation alors que le simulateur local corrige la demande, comme dans le cas précédent, la demande est corrigée avant l'exécution. Ici, puisque la demande est corrigée en amont de la simulation et qu'il n'y a pas de demande dynamique, on n'a pas besoin de retours en arrière car on se retrouve dans le cas présenté par la figure 4.9. Pendant l'exécution, on corrige l'affectation puis on exécute plusieurs pas de temps du simulateur local et enfin on exécute le pas de temps du simulateur régional.

Nous avons vu jusqu'ici les données que s'échangent les simulateurs et nous souhaitons à présent donner un aperçu global de la manière dont les voyageurs seront créés, affectés, déplacés et détruits par les deux simulateurs. Nous concluons ici la présentation des interactions entre les deux simulateurs en résumant le fonctionnement du système de simulation multi-échelles proposé. Tout d'abord, comme le montre la figure 4.16 nous pouvons différencier les voyageurs en fonction du chemin qu'ils parcourent.

Évidemment, les voyageurs qui ne passent jamais pas la zone d'intérêt - type 5 sur la figure - sont entièrement gérés par le simulateur régional. Les voyageurs dont l'origine est en dehors de la zone d'intérêt mais qui se rendent dans celle-ci - type 3 - ou qui passent par

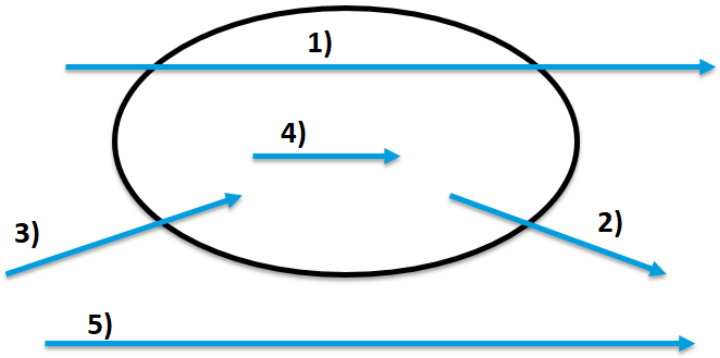

FIGURE 4.16 – Différents types de voyageurs

celle-ci - type 1 - sont dans un premier temps créés, affectés et déplacés par le simulateur régional jusqu'à arriver dans la zone d'intérêt. Si la demande est corrigée par le simulateur local, alors ces voyageurs sont supprimés lorsqu'ils arrivent dans la zone et de nouveaux voyageurs seront créés selon la matrice OD locale. En revanche, si la demande est corrigée par le simulateur régional, alors ils restent en activité et sont également créés dans le simulateur local.

Les voyageurs dont l'origine se situe à l'intérieur de la zone d'intérêt - types 2 et 4 - sont créés par les deux simulateurs en même temps à l'aide de la matrice OD du simulateur régional, qui aura au besoin intégré la matrice OD locale (cf. correction de la demande plus haut).

Dans tous les cas, l'affectation en dehors de la zone d'intérêt se fait par le simulateur régional. L'affectation à l'intérieur de la zone se fait par le simulateur correcteur de l'affectation, s'il s'agit du simulateur local alors les voyageurs qui viennent de l'extérieur - types 1 et 3 - sont simplement ré-affectés une fois dans la zone en conservant leur origine et leur destination.

Enfin, chaque voyageur est déplacé à la fois dans les deux simulateurs séparément selon leur propre modèle et la cohérence est assurée avec la correction de vitesse moyenne des arcs.

4.12 Conclusion

Fondés sur notre état de l'art, nous avons conclu de la pertinence de définir un modèle d'intergiciel pour le couplage de simulateurs, dans le cadre d'une simulation multi-échelles. Dans ce chapitre, nous nous sommes attelés à cette tâche. Nous avons commencé par présenter une méthodologie pour classifier les simulateurs selon leurs caractéristiques et déterminer le type de couplage entre eux. Nous avons ensuite discuté de la question de la validation des simulateurs de mobilité, notion nécessaire pour la correction des simulateurs les uns les autres. Puis, nous avons présenté le modèle d'intergiciel pour le couplage de simulateurs, en nous focalisant successivement sur les aspects qui doivent être traités. Le modèle sert à l'implémentation d'un outil informatique avec lequel deux simulateurs peuvent être connectés pour obtenir une simulation multi-échelles, développé dans le chapitre suivant.

Les principaux choix effectués dans ce chapitre sont :

1. L'application de fonctions d'agrégation d'information et des fonctions de probabilité afin de créer de l'information pour passer d'une représentation des voyageurs à une autre
2. La conversion de chaque section du réseau agrégé en un ensemble de chemins correspondants dans le réseau détaillé puis de faire correspondre l'affectation et le temps de parcours, également à l'aide de fonctions d'agrégation et de probabilité
3. Lorsque c'est nécessaire, l'utilisation d'un retour en arrière pendant la synchronisation des simulateurs permettant de rejouer des pas de temps

Chapitre 5

Expérimentations

Sommaire

5.1 Introduction

Le chapitre précédent nous a permis de détailler les problématiques scientifiques lors de la mise en place d'une simulation multi-échelles, ainsi qu'un modèle abstrait d'intergiciel pour ce faire. Dans ce chapitre, nous concrétisons ces contributions de trois manières. D'abord, nous détaillons les spécificités techniques d'un intergiciel implémentant le modèle, et nous discutons de l'architecture distribuée permettant son déploiement. Ensuite, nous exécutons un ensemble d'expérimentations pour valider nos choix de conception. Nous mettons notamment en évidence l'effet principal de l'intergiciel sur les résultats des simulateurs. Enfin, nous vérifions que l'intergiciel est facilement réutilisable avant de conclure le chapitre.

5.2 Spécification d'un intergiciel multi-échelles

Pour optimiser l'interopérabilité de l'intergiciel, et en nous fondons sur les conclusions du chapitre 3, nous fondons l'écosystème simulateurs/intergiciels sur l'architecture SOA (*Service Oriented Architecture*).

5.2.1 Spécification architecturale

Une architecture orientée service (SOA) est composée d'un ensemble de services indépendants qui peuvent être appelés par les consommateurs de service : les clients. Ces consommateurs de service sont des applications qui nécessitent des fonctionnalités et qui envoient une requête au fournisseur du service requis qui, à son tour, renvoie une réponse. La requête et la réponse dans les services Web de type WS-* obéissent à un langage standard fondé sur XML : SOAP. Les opérations, les types de paramètres et toutes les informations de déploiement des services sont également définis dans un langage standard, également fondé sur XML :WSDL (*Web Service Description Language*). Les architectures orientées services permettent un couplage faible des différents composants et donc une approche modulaire de programmation ainsi que la réutilisation d'applications existantes. Cela diffère de la plupart des architectures orientées objet, particulièrement pour les simulations distribuées, avec des architectures comme HLA (*High Level Architecture*) [Paul *et al.*2005].

L'intergiciel est donc un service Web de type WS-*, qui assure la transmission des messages entre les simulateurs (concernant la demande et le graphe de transport). Mais comme on l'a vu à la fin du chapitre précédent, il peut également agir comme un ordonnanceur des différentes simulations. Dans ce cas, l'écosystème intergiciel/simulateurs devient une orchestration de services Web (cf. Figure 5.1). Pour simplifier la présentation, le cas du couplage de deux simulateurs régional (appelé macro par abus de langage) et local (appelé micro) sera utilisé comme illustration principale dans ce chapitre.

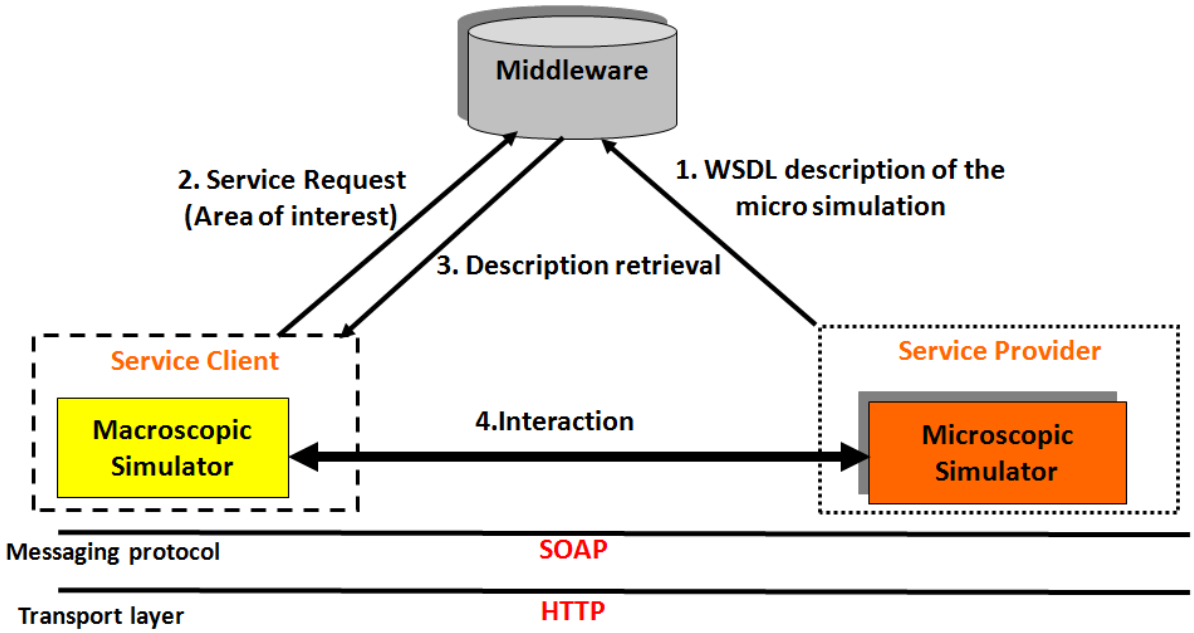

FIGURE 5.1 – SOA pour une simulation multi-échelles (cas micro/macro)

5.2.2 Orchestration de simulations par l'intergiciel

Nous avons défini un schéma XSD spécifiant la structure des fichiers XML, définissant l'échelle de chaque simulation participant à la nouvelle simulation multi-échelles. Chaque fichier de description contient les informations suivantes :

— Le type de représentation des voyageurs : sous forme de flux ou d'agents, le nombre de voyageurs représentés par chaque agent.
— Le type de pas de temps. Pas de temps basé sur les évènements ou fixe. Si le pas de temps est fixe, il faut préciser s'il est ou non réglable, ainsi que sa valeur par défaut.
— La représentation du réseau de transport (1D ou 2D)

— La théorie de flux de trafic utilisée (basée sur les files d'attentes, modèle de poursuite, diagramme fondamental, etc.)

