

Normer pour mieux varier ? La différenciation comportementale par les normes, et son application au trafic dans les simulateurs de conduite

Benoit Lacroix

▶ **To cite this version:**

Benoit Lacroix. Normer pour mieux varier ? La différenciation comportementale par les normes, et son application au trafic dans les simulateurs de conduite. Informatique [cs]. Université des Sciences et Technologie de Lille - Lille I, 2009. Français. NNT : . tel-00527846

HAL Id: tel-00527846

https://theses.hal.science/tel-00527846

Submitted on 20 Oct 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Normer pour mieux varier ?

La différenciation comportementale par les normes, et son application au trafic dans les simulateurs de conduite

THÈSE

présentée et soutenue publiquement le 1er octobre 2009

pour l'obtention du

Doctorat de l'Université des Sciences et Technologies de Lille

(spécialité informatique)

par

Benoît Lacroix

Composition du jury

Président :	Pr. René Mandiau	Université de Valenciennes et du Hainaut-Cambrésis
Rapporteurs :	Pr. Olivier Boissier	École Nationale Supérieure des Mines de St Étienne
	Pr. Marie-Pierre Gleizes	Université Paul Sabatier (Toulouse 3)
Examinateur :	Dr. Viola Cavallo	INRETS
Directeurs :	Dr. Andras Kemeny	RENAULT SAS
	Pr. Philippe Mathieu	Université Lille 1 - Sciences et Technologies

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE
Laboratoire d'Informatique Fondamentale de Lille — UMR USTL/CNRS 8022
UFR d'IEEA – Bât. M3 – 59655 VILLENEUVE D'ASCQ Cedex – France
Tél. : +33 (0)3 28 77 85 41 – Télécopie : +33 (0)3 28 77 85 37 – email : direction@lifl.fr

Mis en page et composé avec LaTeX et la classe thlifl.

Résumé

Les simulateurs de conduite sont utilisés par l'industrie automobile pour le développement de systèmes d'aide à la conduite, des études d'ergonomie, de design, ou encore du comportement des conducteurs. L'objectif est d'améliorer la sécurité des véhicules, et de réduire coûts et délais des projets. Dans les simulateurs, le conducteur évolue dans un trafic simulé dont le réalisme est crucial pour la validité des résultats : les réactions d'un conducteur sont d'autant plus correctes qu'il perçoit l'environnement comme réel.

Dans les approches centrées individu, comme ici le trafic, un des critères important pour le réalisme est la variété comportementale des agents. De plus, les comportements doivent également être cohérents : l'apparition de situations aberrantes est très pénalisante pour l'immersion. Cependant, ces deux aspects ne sont généralement pas pris en compte simultanément. Dans ce travail, nous nous sommes donc intéressés à la question suivante : dans les approches centrées individus, augmenter la variété des comportements des agents tout en préservant leur cohérence améliore-t-il le réalisme ? Par ailleurs, le contexte industriel des travaux, effectués en convention Cifre chez Renault, impliquait des besoins spécifiques : il fallait concevoir un outil qui permette à la fois aux experts de spécifier des comportements variés et cohérents, et aux utilisateurs finaux de les exploiter facilement.

Nous proposons dans ce travail un modèle de différenciation comportementale, qui se décline en un outil dont les principaux apports sont d'être générique, non-intrusif, et de permettre une conception en dehors de l'agent. Le modèle s'articule selon trois axes. Tout d'abord, il décrit les comportements des agents par des normes. Celles-ci fournissent un profil comportemental à la conception, et un contrôle de la conformité à l'exécution. Ensuite, le processus de génération des comportements permet d'autoriser la création d'agents déviants ou en violation. Il influe pour cela sur le déterminisme du mécanisme. Enfin, les normes peuvent être inférées à partir de simulations enregistrées ou de situations réelles, afin d'analyser les résultats des expérimentations et d'automatiser la configuration du modèle.

Nous avons appliqué cet outil à la simulation de trafic dans SCANeR™, l'application développée et utilisée par le Centre Technique de Simulation de Renault pour ses simulateurs de conduite. Les modules logiciels développés au cours de la thèse introduisent dans le trafic des styles de conduite spécifiés sous forme de normes, par exemple des conducteurs prudents ou agressifs. Ils permettent ensuite de peupler l'environnement de manière automatisée, et de faire évoluer facilement les scénarios existants. Ces développements sont d'ores et déjà intégrés dans la version commerciale de l'application. Au delà de l'amélioration subjective du réalisme, les expérimentations réalisées démontrent les apports de l'outil sur la variété et la représentativité des comportements obtenus.

Mots clés : systèmes multi-agents, comportement, variété, conformité, cohérence, norme, génération, simulation, trafic routier

Abstract

Driving simulators are used by car manufacturers to develop driver aid systems, and to carry out experiments on ergonomics, design, or drivers' behavior : the objective is to improve the vehicles safety, and to reduce delays and development costs. In simulators, the driver is immersed in a simulated traffic, which realism is crucial for the validity of the results : the more realistic the environment is perceived, the more significant the drivers' reactions are.

In individual-centered approaches, like traffic simulation in our case, the agents' behavioral variety is an important realism criteria. The behaviors also have to be consistent : if aberrant situations appear, the simulation realism is strongly impacted. In this work, we adressed the issue of the simultaneous influence of these two elements. Furthermore, this work was led in an industrial context : it took place at Renault, in collaboration with the Computer Science Laboratory of Lille. This involved taking into account specific needs : the designed tools had to allow experts specifying various and consistent behaviors, and final users easily using them.

In this work, we propose a behavioral differentiation model, which provides the basis for a generic and non-intrusive tool allowing an out-of-the-agent design. The model involves three dimensions. First, it describes the agents' behaviors using norms. They provide a behavioral pattern during conception, and a compliance reference during execution. Then, the generation process of the behaviors allows the creation of deviant or violating agents, by influencing the determinism of the mechanism. Finally, the norms can be inferred from previous simulations records or real data, providing an analysis tool of the results and allowing automating the model configuration.

We applied the model to the traffic simulation in SCANeR™, the driving simulation software developed and used at the Technical Center for Simulation of Renault. The developments carried out during the thesis introduced driving styles in the traffic (e.g. cautious or aggressive drivers) specified using norms. The use of norms allows populating the environment easily and in an automated way. These developments are already included in the commercial version of the software. The behavioral realism of the traffic was improved, and the experimentations show how the model contributes to the variety and the representativeness of the produced behaviors.

Keywords : multi-agent systems, behavior, variety, compliance, consistency, norm, generation, simulation, road traffic

Remerciements

Je remercie l'ensemble des membres de mon jury de thèse : René Mandiau d'avoir bien voulu le présider, Marie-Pierre Gleizes et Olivier Boissier d'avoir accepté d'être rapporteurs et d'avoir contribué à l'amélioration de ce travail par leurs remarques et leurs observations, et enfin Viola Cavallo qui a accepté d'examiner mon travail.

Je remercie Andras Kemeny de m'avoir donné l'opportunité de travailler dans un contexte passionnant, en me proposant un sujet puis en co-dirigeant mes recherches. Ces trois années passées au Centre Technique de Simulation de Renault ont été extrêmement enrichissantes, en me permettant de combiner les exigences du travail de recherche et les contraintes liées aux problématiques industrielles.

Je remercie Philippe Mathieu qui a accepté de co-diriger ma thèse et qui a été d'une aide inestimable pour construire ce travail. Sa sensibilité aux problématiques industrielles et sa rigueur scientifique m'ont permis d'atteindre les objectifs de ce projet. Je le remercie de m'avoir consacré tant de son temps toujours précieux, de m'avoir poussé à me dépasser et de m'avoir guidé quand il le fallait. La réussite de ce travail lui doit beaucoup.

Je remercie Vincent Rouelle d'avoir partagé avec moi la volonté de voir les réalisations de ma thèse intégrées dans l'application finale, et de m'avoir poussé à ne jamais négliger cet objectif. Son expertise technique a été un support considérable tout au long de ces années. Je remercie également tous les membres de l'équipe de Support et Développement aux Simulateurs, Matthieu Vallet, Thierry Lefebvre, Mehmet Dagdelen, Hakim Mohellebi, ainsi que Kevin Eeckamn et Jérémie Thierry.