Nous avons également défini un schéma pour le fichier de configuration de l'intergiciel, qui contient les informations suivantes :

— Le nombre de simulateurs à coupler avec le simulateur client.
— La description des simulateurs recherchés, une description par simulateur reprenant les caractéristiques du fichier de description.
— Un fichier XML représentant le réseau de transport.
— Un fichier indiquant les différentes zones d'intérêt et leurs limites.
— Le type de couplage recherché (l'aspect corrigé par chaque simulateur - demande, affectation ou déplacement)

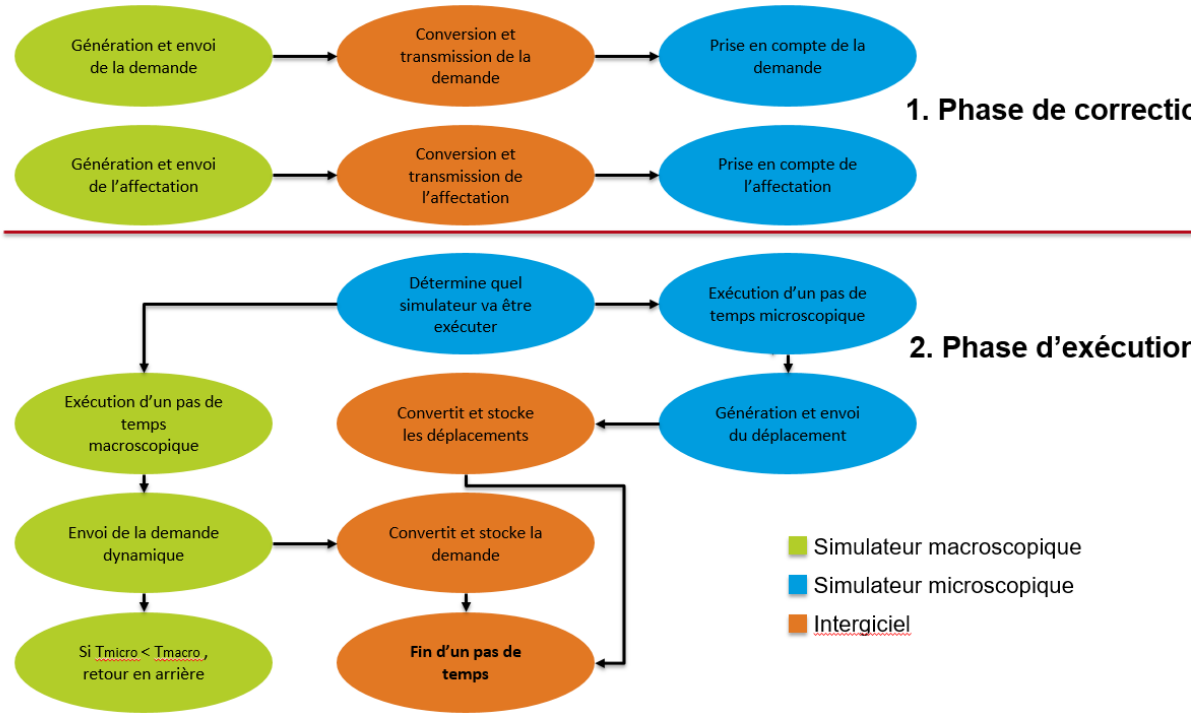

FIGURE 5.2 – Workflow de l'intergiciel

Pour être correctement synchronisés, les simulateurs doivent passer par deux phases : une phase de correction et une phase d'exécution l'une après l'autre à chaque pas de temps. La figure 5.2 présente l'ordre d'exécution des différentes tâches entre les simulateurs.

Dans un premier temps, les simulateurs démarrent la phase de correction lors de laquelle toutes les données s'échangent. Chaque simulateur envoie sa demande, son affectation et les vitesses moyennes des arcs à l'intergiciel, celui-ci détermine ce qui doit être

corrigé par chaque simulateur et renvoie au simulateur qui doit effectuer une correction les données qu'il doit prendre en compte. Le simulateur qui va s'exécuter corrige donc les données nécessaires à l'exécution du pas de temps et la cohérence est ainsi assurée.

Lorsque cette phase de correction est terminée, la phase d'exécution commence. Lors de cette phase, un simulateur est exécuté et le second retourne au début de la phase de correction, attendant la fin de l'exécution du premier simulateur. L'intergiciel détermine le simulateur qui va s'exécuter, celui-ci exécute son pas de temps en calculant les nouvelles vitesses sur chaque arc du réseau et en avançant les voyageurs. À la fin de son pas de temps, le simulateur envoie à l'intergiciel le résultat des vitesses et si nécessaire la demande dynamique, c'est à dire les voyageurs qui sont entrés dans la zone d'intérêt durant l'exécution du pas de temps. L'intergiciel stocke ces données qui serviront lors de la correction du second simulateur.

Des modifications doivent être apportées aux simulateurs afin de pouvoir s'exécuter dans ce contexte. Nous veillons à ce qu'elles soient minimales. Nous définissons une interface pour l'intergiciel ainsi qu'une interface que doivent implémenter les simulateurs pour interagir avec l'intergiciel (cf. Figure 5.4).

5.2.3 Structures de données

Nous spécifions une bibliothèque définissant les données échangées, vers laquelle doivent être converties toutes les données échangées. Les données concernant la représentation du réseau de transport doivent être converties dans les classes `Node`, `Link` et `Network` qui sont utilisées par l'intergiciel. Ces classes sont volontairement simples et ne contiennent que des attributs que l'on peut retrouver dans tout simulateur. De la même manière, les classes `OriginDestination` (*Constrained*) et `Demand` (*Constrained*) correspondent aux données pour la demande (respectivement pour l'affectation). Finalement, `Simulator` correspond à une interface que doivent implémenter les simulateurs et `Middleware` à une interface que doit implémenter l'intergiciel.

— Dans la classe `Node`, les attributs `isMacroNode` et `isMicroNode` servent à déterminer les nœuds qui appartiennent aux deux représentations spatiales du réseau.
— Dans la classe `Link`, l'attribut `freeflowspeed` correspond à la vitesse maximale sur l'arc et `realSpeed` à la vitesse moyenne au cours du dernier pas de temps.
— La classe `OriginDestination` correspond à une case d'une matrice origine-destination temporalisée. Combien de voyageurs partent d'une origine, vont à une destination

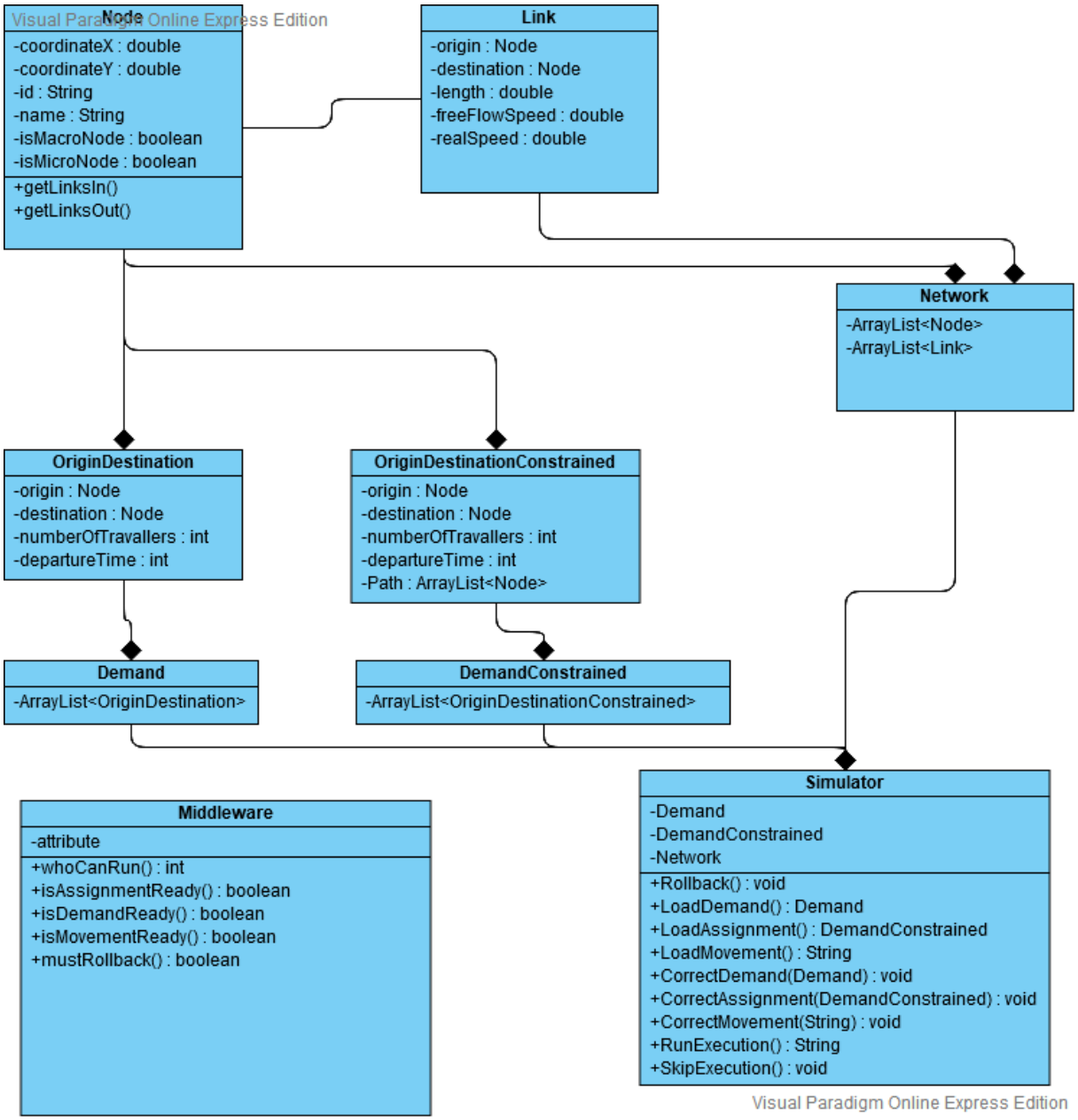

FIGURE 5.3 – Classes utilisées pour l'intergiciel

et quand ils partent.

— La classe `OriginDestinationConstrained` (ODC) correspond à une `OriginDestination` (OD) à laquelle on rajouter une suite de nœuds par lesquels passent les voyageurs. On note que les nœuds peuvent être adjacents dans le réseau agrégé mais pas dans le réseau détaillé, dans ce cas le chemin entre chaque nœud devra être retravaillé dans le simulateur détaillé, c'est l'affectation contrainte. Pour une OD on peut avoir plusieurs ODC correspondantes, une pour chaque chemin emprunté entre l'origine

```java
@WebService
@SOAPBinding(style = Style.RPC)
public interface IntergicielService {
    @WebMethod boolean mustRollback();
    @WebMethod int nextRunner();
    @WebMethod int whoCanRun(int simID);
    @WebMethod boolean canICorrect(int simID);
    @WebMethod void executionFinished(int simID, int timeSpent);
    @WebMethod void correctionFinished(int simID);
    @WebMethod boolean isDemandReady();
    @WebMethod boolean isAssignmentReady();
    @WebMethod boolean isMovementReady();
    @WebMethod void loadDemand(int simID, String simOD, int time, int secPerTS);
    @WebMethod void loadAssignment(int simID, String simODC, int time, int secPerTS);
    @WebMethod void loadMovement(int simID, String simMovement, int time, int secPerTS);
    @WebMethod String correctDemand(int time, int secPerTS);
    @WebMethod String correctAssignment(int time, int secPerTS);
    @WebMethod String correctMovement(int time);
    @WebMethod void simulationFinished(int simID);
}
```

FIGURE 5.4 – Interface utilisée par les différents composants

et la destination.

— `Demand` et `DemandConstrained` correspondent à des listes d'OD/ODC, ce sont donc les matrices OD avec ou sans l'affectation.

— L'intergiciel ordonnance les exécutions avec la méthode `whoCanRun` en renvoyant l'id du simulateur pouvant être exécuté et indique si celui-ci doit effectuer un retour en arrière avec la méthode `mustRollBack`(). Les méthodes `isXready`() permettent lors de la phase de correction de s'assurer que les demandes, affectations et vitesses sont correctement corrigées, c'est à dire égales dans les deux simulateurs.

— Enfin, le simulateur doit être capable d'envoyer sa demande, son affectation et sa vitesse avec les méthodes `loadX`() ; de corriger en important les données de l'intergiciel avec les méthodes `correctX`() ; d'effectuer un retour en arrière et d'exécuter un pas de temps.

5.3 Expérimentations

5.3.1 Configurations

Toutes les expérimentations de ce chapitre ont été faites sur un ordinateur utilisant :

— Un processeur Intel i5-7400.

— 16Go de RAM.

L'intergiciel ainsi que tous les simulateurs exemples utilisés sont codés en Java. Deux simulateurs simplifiés ont été codés pour les expérimentations de validation de l'intergiciel. Deux autres simulateurs existants (MatSIM et SM4T) sont présentés dans la suite du cha-

pitre, ont servi à vérifier la réutilisabilité de l'intergiciel. Pour toutes les expérimentations, les réseaux suivants sont utilisés :

— La Figure 5.5 représente le réseau régional utilisé par le simulateur macroscopique.
— La Figure 5.6 représente le sous-réseau de la zone d'intérêt utilisé par le simulateur macroscopique.
— La Figure 5.7 représente le sous-réseau de la zone d'intérêt utilisé par le simulateur microscopique.

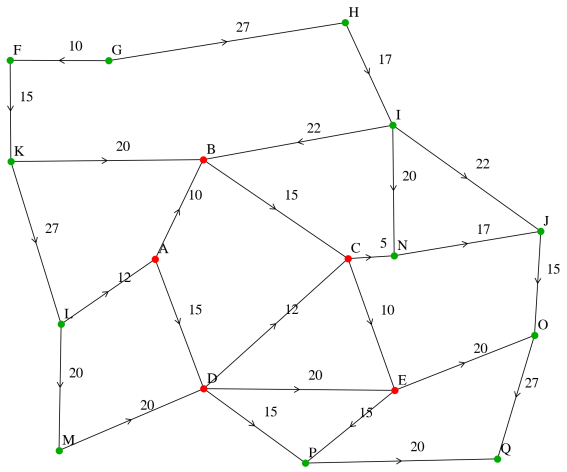

FIGURE 5.5 – Réseau macroscopique de la région

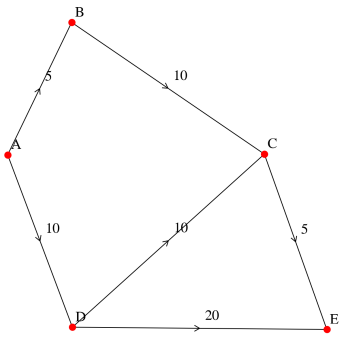

FIGURE 5.6 – Réseau macroscopique de la zone d'intérêt

Pour une durée totale de simulation de 3h20, 12.000 voyageurs sont générés dans la zone d'intérêt pendant la première moitié de la simulation (1h40). Lorsque la demande en provenance du reste de la région - désignée comme demande supplémentaire plus tard dans

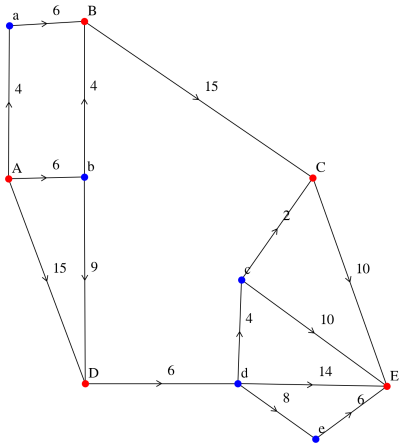

FIGURE 5.7 – Réseau microscopique de la zone d'intérêt

le chapitre - est activée, à ces 12.000 voyageurs s'ajoutent 8.000 voyageurs supplémentaires qui sont générés dans la région dès le début de la simulation.

5.3.2 Expérimentations sur les méthodes de coordination

Ces expérimentations ont été faites afin de valider et d'observer les effets des choix de conception de l'intergiciel suivants :

— Le retour en arrière.
— La répartition des agents générés dans un simulateur multi-agents lorsque ce dernier est corrigé par un simulateur de flux au niveau de la demande. Comme vu dans la section 4.8 du chapitre 4, la répartition des agents ainsi créés est fondée sur [Sewall *et al.*2011].

Pour cela, nous avons implémenté deux simulateurs basiques et nous avons configuré l'intergiciel pour les faire fonctionner ensemble.

— Le premier simulateur fonctionne avec des flux de voyageurs, sur un réseau agrégé et avec un pas de temps de 600 secondes, il correspond à notre simulateur régional. Ce simulateur est appelé dans la suite le simulateur macroscopique. Il affecte les flux de voyageurs sur le plus court chemin de leur origine à leur destination en fonction du temps de parcours des arcs pendant le dernier pas de temps. Ce simulateur macroscopique détermine la vitesse sur les arcs à l'aide d'un diagramme fondamental ([Geroliminis and Daganzo2008]).

— Le second simulateur fonctionne avec des agents représentant les voyageurs individuellement, sur un réseau plus détaillé et avec un pas de temps de 20 secondes. Ce simulateur est appelé le simulateur microscopique. Il affecte les voyageurs en utilisant l'algorithme de K-plus courts chemins de manière à ce que deux voyageurs ayant la même origine et la même destination ne prennent pas forcément le même chemin. Ce simulateur microscopique fonctionne avec un modèle de poursuite (Modèle de Gipp simplifié de [Treiber and Kesting2013]) afin de calculer à chaque pas de temps la vitesse de chaque agent individuellement, en fonction de la vitesse qu'il désire atteindre et de la vitesse de l'agent devant lui sur la section.

Le simulateur macroscopique corrige la demande et l'affectation alors que le simulateur microscopique corrige la vitesse de déplacement dans cette expérimentation. Environ 20.000 voyageurs sont simulés pendant une durée de 3h20.

Les résultats présentés correspondent à la vitesse moyenne sur l'ensemble des arcs du réseau de la zone d'intérêt sous la représentation macroscopique. Pour le simulateur macroscopique on calcule donc la vitesse moyenne de chaque arc de la zone d'intérêt, puis on en fait la moyenne. Pour le simulateur microscopique on retrouve la vitesse moyenne des arcs de la représentation agrégée en effectuant une moyenne pondérée par le nombre de voyageurs des différents arcs de la représentation détaillée composant le chemin correspondant à l'arc de la représentation agrégée.

Pour les expérimentations liées à la synchronisation des simulateurs :

— La figure 5.8 est la solution que nous proposons, elle correspond à la synchronisation avec retour en arrière. Le temps d'exécution de la simulation est de 30.000ms.
— La figure 5.10 correspond à la synchronisation sans retour en arrière où la demande est correctement corrigée mais où la vitesse n'est corrigé qu'à partir d'un pas de temps microscopique. On voit clairement que le pas de temps du simulateur macroscopique est différent de celui de la première figure et qu'on perd en exactitude en ne prenant en compte qu'une partie des résultats microscopiques. Le temps d'exécution de la simulation est de 12.000ms.
— La figure 5.9 correspond à la synchronisation sans retour en arrière où la vitesse est correctement corrigée mais où la demande est prise en compte uniquement lors de certains pas de temps microscopiques. Cela résulte en une surcharge des arcs du réseau car tous les voyageurs arrivent en même temps et se bloquent les uns les autres à cause du modèle de poursuite. Le ralentissement s'amplifie au cours de la simulation car de nouveaux voyageurs arrivent sur des arcs déjà encombrés. Le

temps d'exécution de la simulation est de 11.000ms.

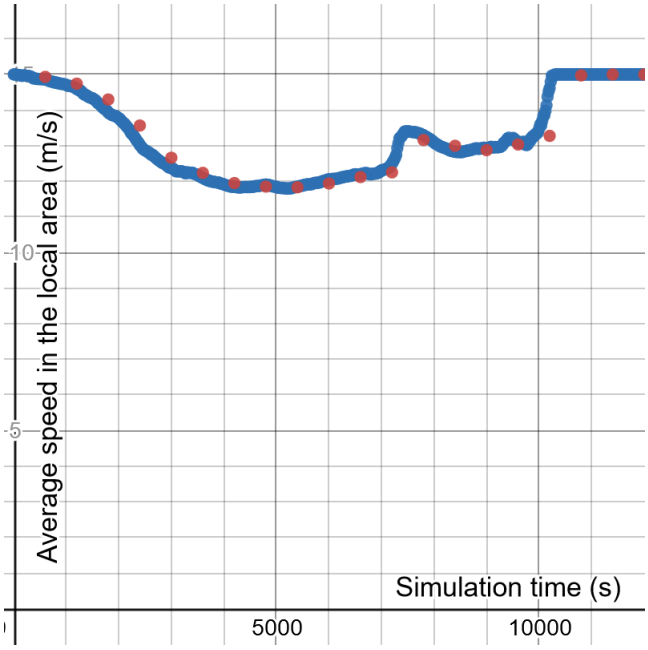

FIGURE 5.8 – Vitesse moyenne des simulateurs avec retour en arrière

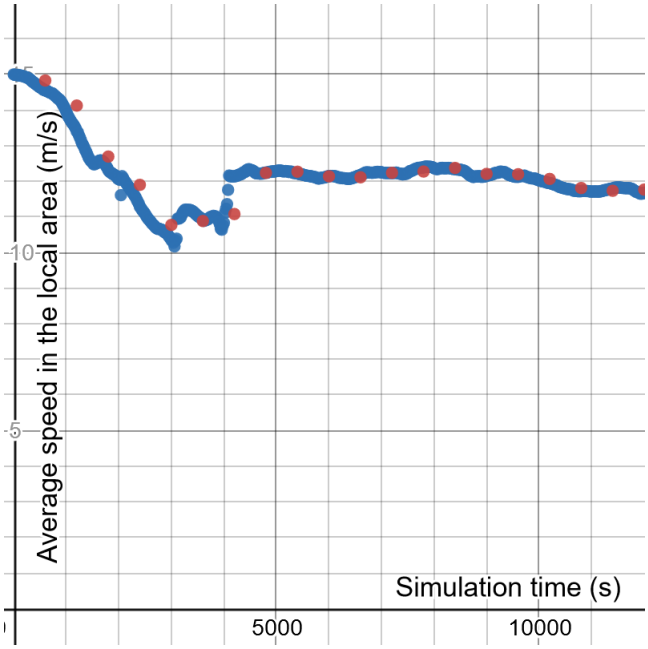

FIGURE 5.9 – Vitesse moyenne des simulateurs avec correction du mouvement

Comme prévu, le mécanisme de retour en arrière assure une simulation plus cohérente et pallie les problèmes rencontrés avec les deux autres solutions. On voit cependant que le temps d'exécution est environ trois fois plus important ce qui rend cette simulation difficilement utilisable dans des cas de simulations importantes.

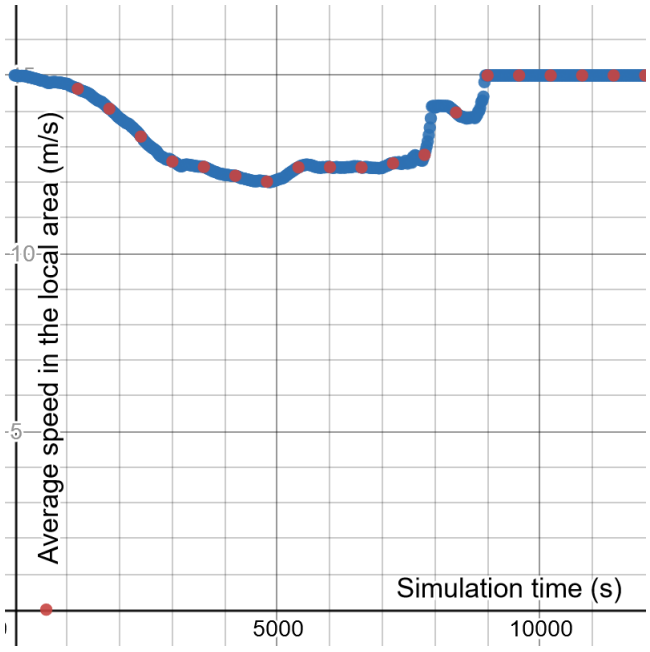

FIGURE 5.10 – Vitesse moyenne des simulateurs avec correction de la demande

Le second choix de conception que nous souhaitons valider est la répartition des voyageurs générés dans le simulateur microscopique. Lorsque les voyageurs arrivent depuis la région - macroscopique - dans la zone d'intérêt - microscopique -, leur passage se fait au niveau des nœuds. Nous proposons de répartir ces voyageurs le long des sections composant leur chemin afin de limiter la congestion liée à une arrivée trop importante provenant du simulateur macroscopique. Cette partie ne concerne donc pas directement l'intergiciel mais le simulateur multi-agents (local), il convient cependant de mettre en avant l'importance de la génération des voyageurs dans ce dernier afin que l'intergiciel soit efficace.