Je remercie Gilles Reymond, dont j'ai rejoint l'équipe pour les derniers mois de ma thèse, de m'avoir accordé sa confiance et laissé l'autonomie dont j'avais besoin pour terminer ce travail. Je remercie les membres de l'équipe, qui ont chacun à leur manière contribué à la réussite de cette thèse, Slim Azzi, Adrien Barthou, Emmanuelle Combe, Renaud Deborne, Nicolas Filliard et Benjamin Vailleau, et mes compagnons de vignette cette dernière année, Arnaud Mas et Thomas Denoual. Merci pour votre patience, nos discussions, et tous les cafés que nous aurions dû partager mais auxquels vous avez échappé pour cause de rédaction !

Je remercie tous mes collègues du Centre Technique de Simulation de Renault, Jérome Anty, Gilles Arnould, Sophie Bellanger, Jean-Charles Berlioux, Stéphanie Fortin, Pierre Legay, Olivier Legrand, Christian Morel, Jean-François Rivière, Frédéric Solnon, Stéphanie Thirard, David Toffin, Serge Vallée, Agnès Zimniak. Un merci tout particulier à Jean-Christophe Collinet et à Philippe Marillia, leurs concerts improvisés me manquent déjà.

Je remercie mon équipe de recherche, l'équipe Systèmes Multi-Agents et Comportements du Laboratoire d'Informatique fondamentale de Lille, pour leur accueil et leur aide lors de mes apparitions en pointillés. Merci à Bruno Beaufils, Jean-Paul Delahaye, Tony Dujardin, Patricia

Everaere, Maxime Morge, Sébastien Picault, Jean-Christophe Routier, Yann Secq, Irina Veryzhenko, aux désormais « anciens », Laetitia Bonte, Julien Derveeuw, Damien Devigne et Cédric Dinont, ainsi qu'à tous ceux, stagiaires ou étudiants, qui ont fait un passage dans l'équipe. Je tiens également à rendre un hommage tout particulier à Yoann Kubera et Antoine Nongaillard : messieurs, votre accueil chaleureux lors de mes arrivées à l'improviste étaient un plaisir sans cesse renouvelé, et je vous en remercie sincèrement. Merci enfin à tout le personnel du laboratoire, et en particulier à Maria Da Silva, pour mettre tant de soins à s'occuper de nos exigences souvent bien alambiquées.

Je remercie tous les collaborateurs de la société Oktal avec qui j'ai pu travailler au cours de ce projet, et qui ont contribué à sa réussite. Merci à Olivier Anselme, Gilles Gallée, Guillaume Millet et Nicolas Laurent d'avoir rendu cette collaboration possible, à Julien Hanower et Alexandre Troale pour leur support, dans tous les sens du terme. Merci également à ceux qui nous côtoient presque quotidiennement pour leur aide régulière, en particulier Thomas Nguyen-That, Julien Baessens et Yohan Chalumeaux, et avant eux Vincent Labussière.

Je remercie Biago Ciuffo d'avoir aidé un néophyte à pénétrer les arcanes de l'ingénierie de trafic, et d'avoir contribué à la validation de mon travail. Je remercie Erwin Boer pour les échanges que nous avons pu avoir sur mon travail et pour l'intérêt qu'il a porté à celui-ci.

Enfin, je ne saurai trop remercier mes parents, Béatrice et Didier, mes frères et sœurs, François, Pauline et Thomas, ainsi que Mamé et Hervé et tous mes amis, bibis, Lillois, Lyonnais et tous ceux que je ne peux énumérer ici. Merci pour votre présence et vos encouragements, vous êtes pour énormément dans la réussite de ce travail.

A tous, merci.

Table des matières

Résumé **i**

Abstract **iii**

Remerciements **v**

Introduction **1**

1 Contexte **5**

1.1 Simulation de trafic et simulateurs de conduite 5

1.1.1 Simulation de trafic routier 5

1.1.2 La simulation de trafic dans les simulateurs de conduite 8

1.1.3 Modélisation du trafic 12

1.1.4 Exemples d'applications existantes 17

1.1.5 Le cas des jeux vidéos 18

1.2 Approche centrée individu et systèmes multi-agents 19

1.2.1 Les systèmes multi-agents 19

1.2.2 Composition d'un système multi-agents 20

1.2.3 La simulation centrée individus 25

1.3 Travaux connexes 29

1.3.1 Gestion de configuration 29

1.3.2 Génération de paramétrage de modèles complexes 31

1.3.3 Variété et simulation de foules 32

1.3.4 Des personnalités virtuelles pour les conducteurs 34

2 Une description normative **37**

2.1 Les approches normatives 37

2.1.1 Introduction 37

2.1.2 Normes et systèmes multi-agents 39

2.1.3 Exemples d'applications 44

2.2 Description normative des comportements 46
2.2.1 Notre approche 47
2.2.2 Sémantique 48
2.2.3 Violations de la norme 55
2.3 Génération des comportements 60
2.3.1 Présentation 60
2.3.2 Application à la structure de données 63

3 Variété et conformité **71**
3.1 Gestion du système 71
3.1.1 Introduction 71
3.1.2 Les cartes auto-adaptatives ou réseaux de Kohonen 73
3.1.3 Applications 77
3.2 Variété et conformité 82
3.2.1 Variété 82
3.2.2 Conformité 84
3.3 Utilisation de l'outil 86
3.3.1 Différents modes d'utilisation 87
3.3.2 Utilisation de l'outil 87
3.4 Généricité 89
3.4.1 Variété dans la simulation de foules 89
3.4.2 Un créateur de créature 91

4 Application à la simulation de trafic **95**
4.1 La suite logicielle SCANeR™ 95
4.1.1 Présentation 95
4.1.2 La simulation de trafic dans SCANeR™ 99
4.1.3 Le modèle de trafic 104
4.1.4 Le cas des piétons 108
4.2 Application du modèle à SCANeR™ 109
4.2.1 Objectif 109
4.2.2 Implémentation 111
4.2.3 Intégration du modèle dans SCANeR™ 114
4.2.4 Synthèse 120

5 Évaluation, discussion et perspectives **125**
5.1 Démarche de validation 125
5.1.1 Calibration des modèles 125

5.1.2 Collecte de données . . . 126
5.1.3 Validation du trafic dans les simulateurs de conduite . . . 127
5.2 Évaluation de l'apport des normes dans SCANeR™ . . . 128
5.2.1 Description des simulations . . . 128
5.2.2 Apports du modèle . . . 130
5.2.3 Apprentissage des normes . . . 132
5.3 Évaluation du trafic dans SCANeR™ . . . 133
5.3.1 Objectif . . . 134
5.3.2 Congestion sur autoroute . . . 134
5.3.3 Influence des voies d'insertion . . . 136
5.3.4 Synthèse . . . 136
5.4 Discussion . . . 137
5.4.1 Discussion du modèle de différenciation comportementale . . . 137
5.4.2 Discussion de l'application à la simulation de trafic . . . 138
5.5 Perspectives . . . 139
5.5.1 Perspectives du modèle de différenciation comportementale . . . 139
5.5.2 Perspectives de l'application à la simulation de trafic . . . 140
5.5.3 Évolutions complémentaires du trafic . . . 142

Conclusion **145**

Bibliographie **149**

Table des figures **161**

Liste des tableaux **163**

Introduction

Contexte et motivations

Une des missions du Centre Technique de Simulation de Renault est de développer et d'exploiter des simulateurs de conduite d'étude et de recherche. Ces simulateurs sont utilisés pour des applications variées : mise au point de systèmes d'aide à la conduite, études d'ergonomie et de design, évaluation de l'éclairage des projecteurs, études du comportement humain dans des situations accidentogènes... Pour leur mise en œuvre, Renault développe depuis plus de quinze ans un ensemble de logiciels, SCANeR™, composé de modules dédiés aux différents aspects de la simulation : rendu visuel et sonore, calcul des trajectoires du véhicule, modification des conditions météorologiques, ou encore contrôle des caméras.