— La figure 5.11 correspond à un simulateur microscopique qui génère les voyageurs répartis sur les premieres sections de leur parcours comme expliqué dans la section 4.8 du chapitre 4.
— La figure 5.12 correspond à un simulateur microscopique qui génère les voyageurs au niveau nœud correspondant à leur origine dans la zone d'intérêt.

On en conclue donc que lorsque le simulateur microscopique utilise un modèle de poursuite, si le simulateur macroscopique corrige la demande, les voyageurs provenant de la région doivent être répartis dans l'espace lorsqu'ils arrivent dans la zone d'intérêt. En effet, si tous les voyageurs sont générés à l'entrée de la section par laquelle ils arrivent, alors cela crée de la congestion. Cela est directement lié à la représentation discrète du temps dans les simulations et l'ampleur du phénomène dépend des pas de temps des deux

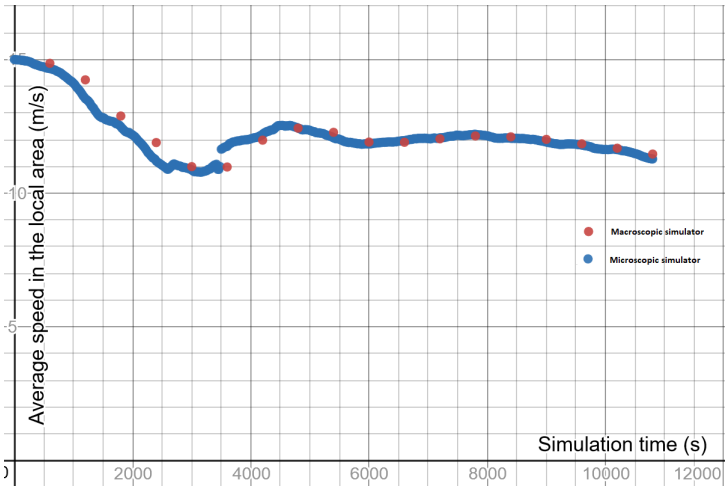

FIGURE 5.11 – Vitesse moyenne des simulateurs avec génération des voyageurs le long de la section d'entrée

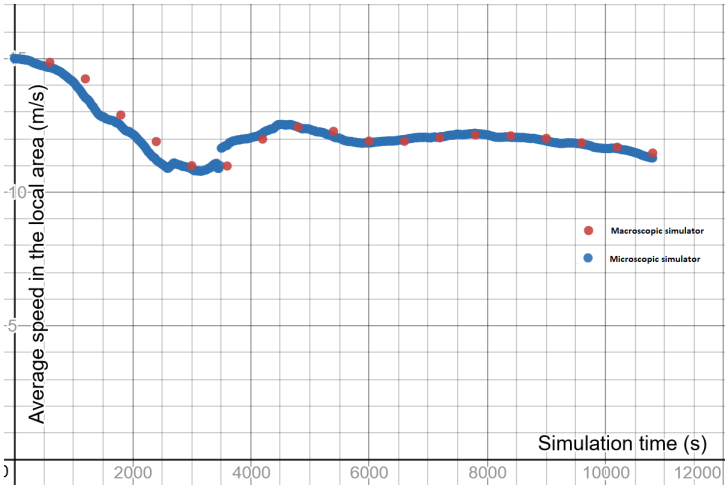

FIGURE 5.12 – Vitesse moyenne des simulateurs avec génération des voyageurs sur le nœud d'entrée

simulateurs. Ici, on se retrouve avec une demande 30 fois trop importante pour le simulateur microscopique. Il s'agit ici d'une modification à faire sur le simulateur directement et non pas dans l'intergiciel.

5.3.3 Expérimentations sur l'effet de l'intergiciel

Ces expérimentations permettent de mettre en avant les effets de l'intergiciel. Les simulateurs utilisés sont les mêmes que dans la section précédente et nous présentons ici 4 scénarios :

— Lors du premier scénario, nous utilisons les deux simulateurs indépendamment pour simuler la même demande sur la même zone, sans utilisation de l'intergiciel et sans aucune correction. Il s'agit d'un ajout constant de passagers à chaque pas de temps pendant tout le début de la simulation - figure 5.13. L'objectif est d'observer la différence de simulation entre les deux simulateurs pour le même cas d'utilisation simple.

— Pour le second scénario, nous ajoutons un pic de demande provenant de la zone extérieure et passant par la zone d'intérêt. Seul le simulateur macroscopique permet de simuler cette zone extérieure et donc seul ce simulateur macroscopique est utilisé dans ce scénario. (cf. figure 5.14).

— Le troisième scénario est le même que le premier sauf que cette fois nous utilisons l'intergiciel pour faire interagir les deux simulateurs. (cf. figure 5.15)

— Finalement, le dernier scénario correspond au cas réel que nous souhaitons simuler. Les deux simulateurs interagissent avec l'intergiciel et la demande supplémentaire venant de la région est prise en compte. (cf. figure 5.16).

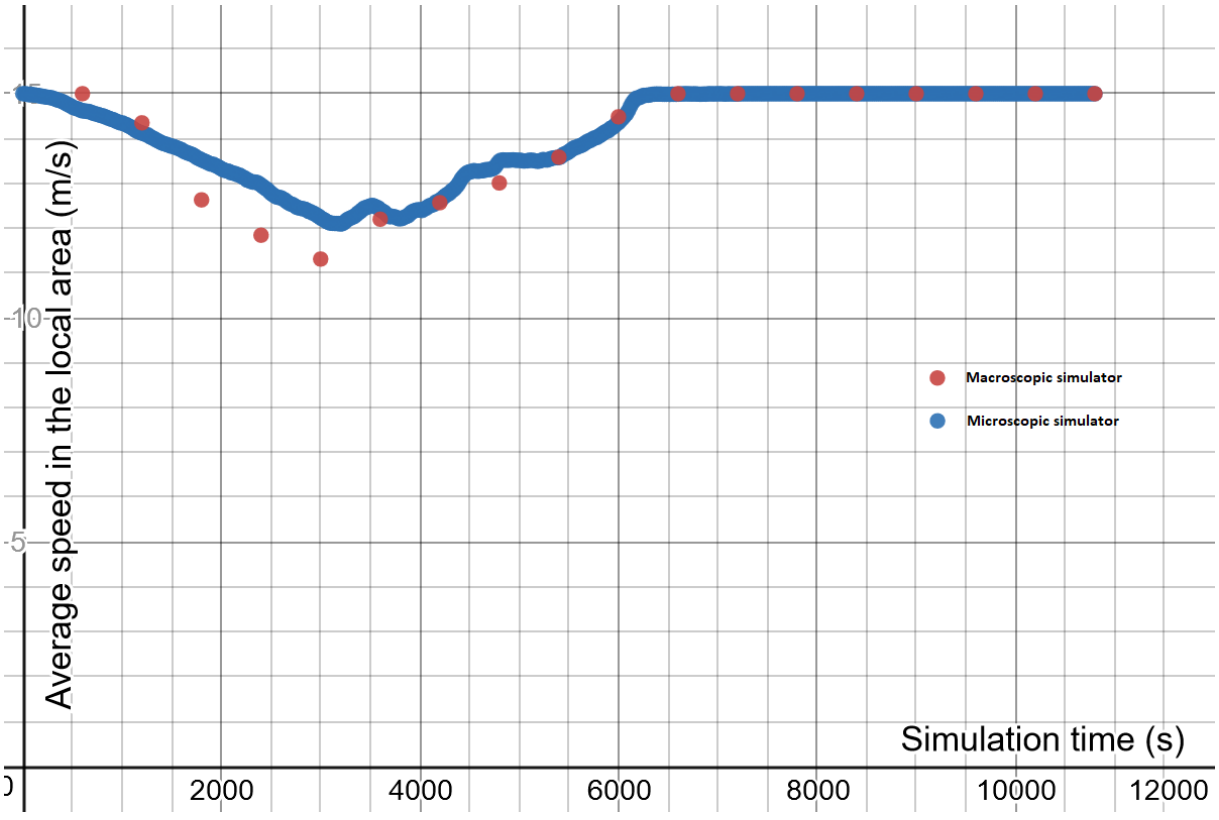

FIGURE 5.13 – Les deux simulateurs utilisent la même demande au sein de la zone d'intérêt

Dans le premier scénario, la congestion est faible et le trafic retrouve rapidement sa vitesse en flux libre, on note cependant des vitesses légèrement différentes entre les

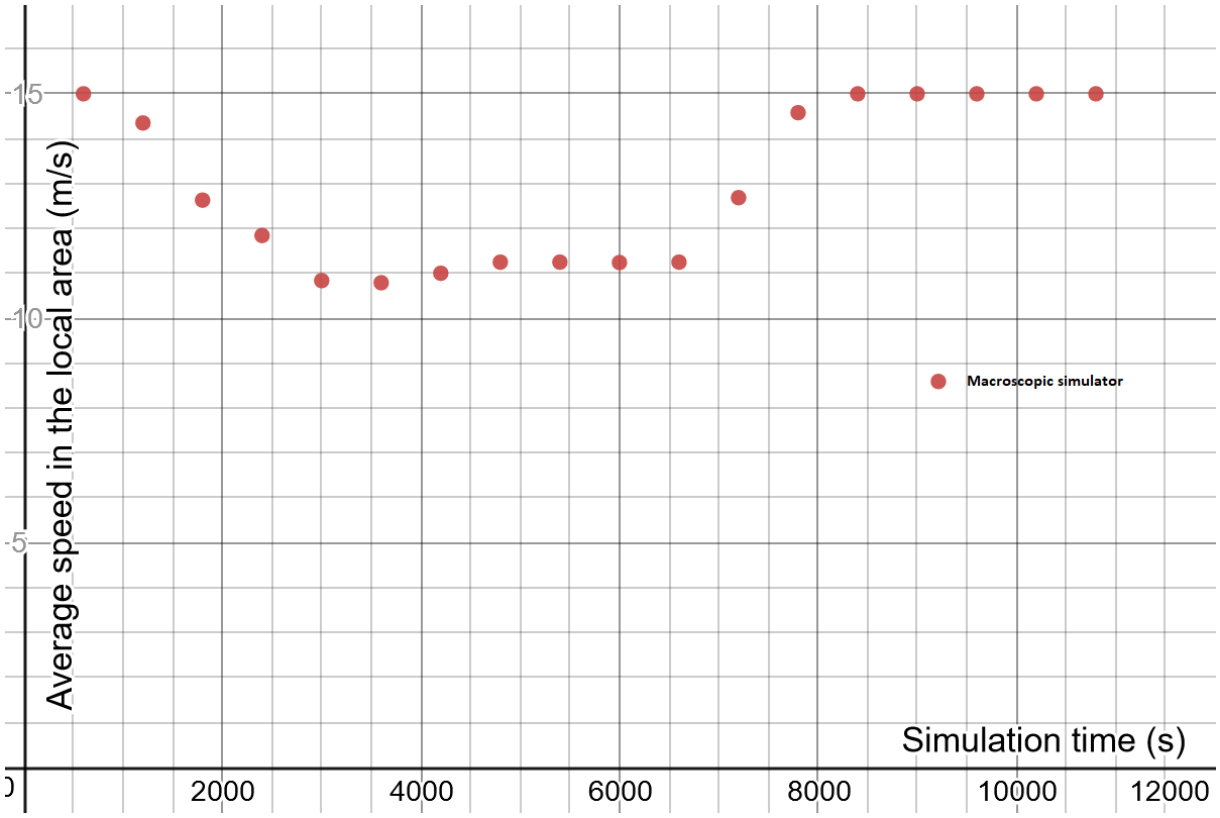

FIGURE 5.14 – Le simulateur macroscopique utilise une demande supplémentaire provenant de la région

deux simulateurs. Le second scénario introduit une demande supplémentaire dont on voit clairement l'effet sur la vitesse moyenne. Lorsque l'intergiciel est utilisé dans le troisième scénario, l'affectation du simulateur macroscopique est utilisée également par le simulateur microscopique ce qui conduit à des sections du réseau saturées et donc à une baisse de vitesse moyenne du trafic que l'on retrouve dans les deux simulateurs. Enfin, dans le dernier scénario, la demande supplémentaire cause une congestion encore plus importante et les simulateurs ne parviennent pas à retrouver un trafic en flux libre.

Ainsi, l'implémentation de l'intergiciel est vérifiée puisque la demande créée dans le simulateur macroscopique ainsi que l'affectation de ce dernier sont bien prises en compte par le simulateur microscopique, et les vitesses des deux simulateurs sont cohérentes. Les trois corrections proposées donc correctement effectuées.

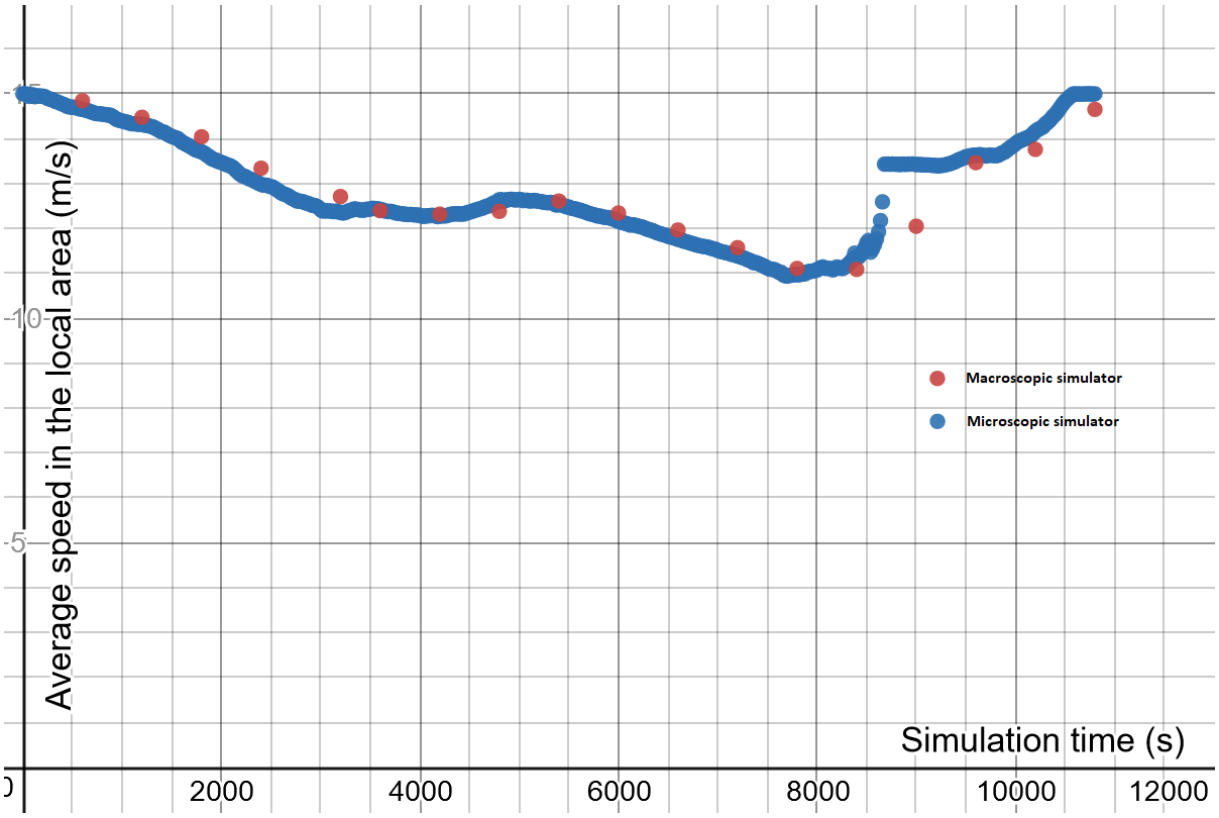

FIGURE 5.15 – Les deux simulateurs interagissent, sans la demande supplémentaire

5.4 Réutilisabilité de l'intergiciel

5.4.1 Utilisation avec les simulateurs précédents

Afin de valider le modèle d'intergiciel et l'interface proposée aux simulateurs souhaitant s'ajouter à l'architecture, nous avons utilisé deux simulateurs microscopiques ainsi que l'intergiciel et le simulateur macroscopique présentés précédemment.

Deux objectifs sous-tendent cette configuration :

1. Le premier objectif est de vérifier que deux simulateurs microscopiques différents peuvent être utilisés de la même manière s'ils implémentent l'interface proposée sans modifier ni l'intergiciel ni le simulateur macroscopique.
2. Le second objectif est d'observer la différence de comportement entre deux simulateurs microscopiques utilisant des modèles différents lors de leur utilisation avec le simulateur macroscopique.

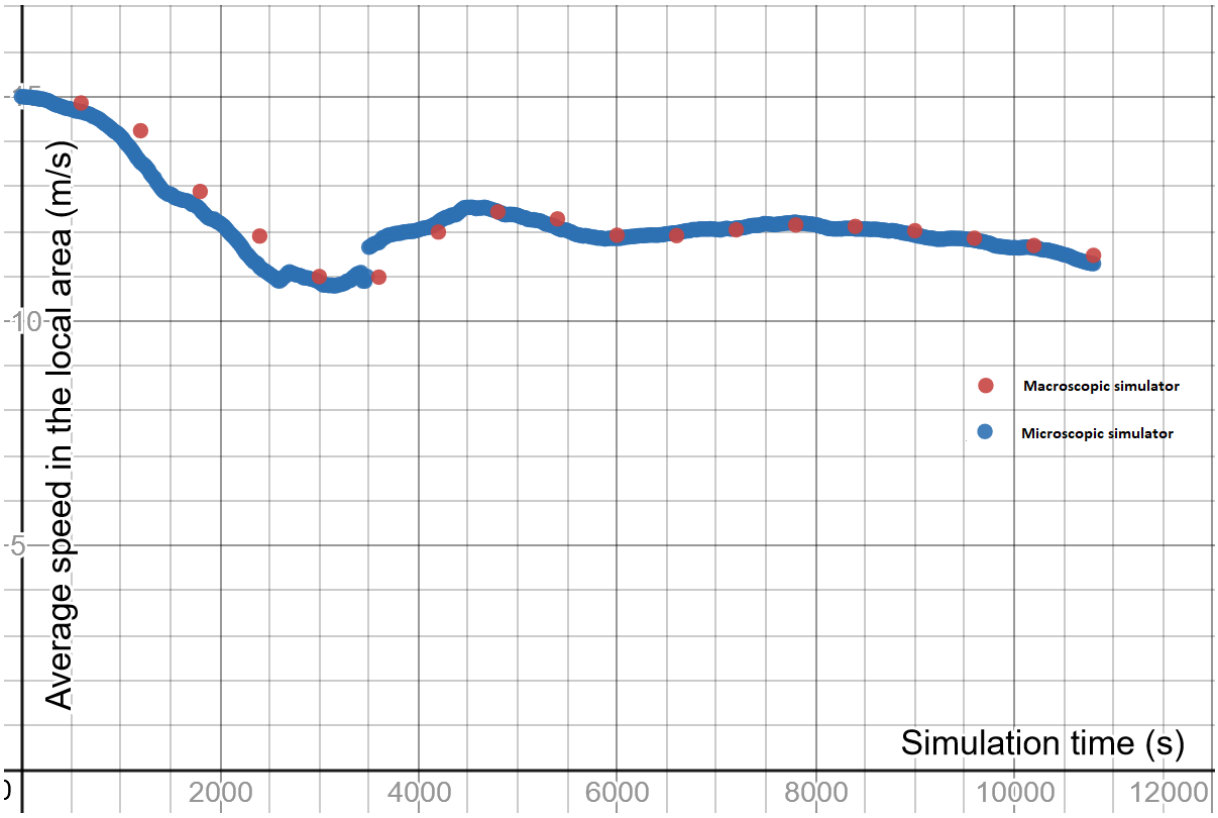

FIGURE 5.16 – Les deux simulateurs interagissent, avec la demande supplémentaire

Trois configurations sont proposées ici :

1. Le simulateur macroscopique seul,
2. le simulateur macroscopique en interaction avec le premier simulateur microscopique
3. le simulateur macroscopique en interaction avec le second simulateur microscopique.

Le simulateur macroscopique a été défini précédemment, le premier simulateur microscopique est également celui que nous avons déjà utilisé, quant au dernier il s'agit d'un autre simulateur microscopique basé sur le modèle de conducteur intelligent (IDM) décrit dans [Derbel *et al.*2013].

Le premier objectif a été atteint puisque les deux simulateurs microscopiques ont pu être utilisés sans modifier l'intergiciel ou le simulateur macroscopique. Comme nous l'observons dans la figure 5.17, l'intergiciel corrige la vitesse du simulateur macroscopique à l'aide de deux modèles de poursuite différents, qui donnent des résultats proches. Le temps de simulation du simulateur macroscopique seul est de 20ms, ceux des simulateurs microscopiques d'environ 3000ms et le temps de simulation des simulations multi-échelles de