L'un de ces modules est destiné à la simulation du trafic ambiant, et répond à deux objectifs principaux. Le premier est la spécification de situations de conduite permettant de confronter différents conducteurs à des conditions similaires, afin d'obtenir des mesures comparables lors des expérimentations. Le trafic doit donc permettre la scénarisation du comportement des conducteurs simulés. Le second objectif est de soumettre le conducteur à une charge mentale correspondant à une situation de conduite réelle, en assurant une immersion aussi complète que possible. Cela permet d'améliorer la validité des résultats, qui repose en partie sur le ressenti des utilisateurs. L'objectif applicatif principal de la thèse, effectuée en convention Cifre chez Renault et en partenariat avec l'Équipe Systèmes Multi-Agents et Comportements du Laboratoire d'Informatique Fondamentale de Lille, est ainsi d'améliorer le réalisme comportemental du trafic pour renforcer encore cette immersion.

Le module trafic de SCANeR™ est construit suivant une architecture multi-agents : chaque véhicule autonome est un agent qui prend ses décisions en fonction de ses caractéristiques individuelles, et qui évolue dans l'environnement simulé en interagissant avec les autres agents et le conducteur réel. Dans les simulations centrées individus, la simulation de trafic dans notre cas, la variété du comportement des agents est un élément important pour le réalisme de la simulation. Leur cohérence joue également un rôle crucial : l'apparition de comportements aberrants nuit fortement au réalisme et à l'immersion des utilisateurs. Cependant, elle n'est pas prise en compte explicitement dans les approches existantes. Nous nous sommes donc intéressés dans ce travail à l'intégration de ces deux aspects, à travers la question suivante : augmenter la variété des comportements des agents, tout en préservant leur cohérence, améliore-t-il le réalisme ?

Au delà de cette question, le contexte industriel des travaux requiert la prise en compte de besoins spécifiques. Tout d'abord, l'approche doit pouvoir être mise en œuvre facilement par tout type d'utilisateur, et pas seulement par des experts. D'autre part, elle doit être flexible, afin de prendre en compte par exemple les spécificités culturelles liées à la localisation. Elle doit également être générique, afin d'être exploitable dans différents domaines applicatifs. Enfin, elle

doit permettre d'intégrer des comportements variés, pouvant être définis par des paramètres nombreux et interdépendants.

Contributions

Afin de répondre à ces questions, nous proposons un modèle de différenciation comportementale pour les simulations centrées individus, qui se décline en un outil répondant aux problématiques industrielles ciblées. Ce modèle s'articule en trois axes.

1. Le premier axe repose sur l'utilisation de normes pour définir des profils comportementaux pour les agents. La variété est assurée d'une part par la définition des normes, qui peuvent être aussi nombreuses que souhaité, et d'autre part par la possibilité de s'éloigner de cette norme par la violation. La détection des violations permet par ailleurs de contrôler la conformité des comportements.
2. Le second axe correspond à la création des comportements à partir de la description normative introduite. Les algorithmes proposés permettent de définir simplement l'équilibre entre variété et cohérence, en autorisant ou non la création de comportements déviants. Découplés du domaine d'utilisation, ils introduisent en outre la généricité et la flexibilité désirées. Leurs différents niveaux de configuration rendent leur utilisation aisée pour des non spécialistes et riche pour des experts.
3. Enfin, le troisième axe s'intéresse à la configuration du modèle et au contrôle des comportements. À partir de l'observation d'une simulation et de techniques d'apprentissage automatique, les normes utilisées en pratique par les agents sont inférées. La configuration ainsi obtenue est utilisée pour spécifier, analyser ou reproduire des simulations.

Ces trois axes nous permettent de spécifier la variété du comportement des agents, et de contrôler leur cohérence à tous les stades de la simulation. Ils répondent ainsi aux besoins de deux populations différentes : l'utilisateur final de la simulation, qui s'intéresse aux apports relatifs au réalisme ; et le concepteur ou le scénariste, qui recherche un outil lui permettant de fournir aisément ce réalisme et de s'adapter aux différents utilisateurs.

D'une manière générale, le cadre d'utilisation de cette approche est la simulation centrée individus. Cependant, elle apporte une valeur ajoutée plus importante lorsqu'elle est appliquée sur des plateformes de simulation existantes. En effet, l'ajout de nouvelles fonctionnalités soulève des problématiques importantes liées à la non régression, la rétro-compatibilité et les impacts sur l'existant. Par ailleurs, les utilisateurs n'ont pas nécessairement la possibilité de modifier le cœur de la simulation, typiquement parce qu'ils n'y ont pas accès, comme dans le cas d'outils provenant d'éditeurs extérieurs. Notre approche présente alors un avantage important : elle permet d'introduire les apports relatifs à la variété et la cohérence de manière non-intrusive, en se basant uniquement sur le paramétrage externe de la simulation. Si l'utilisateur peut intervenir directement sur le moteur de simulation, notre approche permet d'intégrer les apports de manière modulaire, et de limiter les risques liés au développement. Cependant, elle n'offre pas autant de flexibilité qu'une intervention directe sur le paramétrage interne de l'application.

Par ailleurs, les paramètres disponibles dans les simulations sont souvent interdépendants et peu documentés. Leur configuration nécessite alors l'intervention d'experts fonctionnels, possédant des connaissances acquises empiriquement. L'utilisation de notre approche permet de capitaliser cette expertise, et de la mettre à profit pour l'ensemble des utilisateurs : les experts

réalisent la configuration des normes, qui peuvent alors être exploitées simplement par les autres utilisateurs.

Nous avons appliqué cet outil à la simulation de trafic dans SCANeR™. Trois modules distincts, développés au cours de la thèse, intègrent nos propositions. Ils ont permis d'introduire dans la simulation différents styles de conduite pour les conducteurs virtuels (agressifs, prudents...) : ceux-ci représentent un élément important pour l'immersion des conducteurs réels. L'obtention des styles de conduite nécessite de contraindre simultanément tout ou partie des paramètres comportementaux des conducteurs : un conducteur agressif conduit vite, mais il utilise également de faibles marges de sécurité et a tendance à accélérer et freiner brusquement... Par ailleurs, chaque conducteur agressif doit se comporter de manière différente, le conducteur réel risquant sinon d'avoir l'impression d'être perpétuellement face au même véhicule. Le modèle présenté propose une solution à ces problématiques, et a ainsi été utilisé pour reproduire les styles de conduite et introduire une variété maîtrisée dans la simulation. Notons que ces réalisations sont d'ores et déjà incluses dans la version commerciale de SCANeR™.

Les besoins des différents types d'utilisateurs sont ainsi couverts. D'une part, les utilisateurs finaux qui conduisent le simulateur sont mieux immergés dans la simulation, car ils sont confrontés à des comportements plus variés et plus cohérents. D'autre part, les scénaristes ont à leur disposition des comportements pré-conçus qui leur permettent de construire des simulations plus complexes et plus réalistes, tout en diminuant leur charge de travail. Par ailleurs, la flexibilité de l'approche permet de l'adapter facilement à différents contextes : un conducteur agressif ne se comporte pas de la même manière en France et en Italie... La simplicité d'utilisation permet en outre de créer de nombreux comportements, sans risquer de produire des situations aberrantes par méconnaissance des effets des paramètres comportementaux. Enfin, la généricité permettra d'utiliser la même approche pour définir le comportement des piétons dans la simulation.

Nous avons enfin évalué les apports des outils proposés dans le cadre de la simulation de trafic, à travers différentes expérimentations. Tout d'abord, des situations classiques de trafic peuvent désormais être reproduites, comme les phénomènes de ralentissement en fonction du flux de trafic, ainsi que l'influence des voies d'insertion sur autoroute. Par ailleurs, le comportement simulé des conducteurs correspond à la norme attribuée, et la présence de différents styles de conduite rend le trafic plus dynamique. Enfin, la calibration de la configuration utilisée pour les normes a permis de constituer des ensembles de normes adaptés à différentes situations de conduite et à différents types d'utilisateurs, suivant l'équilibre désiré entre reproductibilité et variété.

Organisation du document

Le premier chapitre est consacré à la description du contexte des travaux. Nous présentons tout d'abord le cadre applicatif, la simulation de trafic routier, et ses particularités dans le contexte de la simulation de conduite. Nous nous intéressons ensuite aux simulations centrées individus et aux systèmes multi-agents. La dernière partie de ce chapitre est consacrée aux travaux relatifs à l'introduction et à la maîtrise de la variété dans les systèmes, qui sont connexes à notre approche.

Le second chapitre introduit les deux premiers axes de notre modèle, concernant la description et la génération des comportements. Après avoir présenté la notion de norme et ses applications, nous présentons notre proposition, basée sur l'utilisation des normes pour décrire

les comportements. La méthode de génération proposée permet de créer les comportements spécifiés tout en contrôlant l'équilibre entre variété et conformité, à travers l'utilisation de la déviance et des violations. Nous présentons ici les différents algorithmes proposés.

Le chapitre trois décrit tout d'abord les outils de contrôle qui permettent d'inférer les normes de la simulation, de détecter la norme d'appartenance des agents, et de configurer automatiquement les normes. Nous précisons ensuite comment l'utilisation des trois axes du modèle permet de répondre à notre question : créer de la variété et contrôler simultanément la cohérence. Nous décrivons ensuite l'outil et ses modes d'utilisation, avant d'illustrer la généricité de l'approche en décrivant sa mise en œuvre sur plusieurs exemples.

Le quatrième chapitre présente l'application de l'approche proposée à la simulation de trafic dans SCANeR™. Nous décrivons tout d'abord la suite logicielle, en détaillant le fonctionnement du modèle de trafic. Nous décrivons ensuite l'implémentation du modèle de différenciation comportementale dans l'application, et motivons les choix qui ont été réalisés. Nous présentons enfin les modules développés au cours de la thèse, ainsi que leurs apports respectifs.

Enfin, le cinquième chapitre est consacré à l'évaluation de l'approche. Nous nous intéressons à la validation des apports dans le cadre de notre application et présentons les différentes expérimentations réalisées : reproduction de phénomènes de trafic et évaluation des normes. Nous montrons ainsi que de nombreux aspects du comportement du trafic ont été améliorés. Nous discutons ensuite notre approche, ainsi que la manière dont elle a été appliquée. Nous terminons enfin en présentant les perspectives de nos travaux.

Chapitre 1

Contexte

Ce premier chapitre présente le contexte des travaux menés au cours de la thèse. Nous introduisons tout d'abord le cadre de notre application principale, la simulation de trafic routier. Objectifs, méthodes et outils utilisés dans ce domaine sont présentés, ainsi que les particularités liées à la simulation de conduite. La partie trafic de l'application de simulation de conduite SCANeR™ est bâtie suivant une architecture multi-agents. Dans un second temps, nous nous intéressons donc à l'approche centrée individus. Nous rappelons les principales propriétés et caractéristiques des systèmes multi-agents, avant d'illustrer leur utilisation par des exemples d'application. Enfin, la dernière partie est dédiée à la présentation de travaux connexes à notre problématique de recherche. Nous présentons les principales approches mises en œuvre dans les domaines s'intéressant à l'obtention de comportements variés et cohérents.

1.1 Simulation de trafic et simulateurs de conduite

Cette première partie a pour objet de présenter le cadre applicatif de notre étude : la simulation de trafic dans les simulateurs de conduite. Nous nous intéressons tout d'abord à la simulation de trafic routier, avant de préciser les problématiques liées au contexte des simulateurs de conduite. Enfin, nous présentons les approches utilisées en modélisation du trafic, avant de terminer par un aperçu du cas des jeux vidéos.

1.1.1 Simulation de trafic routier

Dans cette section, nous abordons le contexte de la simulation de trafic, avant de nous intéresser aux différentes approches utilisées et aux possibilités qu'elles offrent.

1.1.1.1 Introduction

L'étude des mécanismes du trafic et de ses acteurs, les conducteurs, est un domaine largement étudié [Salvucci et al., 2001, Elvik, 2004, Summala, 2005, Bazzan, 2005]. Ces études ne répondent pas toutes aux mêmes besoins, et sont abordées suivant différents axes. Parmi ceux-ci figurent l'ingénierie du trafic et la psychologie de la conduite, que nous introduisons rapidement dans cette section.

L'ingénierie du trafic a pour objectifs de répondre à des problématiques de planification des transports (prévision du trafic, comme par exemple Bison futé [Danech-Pajouh, 2003]) ou de conception de nouveaux aménagements routiers [Hourdakis & Michalopoulos, 2002]. Pour répondre à ces problématiques, la modélisation des caractéristiques du trafic est un des axes de recherche fondamentaux, qui vise à reproduire et prédire son fonctionnement [Cohen, 1993].

Dans une perspective différente, la psychologie de la conduite vise à comprendre et expliquer le comportement des conducteurs [Letirand & Delhomme, 2005]. Identifier les facteurs conduisant à l'apparition de comportements à risque permet en effet de développer des techniques de prévention. Ces études mettent souvent en jeu l'observation du comportement de conducteurs réels, confrontés à des situations particulières comme la conduite dans le brouillard, et dont on analyse le comportement et les réactions [Saad, 1992, Priez et al., 1998, Caro et al., 2007]. Elles passent également par la conception de simulateurs dédiés à la simulation des conducteurs, visant à reproduire les processus cognitifs mis en œuvre [Salvucci, 2001, Bellet et al., 2005].

Dans ce contexte, la simulation du trafic trouve une place importante. Elle permet par exemple d'évaluer de nouvelles solutions de régulation du trafic, comme des stratégies de signalisation innovantes [Saunier, 2005]. Elle offre également un moyen de développer des systèmes en rupture, comme la prise en compte de la présence de véhicules robotisés dans un trafic réel [Dresner & Stone, 2007, Dresner & Stone, 2008]. Dans la plupart des cas, ces études ne peuvent pas être menées dans des situations réelles, principalement pour des contraintes de coût et de sécurité. La simulation offre une solution peu coûteuse et sûre pour mener ce type d'expérimentation, ainsi que la possibilité de tester de multiples possibilités de manière reproductible.

1.1.1.2 Les différentes approches

En fonction des besoins, différentes approches peuvent être utilisées pour simuler le trafic. On distingue les approches macroscopiques, mésoscopiques et microscopiques, en fonction du niveau de description utilisé.

Approches macroscopiques Ce type d'approche utilise un niveau de détail faible. Les véhicules ne sont pas représentés individuellement, et leurs interactions ne sont prises en compte que dans les équations décrivant le système, au niveau global. Des matrices « origine – destination » décrivent les entrées – sorties du réseau : pour chaque pas de temps, un nombre de véhicules est associé à chaque itinéraire possible. Des modèles d'assignement statique ou dynamique permettent ensuite de calculer l'occupation du réseau [Taylor, 2003] et d'obtenir des indicateurs (Fig. 1.1), tels que le flux sur les différentes routes ou des estimations de durées de parcours.

Certains modèles macroscopiques [Cayford et al., 1997, Haj-Salem et al., 1998], dits continus, assimilent le trafic à un fluide . Des relations flux-vitesse ainsi que des équations de conservation sont alors utilisées pour calculer son évolution. La conservation de la masse devient la conservation du nombre de véhicules, et la pression est remplacée par une fonction empirique décrivant le comportement des conducteurs. Ces modèles sont principalement destinés à la simulation de réseaux fortement chargés.

Approches mésoscopiques Les modèles mésoscopiques utilisent un niveau de détail plus fin que les modèles macroscopiques. L'idée sous-jacente à cette approche est qu'effectuer des calculs sur des quantités agrégées à un niveau macroscopique ne fournit pas une description