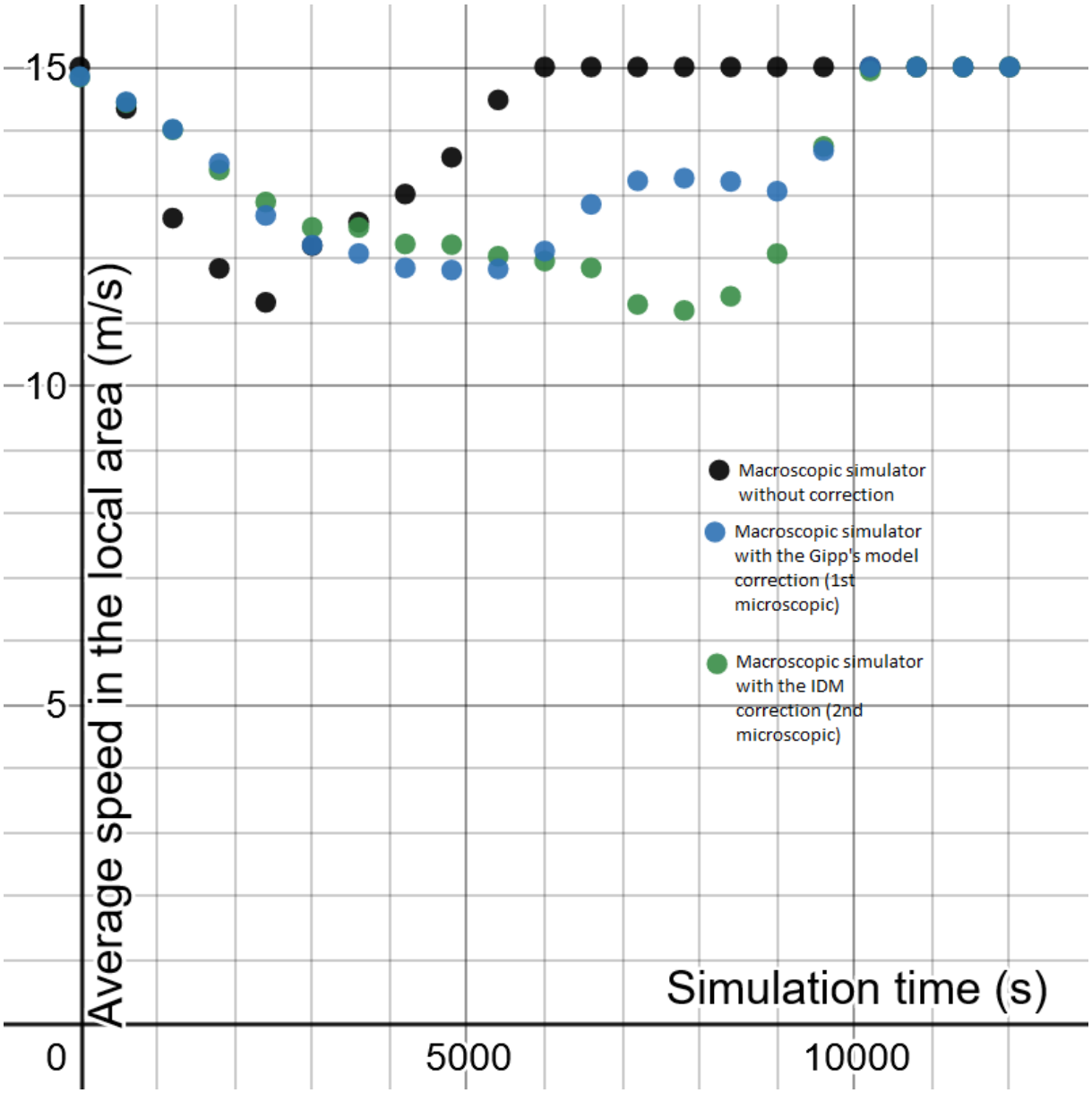

FIGURE 5.17 – Résultats macroscopiques avec et sans correction

30.000ms, ce ralentissement est dû à la synchronisation des simulateurs qui doivent s'attendre mutuellement. Finalement, l'intergiciel classe les deux simulateurs microscopiques en calculant une correction de 13% pour le premier et de 16% pour le second ce qui conclue que le premier simulateur microscopique serait plus pertinent pour ce simulateur macroscopique.

5.4.2 Utilisation avec MATSim et SM4T

Les expérimentations précédentes ont permis de montrer l'efficacité de l'intergiciel avec des simulateurs simplifiés, développés dans le cadre de cette thèse. Nous avons ainsi pu vérifier qu'il fonctionnait comme prévu et qu'il était réutilisable pour des simulateurs différents. Dans cette section, nous essayons de l'utiliser sur des simulateurs plus complexes utilisés dans de vraies études de transport, viz. SM4T [Zargayouna *et al.*2014] et MATSim [Balmer *et al.*2009]. L'objectif de ces expérimentations est d'évaluer la facilité d'intégration de deux simulateurs différents en listant les modifications à leur apporter pour les faire fonctionner ensemble.

Le premier simulateur SM4T joue le rôle de simulateur régional. SM4T est un simulateur multi-agents développé par le laboratoire GRETTIA de l'université Gustave Eiffel pour la mobilité multimodale des voyageurs. Il simule donc le déplacement des voyageurs sur les réseaux routiers et de transport public en prenant en compte les changements de temps de parcours sur ces réseaux au cours du temps. Il est fondé sur la plateforme multi-agents Repast Symphony [North *et al.*2005]. Plusieurs paramètres d'entrée sont nécessaires à la simulation (durée de la simulation, nombre d'agents sur chaque réseau, la vitesse par défaut des agents, etc.). Le pas de temps est fixé au début de la simulation et lors de chaque pas de temps les agents avancent, cadencés par un ordonnanceur qui décide de l'ordre d'exécution des méthodes des différents agents. Contrairement à nos simulateurs simplifiés, les transports en commun dans SM4T suivent un tableau de marche qui est un paramètre d'entrée et leur vitesse est directement calculée à partir de celui-ci (ces transports en commun ne peuvent donc pas être en retard). L'affectation se fait par un K plus court chemin au moment de la création des agents. Les véhicules privés quant à eux suivent un diagramme fondamental dans leur déplacement. Les piétons ont une vitesse constante (paramètre).

Afin de faire fonctionner SM4T avec l'intergiciel et MATSim, il a fallu effectuer plusieurs modifications. Dans un premier temps, il a évidemment fallu implémenter l'interface proposée dans la section 5.2.2 en donnant au simulateur la possibilité de charger et de corriger les différents paramètres ainsi qu'un mécanisme d'attente. Deux autres principaux changements ont été nécessaires :

— La création des agents ainsi que leur affectation se faisait au début de la simulation en fonction de l'état initial du réseau de transport, cela a été modifié pour que les agents soient créés et affectés au moment où ils deviennent actifs dans la simulation

de manière à pouvoir modifier la demande et l'affectation de manière dynamique entre deux pas de temps.

— Un mécanisme de modification des tableaux de marche des transports en commun pendant l'exécution a été ajouté. Pour corriger le mouvement de ces transports pour suivre ceux de MATSim, l'ancien fonctionnement n'était pas suffisant.

Au final, le simulateur SM4T a eu besoin de peu de modifications car il correspond au schéma envisagé avec une demande fixe qui est affectée, puis se déplace. La seule modification en dehors de l'implémentation de l'interface est le changement des tableaux de marche des transports en commun.

Le second simulateur utilisé pour valider l'utilisation de l'intergiciel est MATSim (*MultiAgent Transport Simulation*) qui est un simulateur multi-agents qui est très utilisé dans la littérature ([Balmer *et al.*2008],[Balmer *et al.*2009],[Horni *et al.*2016]). Contrairement à tous les simulateurs utilisés jusqu'à présent, MATSim utilise un modèle basé sur les activités, il possède donc un fichier d'entrée avec des voyageurs et les différentes activités que vont effectuer ces derniers au cours de la simulation. Ainsi, contrairement à SM4T ou les voyageurs ne font qu'un trajet d'une origine à une destination puis disparaissent, les voyageurs dans MATSim peuvent effectuer plusieurs trajets au cours de la simulation. De plus, le modèle de déplacement utilisé pour les véhicules privés dans MATSim est basé sur les files d'attente, ce qui signifie que l'arc routier est séparé en deux parties, une partie dans laquelle les véhicules vont à la vitesse maximale et une partie dans laquelle ils sont en attente de pouvoir passer à l'arc suivant avec un système FIFO. La taille des deux parties de l'arc est calculée en fonction du nombre de véhicules en attente et le nombre de véhicules sortant à chaque pas de temps dépend de la capacité des arcs devant les accueillir.

Comme pour tous les autres simulateurs, il a fallu implémenter l'interface de l'intergiciel et le mécanisme d'attente. Les autres modifications nécessaires ont été :

— Lorsque MATSim corrige la demande, il envoie des informations concernant le trajet des voyageurs uniquement jusqu'à leur prochaine activité. Lorsque le voyageur atteint l'activité, il est détruit dans SM4T et lorsqu'il quitte l'activité MATSim renvoie des informations le concernant.

— Dans l'autre sens lorsque SM4T corrige la demande, MATSim crée des voyageurs avec une seule activité pour la simulation.

— La vitesse moyenne d'un arc ne peut pas être modifiée directement dans un modèle basé sur les files d'attente. Lorsque SM4T corrige la vitesse, on modifie de manière

dynamique la capacité des arcs côté MATSim. Ainsi, plus de véhicules quittent la file d'attente et on obtient une vitesse moyenne cohérente.

Au final, l'utilisation de ces deux simulateurs demande quelques modifications principalement liées aux modèles de déplacements qu'ils utilisent. L'intergiciel ne prend en compte que la vitesse moyenne de chaque arc afin d'être le plus générique possible. Cependant, cette généricité à un coût, la nécessité de transmettre cette vitesse moyenne qui peut demander quelques modifications du modèle de déplacement ou la modification au cours de l'exécution de paramètres qui devraient être constants.

5.5 Conclusion

Dans ce chapitre, nous avons instancié le modèle d'intergiciel développé dans le chapitre précédent. D'abord, nous avons détaillé les spécificités techniques d'un intergiciel implémentant le modèle, et nous discutons de l'architecture distribuée permettant son déploiement. Notre choix s'est porté sur les architectures orientées-service et les services Web WS-*. Ensuite, nous avons exécuté un ensemble d'expérimentations avec de nouveaux simulateurs développés spécifiquement dans le cadre de cette thèse, ainsi que deux simulateurs existants. Il apparaît de notre étude que la mise en place d'un intergiciel pour coupler deux simulateurs de mobilité a un coût, et demande quand-même des efforts de développements pour que l'ensemble fonctionne correctement. Il n'en reste pas moins que ces efforts sont sans commune mesure avec un développement d'un nouveau simulateur multi-niveaux *ex-nihilo*.

Conclusion générale et perspectives

6 Résumé

La question des échelles de simulation est primordiale dans les études de transport afin d'adapter au mieux les modèles utilisés aux problèmes que l'on souhaite traiter. Certaines études nécessitent des simulations multi-échelles afin d'obtenir les avantages combinés des différentes échelles et de changer de point de vue d'observation selon la zone simulée. Ainsi, plusieurs solutions *ad hoc* pour des cas précis existent dans la littérature.

Cette thèse propose un modèle et un outil générique pouvant servir à de nombreux cas de simulations multi-échelles. La solution proposée est un intergiciel permettant de faire fonctionner ensemble deux simulateurs existants d'échelles différentes en leur apportant le minimum de modifications. Cet intergiciel coordonne les simulateurs sur l'axe temporel, parfois en demandant à l'un des deux de revenir à un état de simulation antérieur ou d'attendre le second. Les expérimentations menées montrent que l'intergiciel permet d'améliorer la pertinence des corrections mutuelles des simulateurs et la facilité relative d'intégration avec des simulateurs existants. Lorsqu'un ordonnancement avancé des simulateurs est nécessaire, la coordination des simulateurs vient au prix d'un temps d'exécution plus important, lié à l'attente mutuelle des simulations et à la réexécution de certains pas de temps.

7 Contributions

Dans cette thèse nous avons apporté les contributions suivantes :

1. Nous avons déterminé trois axes d'agrégation permettant de définir des échelles de simulation et de classer les simulateurs de manière précise. Nous avons mis en

évidence les problématiques à résoudre pour le couplage multi-échelles correspondant à ces trois axes.

2. Nous avons proposé de vérifier la cohérence entre les différentes simulations, par la correction mutuelle entre simulateurs concernant trois aspects : la demande, l'affectation et le déplacement (la vitesse) des voyageurs. Bien que d'autres facteurs puissent être importants lors d'études de problématiques précises - comme l'émission de CO_2 dans une étude portant sur la pollution -, ces trois facteurs assurent que les phénomènes de mobilités apparaissant dans un simulateur se retrouvent dans l'autre.
3. Nous avons spécifié un modèle d'intergiciel qui assure le couplage et la cohérence entre simulateurs. L'intergiciel permet de traduire les données d'une échelle à l'autre grâce au travail sur les trois axes d'agrégation.
4. L'architecture distribuée orientée service est étudiée dans le cadre des simulations multi-échelles de mobilité afin d'y intégrer l'intergiciel évoqué. L'utilisation de services web permet de réutiliser facilement l'intergiciel, mais également d'éventuellement combiner un simulateur client avec un ensemble de simulateurs services.
5. Plusieurs expérimentations appuient la partie théorique de cette thèse. Tout d'abord quelques expériences sont présentées sur les choix de conception de l'intergiciel, ces expériences ont été faites sur des simulateurs simplifiés développés dans le cadre de cette thèse. La facilité de réutilisation de l'intergiciel est également testée sur des simulateurs complexes et utilisés fréquemment dans le cadre de véritables études, MATSim et SM4T.