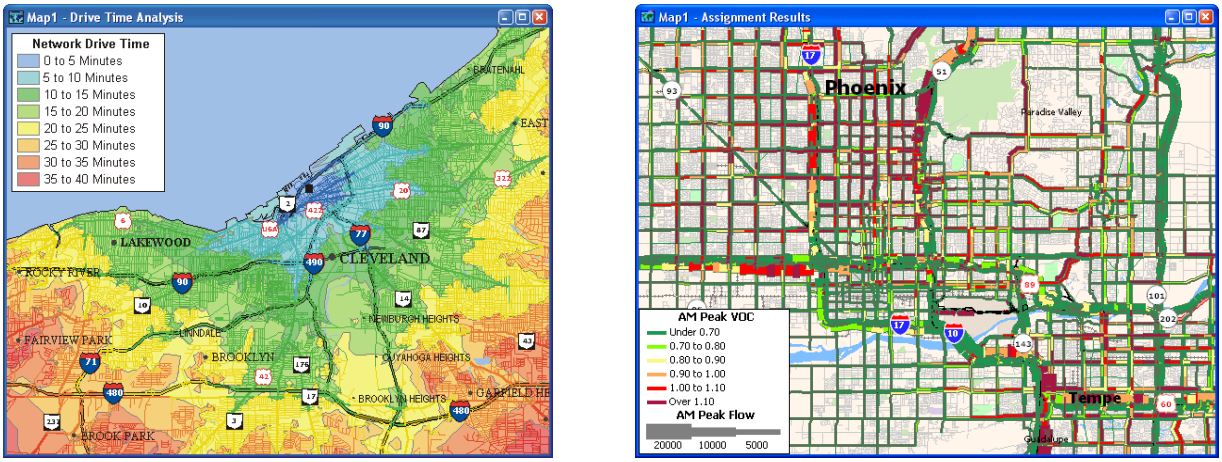


FIGURE 1.1 – Exemple de sortie d'un outil de simulation de trafic macroscopique (d'après [Caliper, 2009]) : à gauche les temps de parcours sur le réseau, à droite la répartition des véhicules résultant d'un calcul d'assignement.

suffisamment détaillée des évolutions du trafic, mais que suivre et gérer les véhicules de manière individuelle est trop coûteux en temps de calcul. Pour pallier à ce problème, ces modèles proposent un compromis en composant approche macroscopique et approche microscopique : ils représentent le trafic par des « paquets » de véhicules regroupés en fonction de certaines caractéristiques [Jayakrishnan et al., 1994]. Différentes classes de conducteurs peuvent par exemple être introduites, ou différents types de véhicules.

Comme pour les modèles macroscopiques, les interactions entre les véhicules ne sont prises en compte qu'à travers des équations globales, tout comme les relations entre les véhicules et l'environnement.

Approches microscopiques Dans les approches microscopiques, les véhicules sont considérés uniquement de manière individuelle, et les interactions entre véhicules et avec l'environnement sont prises en compte au niveau local. Certains modèles sont basés sur l'utilisation d'automates cellulaires [Nagel & Schreckenberg, 1992, Ruskin & Wang, 2002]. Ce type de modèle repose sur une discrétisation de l'espace en cellules, le déplacement des mobiles et l'occupation des cellules étant déterminés par les règles de l'automate. La taille d'une cellule correspond alors à celle d'un véhicule, par exemple une taille fixe de 7.5 m représentant une estimation de l'espace moyen occupé par un véhicule dans un embouteillage [Esser & Schreckenberg, 1997]. Malgré les limitations dans les capacités de description de l'environnement, les modèles basés sur des automates cellulaires permettent d'étudier des systèmes complexes, comme la gestion des intersections [Ruskin & Wang, 2002].

De nombreux outils de simulation de trafic microscopique sont disponibles, notamment VISSIM [Fellendorf & Vortisch, 2001, PTV, 2004], Paramics [Wylie et al., 1994, Duncan, 1995], NGSIM [Alexiadis et al., 2004], AIMSUN [Barcelò et al., 1999, Barcelò & Casas, 2002], ou encore MITSIMLab [Yang & Koutsopoulos, 1996, Toledo et al., 2003]. Ces solutions diffèrent par le modèle de simulation de trafic qu'elles utilisent. La modélisation peut ainsi être mathématique, comportementale, hybride...

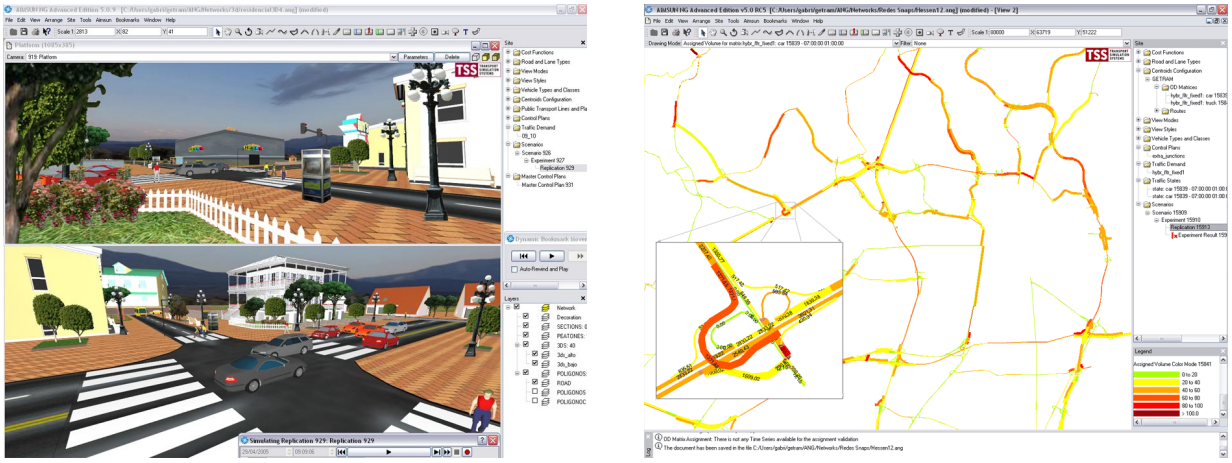


FIGURE 1.2 – Exemple d'un simulateur microscopique de trafic (ici AIMSUN, d'après [TSS, 2009]). À gauche, la représentation 3D du trafic en zone urbaine. À droite, un aperçu du couplage microscopique et macroscopique, avec une vue détaillée d'une intersection associée à une vue agrégée de l'assignement sur le réseau (en arrière plan).

Synthèse Chacune des approches présentées ci-dessus répond à des besoins différents. Une modélisation macroscopique ou mésoscopique permet d'extraire des valeurs statistiques du trafic simulé, ce qui convient pour l'étude du dimensionnement du réseau routier ou des congestions qui peuvent y survenir. Elles permettent de simuler des réseaux de taille importante, impliquant des volumes conséquents de trafic, tout en limitant la charge de calcul. Les interactions entre les véhicules et avec l'environnement restent cependant limitées, bien qu'elles puissent être intégrées en partie dans les équations décrivant le fonctionnement du système.

Les études nécessitant la prise en compte du comportement individuel des conducteurs, comme dans le cas de la sécurité ou des comportements à risque, imposent donc l'utilisation d'un modèle microscopique. Elles nécessitent en effet de prendre en compte de manière fine les interactions entre les véhicules. Plus coûteuses en calcul, elles offrent cependant une précision plus élevée au niveau des comportements individuels.

Notons que certains outils, comme AIMSUN (Fig. 1.2), intègrent ces différentes dimensions dans une approche verticale, afin de fournir des niveaux de description correspondant aux différents types d'études menés par les utilisateurs.

1.1.2 La simulation de trafic dans les simulateurs de conduite

Après cet aperçu des approches utilisées pour la simulation de trafic, intéressons nous aux contraintes spécifiques qu'impose le cadre des simulateurs de conduite.

1.1.2.1 Introduction aux simulateurs de conduite

Les simulateurs de conduite sont utilisés dans l'industrie automobile pour différents usages. Le premier est la mise au point de systèmes d'aide à la conduite comme l'ABS [1] ou l'ESP [2]. Le

1. Antiblockiersystem : système de freinage anti-blocage.
2. Electronic Stability Program : système de sécurité active destiné à améliorer le contrôle de trajectoire du véhicule.

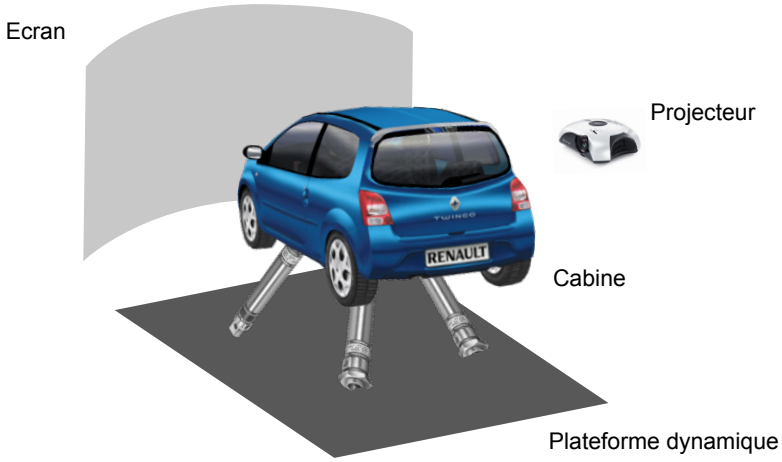


FIGURE 1.3 – Les éléments fondamentaux composant un simulateur de conduite : un système de visualisation (ici un projecteur et un écran), une cabine, et éventuellement une plateforme dynamique.

second est centré sur les problématiques de design et de conception, les simulateurs permettant de réaliser des tests avant la fabrication de prototypes physiques, en utilisant directement les modèles numériques des véhicules. En recherche, les usages sont similaires, mais les objectifs seront orientés vers le développement de nouveaux systèmes d'aide à la conduite plutôt que vers leur mise au point, ou vers l'étude de l'influence de systèmes existants sur le comportement conducteur. L'objectif final est de permettre la conception de véhicules plus sûrs, et de réduire le coût et le délai des études. La dimension facteurs humains, s'intéressant à comprendre l'interaction entre le conducteur et le véhicule, est également présente. Notons enfin que d'autres usages, comme la formation des conducteurs, se développent aujourd'hui fortement.

Afin d'immerger le conducteur dans la simulation, un simulateur de conduite est composé de différents éléments (Fig. 1.3) :

- un poste de conduite, qui peut être un cockpit de véhicule réel ou une reconstitution intégrant un volant, un pédalier et un levier de vitesse. Ces éléments peuvent être équipés de systèmes à retour d'effort et de vibrations, afin d'améliorer l'immersion des conducteurs,
- un système de visualisation, qui affiche l'environnement simulé en fonction de la position du conducteur. Ce système peut être constitué d'un casque de réalité virtuelle, ou d'écrans offrant un champs de vision plus ou moins large (généralement de 60° à 330°),
- enfin, certains simulateurs, dits dynamiques, sont montés sur des plate-formes mobiles permettant de restituer une partie des accélérations longitudinales et latérales du véhicule. En fonction de l'étude réalisée, la restitution précise des sensations de conduite peut en effet être très importante : pour des études de confort ou de tenue de route, le ressenti des accélérations est primordial.

1.1.2.2 Les contraintes sur le trafic

Contrairement aux simulateurs de trafic orientés ingénierie du trafic, l'objectif des simulateurs de conduite est de placer un conducteur réel dans la simulation, et de lui permettre d'interagir avec les véhicules autonomes gérés par l'application. Cela implique différentes contraintes sur le modèle de trafic utilisé. Tout d'abord, chaque véhicule doit être en mesure de réagir aux actions

du conducteur réel, et doit donc être représenté indépendamment, avec ses caractéristiques et son comportement propre. Les modèles macroscopiques ou mésoscopiques ne conviennent pas : la gestion des interactions entre les véhicules ainsi qu'avec l'environnement est trop limitée. De la même façon, les automates cellulaires ne sont pas adaptés, la représentation discrète de l'espace ne s'accordant pas avec l'espace continu dans lequel un conducteur évolue. L'utilisation d'un modèle microscopique intégrant une représentation fine des interactions et de l'environnement est nécessaire.

Par ailleurs, le conducteur doit autant que possible avoir l'impression d'être en situation réelle. L'immersion influe en effet directement sur la validité des études qui peuvent être menées à l'aide des simulateurs de conduite. Elle peut par exemple passer par le rendu des perceptions visuo-vestibulaires en utilisant des plateformes mobiles ou l'amélioration du rendu graphique de la simulation [Reymond et al., 2001, Dagdelen et al., 2004, Lefebvre et al., 2006]. Le réalisme comportemental des véhicules du trafic s'inscrit dans ce cadre : l'objectif est que le conducteur ait l'impression d'évoluer dans un trafic le plus réaliste possible, et soit soumis à une charge mentale cohérente avec une situation réelle. Les questions de validité statistique du trafic, centrales dans les problématiques de simulation en ingénierie du trafic, sont ici secondaires [3].

Une autre propriété importante et spécifique aux simulateurs de conduite est la nécessité de pouvoir confronter les conducteurs à des situations préparées à l'avance. Par exemple, lors de l'évaluation de systèmes d'aide à la conduite, plusieurs conducteurs sont confrontés à une situation similaire, afin d'étudier leur réaction [Deborne et al., 2008]. Pour ce faire, le concepteur prépare le déclenchement de comportements spécifiques pour certains véhicules du trafic, ainsi que les circonstances dans lesquelles ils vont apparaître [Olstam & Espié, 2007]. Le modèle de trafic doit donc intégrer la possibilité de scénariser les véhicules, c'est-à-dire que certaines actions puissent être imposées, et non décidées par le modèle. Cela sous-entend une problématique complémentaire : les autres véhicules doivent être en mesure de réagir à ces évènements imprévus de manière réaliste, afin de préserver l'immersion.

La Figure 1.4 présente l'exemple d'une situation de trafic scénarisée. Sur l'image de gauche est présentée la vue depuis le poste de conduite, telle que l'observe le conducteur réel. Sur l'image de droite, une vue aérienne rend visible l'ensemble de la situation, le véhicule conduit étant au premier plan. La situation est déclenchée à l'approche de celui-ci : le véhicule orange (à l'arrière plan) était à l'arrêt, et déboîte brusquement alors qu'il est en train d'être dépassé par le conducteur réel. Ce déboîtement est causé par des travaux, visibles sur la vue aérienne, mais invisibles pour le conducteur. Cette situation permet de placer le conducteur en situation critique, lui imposant une réaction d'urgence. Cela permet par exemple de mesurer son temps de réaction en fonction de la présence ou non d'un système d'aide à la conduite.

1.1.2.3 La gestion du temps

Dans le cas des simulateurs de conduite, les différents modules utilisés ne travaillent pas nécessairement à des fréquences égales. Par exemple, le calcul du visuel est réalisé à une fréquence de 60 Hertz : ce taux de rafraichissement correspond aux capacités du matériel de visualisation, et est suffisant vis-à-vis des capacités humaines. Cependant, pour d'autres modules, la fréquence doit atteindre 1000 Hz afin de correspondre aux capacités de perception d'un conducteur. C'est

3. Notons cependant que la reproduction de situations réelles avec des simulations ne comportant que des véhicules autonomes permet de démontrer la validité du modèle de trafic utilisé (cf. Chapitre 5).



FIGURE 1.4 – Exemple d'une situation de trafic scénarisée, avec sur l'image de gauche la vue depuis le poste de conduite, et sur celle de droite une vue aérienne rendant visible le véhicule conduit (au premier plan).

par exemple le cas pour la gestion des capteurs et des actionneurs d'une plateforme mobile haute performance.

Notons que l'obtention de fréquences de calcul aussi élevées peut imposer l'utilisation de systèmes conçus pour faire du temps réel. En effet, sans optimisations particulières, un système d'exploitation comme Linux, non conçu pour la gestion d'applications temps réel, présente des temps de latence de l'ordre de 10 à 20 ms [Bruyninckx, 2006]. Il n'est donc pas envisageable de l'utiliser pour obtenir des fréquences de 500 à 1000 Hz, soit des temps de calcul de l'ordre de la milliseconde. La notion de temps réel doit cependant être distinguée de celle de rapidité de traitement. Les systèmes informatiques temps réel ont pour particularité de prendre en compte des contraintes temporelles dont le respect est aussi important que les résultats attendus : le système doit non seulement fournir des résultats exacts, mais les fournir dans des délais imposés [Ficheux, 2005]. Le temps réel ne se préoccupe pas de la vitesse d'exécution du code, dans la limite où les contraintes temporelles sont respectées : un système temps réel n'est pas forcément plus rapide qu'un système à temps partagé.

Pour le trafic, l'objectif est d'obtenir une fréquence qui offre à l'utilisateur un confort de conduite correspondant, dans l'idéal, à la réalité. La perte d'un pas de temps au niveau du calcul n'a pas d'impact bloquant sur l'ensemble de la simulation, mais la gestion du temps a une influence sur l'immersion des conducteurs dans le simulateur (trafic saccadé, véhicules qui « sautent » d'un point à un autre...). Notre objectif se situe donc dans l'obtention de fréquences élevées, qui passe par un souci constant des performances lors des développements.