Cette thèse propose un cadre précis de simulation multi-échelles en définissant la topologie du réseau ainsi que les objectifs de la simulation et propose un intergiciel permettant de réaliser une simulation multi-échelles dans ce cadre. Cet intergiciel a été implémenté dans le cadre de plusieurs simulations multi-échelles avec des simulateurs simplifiés ainsi que des simulateurs existants. Les expérimentations ont validé certains choix de conception et les résultats attendus.

8 Limites et pistes de prolongement

Nous proposons dans cette thèse une solution générique et réutilisable pouvant aider au couplage de simulateurs existants, cette solution connaît cependant des limites que nous proposons de discuter ci-dessous.

8.1 Généricité

La solution technique d'intergiciel développée dans cette thèse est focalisée à un cas particulier de simulation multi-échelles. Nous avons proposé le couplage d'un simulateur travaillant sur une zone avec un second simulateur travaillant sur une partie de cette zone, en cohérence avec le projet MSM. Les deux simulateurs simulent donc en même temps la zone d'intérêt, c'est un cas particulier et il en existe d'autres, couverts par le modèle, mais pas par les expérimentations (zones simulés différentes et exclusives).

Nous limitons également le cas d'utilisation à un modèle de simulation où la demande est représentée par des matrices OD, c'est à dire que les voyageurs font un seul trajet durant la simulation. Avec un modèle basé sur les activités, les voyageurs pourraient faire plusieurs trajets et éventuellement sortir de la zone d'intérêt pour y rentrer à nouveau plus tard. Il y aurait alors la problématique de la perte d'informations lors du passage local-régional qu'il faudrait reconstituer lors du passage régional-local.

8.2 Limites d'utilisation de la solution

Malgré la volonté de rendre l'intergiciel le plus facile à utiliser possible, il reste intrusif puisqu'il ordonnance l'exécution des simulateurs et gère les transferts de données. Les simulateurs doivent être modifiés de manière à prendre en compte des données nouvelles à chaque pas de temps, à posséder une phase de correction, une phase d'exécution, un mécanisme d'attente et même à être capable d'effectuer un retour en arrière.

De plus cette solution se veut générique et ne corrige le mouvement d'un simulateur à l'autre que grâce à la transmission de la vitesse moyenne par arc. On a vu dans le chapitre 5 que cela rentrait en conflit avec notre objectif de facilité d'utilisation puisque plusieurs changements internes ont été nécessaires sur certains simulateurs. Une amélioration possible serait créer plusieurs modules au sein de l'intergiciel qui proposeraient une solution en fonction du modèle de déplacement des simulateurs qui serait alors un nouveau paramètre de configuration.

Finalement la solution se révèle assez lourde en augmentant considérablement les temps d'exécution de la simulation à cause de la synchronisation des simulateurs et des retours en arrière.

8.3 Perspectives

Indépendamment du monde des transports, la question des couplages de simulations spatialisées est un objet de recherche très riche, qui mériteraient de plus amples développements. Quelque soit le modèle et la simulation envisagé, il existe des facteurs exogènes qui viennent influencer le modèle considéré. Comme la modélisation fine des phénomènes demande un grand effort de calibration et est très gourmande en données, il n'est pas envisageable d'avoir des modèles globaux fermés. Seuls des sous-ensemble du monde modélisable peuvent faire partie d'un modèle ouvert. Dans un modèle épidémiologique par exemple, la simulation d'une zone (une ville, un pays, un continent) dépend de la dynamique des zones limitrophes. La question générale de l'interaction entre modèles est donc appelée à être fortement développée dans les années qui viennent, et nous espérons pouvoir y contribuer.

Bibliographie

[Abouaïssa *et al.*2014] Hassane Abouaïssa, Yoann Kubera, and Gildas Morvan. Dynamic hybrid traffic flow modeling. *arXiv preprint arXiv :1401.6773*, 2014.

[Adnan *et al.*2016] Muhammad Adnan, Francisco C Pereira, Carlos Miguel Lima Azevedo, Kakali Basak, Milan Lovric, Sebastián Raveau, Yi Zhu, Joseph Ferreira, Christopher Zegras, and Moshe Ben-Akiva. Simmobility : A multi-scale integrated agent-based simulation platform. In *95th Annual Meeting of the Transportation Research Board Forthcoming in Transportation Research Record*, 2016.

[Al-Jaroodi and Mohamed2012a] Jameela Al-Jaroodi and Nader Mohamed. Service-oriented middleware : A survey. *Journal of Network and Computer Applications*, 35(1) :211–220, 2012.

[Al-Jaroodi and Mohamed2012b] Jameela Al-Jaroodi and Nader Mohamed. Service-oriented middleware : A survey. *Journal of Network and Computer Applications*, 35(1) :211–220, 2012.

[Azevedo *et al.*2017] Carlos Lima Azevedo, Neeraj Milind Deshmukh, Balakumar Marimuthu, Simon Oh, Katarzyna Marczuk, Harold Soh, Kakali Basak, Tomer Toledo, Li-Shiuan Peh, and Moshe E Ben-Akiva. Simmobility short-term : An integrated microscopic mobility simulator. *Transportation Research Record*, 2622(1) :13–23, 2017.

[Balakrishna *et al.*2007] Ramachandran Balakrishna, Constantinos Antoniou, Moshe Ben-Akiva, Haris N Koutsopoulos, and Yang Wen. Calibration of microscopic traffic simulation models : Methods and application. *Transportation Research Record*, 1999(1) :198–207, 2007.

[Balci1994] Osman Balci. Validation, verification, and testing techniques throughout the life cycle of a simulation study. *Annals of operations research*, 53(1) :121–173, 1994.

[Balmer *et al.*2008] Michael Balmer, Konrad Meister, Marcel Rieser, Kai Nagel, and Kay W Axhausen. Agent-based simulation of travel demand : Structure and com-

putational performance of matsim-t. *Arbeitsberichte Verkehrs-und Raumplanung*, 504, 2008.

[Balmer *et al.*2009] Michael Balmer, Marcel Rieser, Konrad Meister, David Charypar, Nicolas Lefebvre, and Kai Nagel. Matsim-t : Architecture and simulation times. In *Multi-agent systems for traffic and transportation engineering*, pages 57–78. IGI Global, 2009.

[Beil and Kolbe2017] Christof Beil and Thomas H Kolbe. Citygml and the streets of new york-a proposal for detailed street space modelling. In *Proceedings of the 12th International 3D GeoInfo Conference 2017*, pages 9–16, 2017.

[Belgacem *et al.*2013] Mohamed Ben Belgacem, Bastien Chopard, Joris Borgdorff, Mariusz Mamoński, Katarzyna Rycerz, and Daniel Harezlak. Distributed multiscale computations using the mapper framework. *Procedia Computer Science*, 18 :1106–1115, 2013.

[Benjaafar *et al.*1997] Saifallah Benjaafar, Kevin Dooley, and Wibowo Setyawan. Cellular automata for traffic flow modeling. 1997.

[Biedermann *et al.*2014] Daniel H Biedermann, Peter M Kielar, Oliver Handel, and André Borrmann. Towards transitum : A generic framework for multiscale coupling of pedestrian simulation models based on transition zones. *Transportation Research Procedia*, 2 :495–500, 2014.

[Biedermann *et al.*2016] Daniel H Biedermann, Carolin Torchiani, Peter M Kielar, David Willems, Oliver Handel, Stefan Ruzika, and André Borrmann. A hybrid and multiscale approach to model and simulate mobility in the context of public events. *Transportation Research Procedia*, 19 :350–363, 2016.

[Bishop and Karne2003] Toni A Bishop and Ramesh K Karne. A survey of middleware. In *Computers and Their Applications*, pages 254–258, 2003.

[Bonabeau2002] Eric Bonabeau. Agent-based modeling : Methods and techniques for simulating human systems. *Proceedings of the national academy of sciences*, 99(suppl 3) :7280–7287, 2002.

[Borgdorff *et al.*2014] Joris Borgdorff, Mariusz Mamonski, Bartosz Bosak, Krzysztof Kurowski, M Ben Belgacem, Bastien Chopard, Derek Groen, Peter V Coveney, and Alfons G Hoekstra. Distributed multiscale computing with muscle 2, the multiscale coupling library and environment. *Journal of Computational Science*, 5(5) :719–731, 2014.

[Bourrel and Henn2002] Emmanuel Bourrel and Vincent Henn. Mixing micro and macro representations of traffic flow : a first theoretical step. In *Proceedings of the 9th meeting of the Euro Working Group on Transportation*, pages 610–616, 2002.

[Bourrel and Lesort2003] Emmanuel Bourrel and Jean-Baptiste Lesort. Mixing microscopic and macroscopic representations of traffic flow : hybrid model based on lighthill–whitham–richards theory. *Transportation Research Record*, 1852(1) :193–200, 2003.

[Brockfeld *et al.*2005] Elmar Brockfeld, Reinhart D Kühne, and Peter Wagner. Calibration and validation of microscopic models of traffic flow. *Transportation Research Record*, 1934(1) :179–187, 2005.

[Burghout and Koutsopoulos2006] Wilco Burghout and Harilaos Koutsopoulos. Hybrid traffic simulation models : Vehicle loading at meso–micro boundaries. In *at International Symposium of Transport Simulation*, 2006.

[Burghout2004] Wilco Burghout. *Hybrid microscopic-mesoscopic traffic simulation*. PhD thesis, KTH, 2004.

[Castiglione *et al.*2015] Joe Castiglione, Mark Bradley, and John Gliebe. *Activity-based travel demand models : A primer*. Number SHRP 2 Report S2-C46-RR-1. 2015.

[Connors and Watling2015] Richard D Connors and David P Watling. Assessing the demand vulnerability of equilibrium traffic networks via network aggregation. *Networks and Spatial Economics*, 15(2) :367–395, 2015.

[Derbel *et al.*2013] Oussama Derbel, Tamas Peter, Hossni Zebiri, Benjamin Mourllion, and Michel Basset. Modified intelligent driver model for driver safety and traffic stability improvement. *IFAC Proceedings Volumes*, 46(21) :744–749, 2013.

[Emmerich and Ellmer2019] Wolfgang Emmerich and Ernst Ellmer. A survey of object-oriented middleware. 03 2019.

[Falcone *et al.*2010] Jean-Luc Falcone, Bastien Chopard, and Alfons Hoekstra. Mml : towards a multiscale modeling language. *Procedia Computer Science*, 1(1) :819–826, 2010.

[Gaud *et al.*2008] Nicolas Gaud, Stéphane Galland, Franck Gechter, Vincent Hilaire, and Abderrafiaa Koukam. Holonic multilevel simulation of complex systems : Application to real-time pedestrians simulation in virtual urban environment. *Simulation Modelling Practice and Theory*, 16(10) :1659–1676, 2008.

[Geroliminis and Daganzo2008] Nikolas Geroliminis and Carlos F Daganzo. Existence of urban-scale macroscopic fundamental diagrams : Some experimental findings. *Transportation Research Part B : Methodological*, 42(9) :759–770, 2008.

[Gong *et al.*2006] JX Gong, Wei Zhong, Jian Huang, XG Qiu, and KD Huang. Application of base object model in hlaâs simulations. *Journal of System Simulation*, 18(suppl 2) :327–331, 2006.

[Gröger *et al.*2012] Gerhard Gröger, Thomas H Kolbe, Claus Nagel, and Karl-Heinz Häfele. Ogc city geography markup language (citygml) encoding standard. 2012.

[Haman *et al.*2017] Igor Tchappi Haman, Vivient Corneille Kamla, Stéphane Galland, and Jean Claude Kamgang. Towards an multilevel agent-based model for traffic simulation. *Procedia Computer Science*, 109 :887–892, 2017.