1.1.2.4 Le déterminisme

Comme nous l'avons vu, la préparation de scénarios de trafic est un des éléments clé pour la réalisation d'expérimentations utilisant des simulateurs de conduite. Cependant, la présence d'un conducteur réel dans la simulation introduit nécessairement des éléments incontrôlables. Garantir le déterminisme de la simulation n'est donc pas un critère pertinent : le résultat final ne pourra pas être reproduit de manière exacte, du fait des actions du conducteur, différentes à chaque essai. On peut s'efforcer de préparer la situation souhaitée avec précision, en prenant en

compte les adaptations aux différents comportements possibles du conducteur, mais le résultat ne pourra pas être certifié.

Cependant, certains usages nécessitent de prendre en compte la problématique du déterminisme. Tout d'abord, il est intéressant de pouvoir rejouer les scénarios de conduite. Dans ce cas, les données provenant du conducteur sont enregistrées (angle volant, pédales...), et peuvent être réintroduites en entrée de la simulation. Cela permet par exemple d'analyser a posteriori le comportement de l'utilisateur. Dans ce cadre, il est important que la simulation soit déterministe : la simulation rejouée doit être identique à la situation initiale.

Par ailleurs, le déterminisme présente également un intérêt pour la reproduction de situations réelles de trafic. Dans ce cas, la reproductibilité des scénarios est importante, et est attendue par l'utilisateur final. Celui-ci souhaite avoir des résultats identiques d'une simulation à l'autre, en partant des mêmes données initiales. Cela permet d'attester de la validité du calcul. Enfin, dans le cadre de la calibration des modèles, une simulation déterministe permet de garantir que seuls les paramètres modifiés volontairement influent sur le résultat de la simulation, et pas d'éventuelles variations introduites par des processus stochastiques.

Notons cependant que dans certains cas, un certain degré de stochasticité est acceptable, voire souhaitable. L'utilisation de variables aléatoires permet en effet de créer une dispersion contrôlée autour des valeurs par défaut spécifiées dans la simulation. Typiquement, on peut penser à la vitesse que les conducteurs autonomes souhaitent atteindre : s'ils utilisent tous la même valeur, cela peut conduire à un manque de réalisme et à des biais dans les résultats de la simulation. Il peut donc être intéressant d'intégrer un degré de variation probabiliste dans ce type de situation. Attention cependant que de tels processus rendent la simulation moins prédictible. Il est nécessaire de trouver un compromis entre le déterminisme requis pour l'expérimentation et le réalisme que permet d'atteindre l'utilisation de facteurs aléatoires dans la simulation [Olstam & Espié, 2007].

1.1.3 Modélisation du trafic

Intéressons nous maintenant aux différents éléments sur lesquels repose la simulation du trafic : la représentation de l'environnement, le modèle de trafic, et les caractéristiques du conducteur prises en compte.

1.1.3.1 Représentation de l'environnement routier

Afin d'évoluer dans le monde simulé, les véhicules autonomes doivent percevoir l'environnement qui les entoure. Pour cela, ils peuvent utiliser différentes techniques.

Tout d'abord, le conducteur simulé peut analyser directement la scène visuelle virtuelle, comme un humain le ferait. Il en extrait les informations dont il a besoin, et les interpréte en temps réel. Ces techniques sont utilisées en vision artificielle, par exemple dans Animat [Terzopoulos & Rabie, 1997], et permettent d'étudier comment les êtres vivants analysent leur environnement et en retirent des informations . Cependant ces méthodes mettent en œuvre des algorithmes complexes de vision et requièrent d'importantes capacités de calcul. Elles ne sont pas adaptées à notre contexte, où l'objectif ne se focalise pas sur les modalités de perception, mais sur la restitution d'un comportement réaliste.

Afin d'éviter ces problèmes de performances, les environnements de simulation de conduite sont généralement construits sous forme de bases de données multi-couches [Carles & Espié, 1999,



FIGURE 1.5 – À gauche la représentation graphique d'une base de données. À droite, la même zone en vue simplifiée, incluant en surimpression la représentation des éléments logiques de l'environnement, qui seront visibles par les véhicules autonomes : voies, panneaux, présence d'une intersection...

Willemsen et al., 2003]. Une des couche contient une description logique de l'environnement, sur laquelle se base le modèle de décision des véhicules. Cette description logique est couplée à une couche contenant une représentation graphique, qui est présentée au conducteur dans le simulateur (Fig. 1.5). La description logique présente l'avantage de pouvoir fournir aux véhicules autonomes une vue simplifiée de l'environnement, incluant en particulier la signalisation (lignes de marquage, panneaux...) et la structure du réseau (nombre de files, largeur des voies...). Cela permet de limiter la charge de calcul liée à l'analyse du réseau. De plus, s'appuyer sur une description de ce type permet de contrôler la perception que les agents ont de l'environnement : ils ne pourront percevoir que les éléments définis par le concepteur. Enfin, il est possible de fournir aux véhicules une description sémantique de l'environnement, qui leur permet de réaliser des raisonnement sur des éléments de plus haut niveau, comme le type d'intersection duquel ils approchent [Donikian, 1997].

Les véhicules s'appuient sur cette description logique afin de construire une représentation locale de l'environnement les entourant. Ils utilisent également cette structure hiérarchique, organisée suivant le réseau routier, afin de de déterminer l'ensemble des véhicules avec lesquels ils sont susceptibles d'entrer en interaction : approche d'un même carrefour, voies adjacentes...

1.1.3.2 Les modèles de trafic

Une fois qu'il possède une représentation de son environnement, le conducteur virtuel peut décider de quelle façon il va se déplacer. En situation de conduite, il a le choix entre différentes actions : continuer tout droit, changer de voie, dépasser, s'arrêter... Cette prise de décision est réalisée de façon variée suivant les modèles.

Approche par sous-modèles Une approche classique consiste à utiliser des sous-modèles des tâches de conduite, et de les appliquer en fonction du contexte. Le conducteur dispose d'un modèle pour le suivi de véhicule, d'un autre pour le changement de voie, d'un troisième pour s'engager en carrefour...

Par exemple, dans le cas du changement de voie, le processus de décision peut se baser sur la nécessité d'un changement (pour atteindre une sortie, pour doubler un véhicule trop lent), le désir d'un changement (va-t-on y gagner si l'on change de voie ?), et enfin sa faisabilité (existe-t-il un écart suffisant pour effectuer le changement ?) [Gipps, 1986]. Les paramètres utilisés prennent en compte la distance à la prochaine intersection : plus on s'en rapproche, plus la nécessité de se placer correctement pour sortir est important. Cela permet d'intégrer dans le modèle les contraintes auxquelles est soumis un conducteur réel.

Un autre de ces sous-modèles de tâche de conduite concerne le suivi de véhicule. Il décrit le comportement du conducteur lorsqu'il suit le véhicule le précédant dans la même voie. Ce type de modèle est fondamental en simulation de trafic, et a été largement étudié. Différentes classes peuvent être distinguées :

- les modèles Gazis-Herman-Rothery, qui considèrent que l'accélération du véhicule est proportionnelle à la vitesse du véhicule suivi, la différence de vitesse entre les véhicules, et la distance inter-véhiculaire [Chandler et al., 1958, Gazis et al., 1961],
- les modèles basés sur la distance de sécurité, qui reposent sur l'hypothèse que le suiveur garde toujours une distance de sécurité avec le véhicule précédant [Gipps, 1981],
- et les modèles de suivi de véhicule dits « psycho-physiques », qui utilisent des limites de déclenchement afin de lisser le comportement du véhicule [Wiedemann, 1974, Fritzsche, 1994] : tant que la différence (de vitesse, par exemple) n'est pas trop importante, il n'y a pas de réaction.

Notons que [Brackstone & McDonald, 1999a] propose une revue très complète de ces modèles.

Approche par automates Une autre méthode de prise de décision est basée sur l'utilisation d'automates [Kemeny, 1993, Cremer et al., 1995]. Dans ce cas, les états de l'automate représentent les différentes actions qui peuvent être réalisées : par exemple rouler sur une zone libre, suivre un véhicule, rester dans une voie, tourner à une intersection, et dépasser sur une autoroute [Cremer & Joseph Kearney, 1996]. Le choix de l'état est basé sur des fonctions de scoring, d'éventuels conflits étant résolus par une priorisation a priori des actions. Par exemple, doubler peut être considéré comme systématiquement plus intéressant que de rester sur la même voie. Les automates permettent de décrire de manière simple le fonctionnement du système, mais présentent des limites dans le cas de comportements complexes. Un conducteur peut en effet être confronté à des actions mettant en jeu un équilibre entre plusieurs actions, ce qui n'est pas envisageable avec une structure d'automate.