[Holmgren *et al.*2014] Johan Holmgren, Linda Ramstedt, Paul Davidsson, Henrik Edwards, and Jan A Persson. Combining macro-level and agent-based modeling for improved freight transport analysis. *Procedia Computer Science*, 32 :380–387, 2014.

[Hoogendoorn and Hoogendoorn2010] Serge Hoogendoorn and Raymond Hoogendoorn. Calibration of microscopic traffic-flow models using multiple data sources. *Philosophical Transactions of the Royal Society A : Mathematical, Physical and Engineering Sciences*, 368(1928) :4497–4517, 2010.

[Horni *et al.*2016] Andreas Horni, Kai Nagel, and Kay W Axhausen. *The multi-agent transport simulation MATSim*. Ubiquity Press London, 2016.

[Isaksen and Payne1973] LEIF Isaksen and H Payne. Suboptimal control of linear systems by augmentation with application to freeway traffic regulation. *IEEE Transactions on Automatic Control*, 18(3) :210–219, 1973.

[Jayakrishnan *et al.*1994] R Jayakrishnan, Hani S Mahmassani, and Ta-Yin Hu. An evaluation tool for advanced traffic information and management systems in urban networks. *Transportation Research Part C : Emerging Technologies*, 2(3) :129–147, 1994.

[Jayakrishnan *et al.*2001] R Jayakrishnan, Jun-Seok Oh, and Abd-El-Kader Sahraoui. Calibration and path dynamics issues in microscopic simulation for advanced traffic management and information systems. *Transportation Research Record*, 1771(1) :9–17, 2001.

[Joueiai *et al.*2014] Mahtab Joueiai, Hans Van Lint, and Serge P Hoogendoom. Multiscale traffic flow modeling in mixed networks. *Transportation Research Record*, 2421(1) :142–150, 2014.

[Klügl2008] Franziska Klügl. A validation methodology for agent-based simulations. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 39–43, 2008.

[Kon *et al.*2001] Fabio Kon, Roy H Campbell, et al. Reflective middleware : From your desk to your hand. *IEEE Distributed Systems Online*, (5) :null, 2001.

[Law2005] A. M. Law. How to build valid and credible simulation models. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 9 pp.–, Dec 2005.

[Lebacque *et al.*2007] Jean-Patrick Lebacque, Salim Mammar, and Habib Haj-Salem. The aw–rascle and zhangâs model : Vacuum problems, existence and regularity of the solutions of the riemann problem. *Transportation Research Part B : Methodological*, 41(7) :710–721, 2007.

[Lighthill and Whitham1955a] Michael James Lighthill and Gerald Beresford Whitham. On kinematic waves ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178) :317–345, 1955.

[Lighthill and Whitham1955b] Michael James Lighthill and Gerald Beresford Whitham. On kinematic waves ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178) :317–345, 1955.

[Lok and Brent2005] Larry Lok and Roger Brent. Automatic generation of cellular reaction networks with moleculizer 1.0. *Nature biotechnology*, 23(1) :131–136, 2005.

[Lu *et al.*2015] Yang Lu, Muhammad Adnan, Kakali Basak, Francisco Câmara Pereira, Carlos Carrion, Vahid Hamishagi Saber, Harish Loganathan, and Moshe E Ben-Akiva. Simmobility mid-term simulator : A state of the art integrated agent based demand and supply model. In *94th Annual Meeting of the Transportation Research Board, Washington, DC*, 2015.

[Mammar *et al.*2006] Salim Mammar, Jean-Patrick Lebacque, and Habib Haj-Salem. Hybrid model based on second-order traffic model. Technical report, 2006.

[McNally2000] Michael G McNally. The four step model. *Handbook of transport modelling*, 1 :35–41, 2000.

[M.D. Hall1998] D. van Vliet M.D. Hall. The saturn users manual. 1998.

[Mittal2007] Saurabh Mittal. Devs unified process for integrated development and testing of service oriented architectures. 2007.

[Montero *et al.*1998] Lidia Montero, E Codina, J Barceló, and P Barceló. Combining macroscopic and microscopic approaches for transportation planning and design of road networks. In *Proceedings of the 19 th ARRB transport research conference, Sydney*, 1998.

[Muqsith *et al.*2011] Mohammed A Muqsith, Hessam S Sarjoughian, Dazhi Huang, and Stephen S Yau. Simulating adaptive service-oriented software systems. *Simulation*, 87(11) :915–931, 2011.

[Navarro *et al.*2011] Laurent Navarro, Fabien Flacher, and Vincent Corruble. Dynamic level of detail for large scale agent-based urban simulations. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 701–708. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[Navarro *et al.*2013] Laurent Navarro, Vincent Corruble, Fabien Flacher, and Jean-Daniel Zucker. A flexible approach to multi-level agent-based simulation with the mesoscopic representation. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 159–166. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[Ni2011] Daiheng Ni. Multiscale modeling of traffic flow. *Mathematica Aeterna*, 1(1) :27–54, 2011.

[Noël *et al.*2018] Romain Noël, Laurent Navarro, and Guy Courbebaisse. Lattice boltzmann method for heterogeneous multi-class traffic flow. *arXiv preprint arXiv :1809.11106*, 2018.

[North *et al.*2005] Michael J North, Thomas R Howe, Nick T Collier, and Jerry R Vos. The repast simphony development environment. In *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, volume 13, page 15. Citeseer, 2005.

[Paul *et al.*2005] G Paul, C Tram, and C Karl. Moving towards a service-oriented architecture (soa) for distributed component simulation environments. In *Proceedings of the 2005 SISO Spring Simulation Interoperability Workshop, San Diego, CA*, 2005.

[Payne1971] Harold J Payne. Model of freeway traffic and control. *Mathematical Model of Public System*, pages 51–61, 1971.

[Poschinger *et al.*2002] A Poschinger, R Kates, and H Keller. Coupling of concurrent macroscopic and microscopic traffic flow models using hybrid stochastic and deterministic disaggregation. In *Transportation and Traffic Theory in the 21st Century. Proceedings of the 15th International Symposium on Transportation and Traffic TheoryUniversity of Adelaide*, 2002.

[Sarjoughian *et al.*2008] Hessam Sarjoughian, Sungung Kim, Muthukumar Ramaswamy, and Stephen Yau. A simulation framework for service-oriented computing systems. In *Proceedings of the 40th Conference on Winter Simulation*, pages 845–853. Winter Simulation Conference, 2008.

[Setola *et al.*2016] Roberto Setola, Vittorio Rosato, Elias Kyriakides, and Erich Rome. *Managing the complexity of critical infrastructures*, volume 90. Springer, 2016.

[Sewall *et al.*2011] Jason Sewall, David Wilkie, and Ming C Lin. Interactive hybrid simulation of large-scale traffic. *ACM Transactions on Graphics (TOG)*, 30(6) :135, 2011.

[Sharpanskykh and Treur2011] Alexei Sharpanskykh and Jan Treur. Group abstraction for large-scale agent-based social diffusion models. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 830–837. IEEE, 2011.

[Shekhar *et al.*2016] Shashank Shekhar, Hamzah Abdel-Aziz, Michael Walker, Faruk Caglar, Aniruddha Gokhale, and Xenofon Koutsoukos. A simulation as a service cloud middleware. *Annals of Telecommunications*, 71(3-4) :93–108, 2016.

[sim] Simmobility.

[Smith *et al.*1995] Mark Smith, Gordon Duncan, and Stephen Druitt. Paramics : microscopic traffic simulation for congestion management. 1995.

[Soyez2013] Jean Baptiste Soyez. *Conception et modélisation de systèmes de systèmes : une approche multi-agents multi-niveaux.* PhD thesis, Université de Lille 1, 2013.

[Szeto and Wong2012] WY Szeto and SC Wong. Dynamic traffic assignment : model classifications and recent advances in travel choice principles. *Central European Journal of Engineering*, 2(1) :1–18, 2012.

[Tordeux2010] Antoine Tordeux. A study of continuous-time processes modelling traffic flow. 06 2010.

[Treiber and Kesting2013] Martin Treiber and Arne Kesting. Traffic flow dynamics. *Traffic Flow Dynamics : Data, Models and Simulation, Springer-Verlag Berlin Heidelberg*, 2013.

[Tsai *et al.*2006] Wei-Tek Tsai, Chun Fan, Yinong Chen, and Ray Paul. Ddsos : A dynamic distributed service-oriented simulation framework1. In *Proceedings of the 39th annual Symposium on Simulation*, pages 160–167. IEEE Computer Society, 2006.

[Wang *et al.*2011] Wenguang Wang, Weiping Wang, Yifan Zhu, and Qun Li. Service-oriented simulation framework : An overview and unifying methodology. *Simulation*, 87(3) :221–252, 2011.

[yu and Fan2017] Miao yu and Wei Fan. Calibration of microscopic traffic simulation models using metaheuristic algorithms. *International Journal of Transportation Science and Technology*, 6, 05 2017.

[Zargayouna *et al.*2014] Mahdi Zargayouna, Besma Zeddini, Gérard Scemama, and Amine Othman. Simulating the impact of future internet on multimodal mobility. In

2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA), pages 230–237. IEEE, 2014.

Résumé

Les études sur la mobilité concernent généralement de grandes zones géographiques et considèrent des réseaux de transports étendus et multimodaux. Par ailleurs, des problématiques de mobilité très locales, à l'échelle d'un quartier, nécessitent une modélisation particulière et permettent une simulation réaliste des déplacements et une supervision à plus petite échelle. D'autres contextes de mobilité, appellent à différentes échelles de représentation de lespace, du temps et des entités modélisées. Cette question des échelles de simulation est primordiale dans les études de transport afin d'adapter au mieux les modèles utilisés aux problèmes traités. Certaines études nécessitent des simulations multi-échelles afin d'obtenir les avantages combinés des différentes échelles et de changer de point de vue d'observation selon la zone simulée. Ainsi, plusieurs solutions ad hoc pour des cas précis existent dans la littérature. Cette thèse propose un modèle et un outil générique pouvant servir à de nombreux cas de simulations multi-échelles. La solution proposée est un intergiciel permettant de faire fonctionner ensemble deux simulateurs existants d'échelles différentes en leur apportant le minimum de modifications. Lintergiciel coordonne les simulateurs et permet lexécution dune simulation multi-échelles correcte et cohérente. Les expérimentations menées montrent que l'intergiciel permet d'améliorer la pertinence des corrections mutuelles des simulateurs et la facilité relative d'intégration avec des simulateurs existants. Lorsqu'un ordonnancement avancé des simulateurs est nécessaire, la coordination des simulateurs vient au prix d'un temps d'exécution plus important, lié à lattente mutuelle des simulations et à la réexécution de certains pas de temps.

Abstract

Mobility surveys and studies generally concern large geographical areas and consider extensive and multimodal transport networks. In addition, very local mobility issues, on a neighborhood scale, require special modeling and allow realistic simulation of travel and supervision on a smaller scale. Other mobility contexts call for different scales of representation of space, time, and modeled entities. This question of simulation scales is crucial in transport studies in order to best adapt the used models to the addressed problems. Some studies require multi-scale simulations in order to obtain the combined advantages of the different scales and to change the observation point of view according to the simulated area. Thus, several ad hoc solutions for specific cases exist in the literature. This thesis proposes a model and a generic tool that can be used for many cases of multi-scale simulations. The proposed solution is a middleware allowing to make two existing simulators of different scales work together with minimal modifications. The middleware coordinates the simulators and allows the execution of a correct and consistent multi-scale simulation. Experiments show that the middleware improves the relevance of the mutual corrections of the simulators and the relative ease of integration with existing simulators. When advanced scheduling of simulators is required, coordination of simulators comes at the cost of higher execution time, linked to the mutual waiting of simulations and the re-execution of certain time steps.