Approche par minimisation des interactions D'autres approches s'appuient sur des modes de décision visant à se rapprocher de ceux mis en œuvre par les conducteurs réels. Par exemple, la méthode utilisée dans l'application de simulation de conduite Archisim est basée sur la minimisation des interactions [Espié et al., 1994]. Élaborée à partir d'études menées par des psychologues de la conduite, elle s'appuie sur l'hypothèse qu'un conducteur prend ses décisions sur la base des conditions de trafic dans différentes zones autour du véhicule (Fig. 1.6). À partir de la distribution des vitesses et de la vitesse minimale dans la zone, la durée de l'interaction est évaluée. En y ajoutant la distance temporelle estimée de l'interaction, le modèle décide s'il est nécessaire de réagir. Par exemple, si l'interaction est estimée de courte durée, aucune adaptation n'a lieu : cela évite qu'un véhicule change sans cesse de file en fonction des conditions très locales de circulation. Ce mécanisme permet aux conducteurs de ne pas réagir instantanément aux variations de l'environnement, et d'anticiper certains changements en intégrant la durée des interactions dans leur processus de décision.

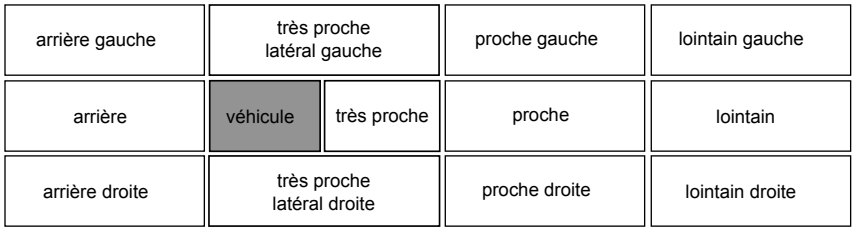


FIGURE 1.6 – Les zones de perception dans Archisim, sur lesquelles s'appuie le modèle de minimisation des interactions.

Calcul de la position finale Après la prise de décision, une étape complémentaire calcule la prochaine position du véhicule, à partir des données issues du modèle de décision : selon les cas il s'agira d'une accélération et d'un angle volant, ou d'un point cible. Afin de déterminer la position absolue du véhicule, ainsi que sa direction, son tangage et son roulis, un modèle dynamique véhicule de complexité variable est utilisé. Il permet de calculer précisément le comportement physique du véhicule, à partir d'entrées correspondant aux actions du conducteur (angle volant, pédales...) et des paramètres environnementaux (adhérence de la route, poids du véhicule...). Selon l'objectif de la simulation, un modèle plus ou moins précis est utilisé : cela peut aller d'un modèle très simple du type masse – ressort, à l'application du même modèle que celui utilisé pour la conception des châssis véhicules.

1.1.3.3 Les caractéristiques conducteurs

De nombreux paramètres influent sur le comportement des conducteurs, impliquant différentes dimensions psychologiques [Dewar, 2002] : personnalité, émotions, motivation, comportement social, etc. Ces paramètres sont souvent interdépendants, et difficiles à séparer : par exemple, la personnalité d'un conducteur a une forte influence sur ses émotions et sa motivation, et ces éléments impactent la prise de risque et l'agressivité au volant. Ils sont pris en compte dans différentes théories qui visent à expliquer le comportement des conducteurs [Keskinen et al., 2004] : certaines sont basés sur le risque [Naatanen & Summala, 1974, Wilde, 1982, Fuller, 1984], d'autres sont des théories comportementales [Fishbein & Ajzen, 1975, Ajzen & Fishbein, 1977], d'autres enfin s'appuient sur des modèles hiérarchiques du conducteur [Michon, 1985]. Cependant, ces modèles restent difficilement quantifiables.

Les ingénieurs du trafic s'orientent donc le plus souvent vers d'autres approches, au détriment de la validité psychologique. La grande majorité des modèles de simulation considèrent ainsi que les conducteurs peuvent percevoir les distances, les vitesses relatives, la vitesse absolue et / ou l'accélération des véhicules précédents. Pourtant, les capacités des humains à déterminer les distances, les vitesses absolues et l'accélération d'autres objets restent limitées, alors qu'ils sont capables d'estimer de manière précise les angles visuels, le temps au contact, la direction et les points saillants d'une scène [Boer, 1999].

De plus, la tâche de conduite est une tâche de satisfaction : le conducteur ne cherche pas à avoir une conduite parfaite, il se contente de se maintenir à un niveau satisfaisant [Hancock, 1999]. Vouloir modéliser le comportement du conducteur à partir d'équations optimales ou dériver des équations depuis une description physique du mouvement, puis essayer de faire correspondre les résultats avec des données issues de réponses comportementales, ne sont pas des démarches satisfaisantes d'un point de vue psychologique. Ingénieurs et psychologues ont donc des difficultés

à rapprocher leurs points de vue [Brackstone & McDonald, 1999b].

La question du niveau de description des modèles et du type de paramètre pris en compte est donc particulièrement complexe. Par exemple, si l'objectif est de concevoir un modèle se rapprochant de la réalité, faut-il simuler le conducteur lui même, jusqu'à la façon dont sa décision se décline en une force appliquée sur la pédale ? Ces questions restent ouvertes, et relèvent de travaux qui visent à faire converger les approches psychologiques et ingénierie. Cependant, comme le disent Brackstone & Mc Donald [Brackstone & McDonald, 1999b] :

> « Although we may argue about which one is the most "real", providing at the end of the day that it captures the required behaviour adequately, it does not really matter. »

La discussion reste finalement une question de choix d'ingénierie et d'axes de recherche.

En pratique, les modèles de simulation de trafic visent à intégrer du mieux possible les comportements observés dans le monde réel. Ils utilisent pour cela des paramètres variés, qui incluent en particulier [Gettman & Head, 2003] :

- la paramétrisation des modèles de gap-acceptance, de changement de ligne et de suivi de véhicules afin de permettre la prise en compte de différents types de conducteurs et de véhicules,
- la réaction aux feux oranges,
- le temps de réaction,
- des taux d'accélération et de décélération variables,
- une limitation des distances de perception,
- la capacité à négocier informellement lors de l'insertion sur route ou sur autoroute,
- la prise en compte de l'absence de clignotant.

L'utilisation de ces différents paramètres rend ainsi possible l'intégration d'une certaine variété dans le comportement des conducteurs. Cependant, les questions relatives à la création de styles de conduite cohérents et conformes à la réalité, ainsi que les problématiques liées à l'interdépendance des paramètres, restent le plus souvent ouvertes dans les approches existantes.

1.1.3.4 Synthèse

En conclusion, un simulateur de trafic pour la simulation de conduite est défini par les éléments fondamentaux suivants (Fig. 1.7) :

- l'environnement, contenant la base de données multi-couches (logique et graphique), ainsi que l'état instantané de la simulation (position et statut des véhicules),
- le modèle de trafic, avec d'une part les conducteurs capables de percevoir l'environnement et de décider de leurs prochaines actions en fonction de leurs caractéristiques propres, et d'autre part les véhicules, qui en fonction des décisions de leur conducteur, de leurs caractéristiques physiques et du modèle dynamique véhicule, vont déterminer la position dans l'environnement au prochain pas de temps de la simulation,
- le scénario, qui peut envoyer des instructions aux conducteurs (par exemple un changement de vitesse), aux véhicules (adhérence modifiée), ou à l'environnement (conditions météorologiques) et modifier ainsi l'état de la simulation,
- la sortie enfin, sous forme de visuel 2D, 3D, ou de statistiques.

Chaque élément est généralement implémenté dans un processus distinct, et chaque processus est amené à interagir avec les autres au sein de la simulation. Le contenu des différents éléments et la manière dont ils interagissent sont spécifiques aux choix de conception et d'implémentation faits pour chaque application.

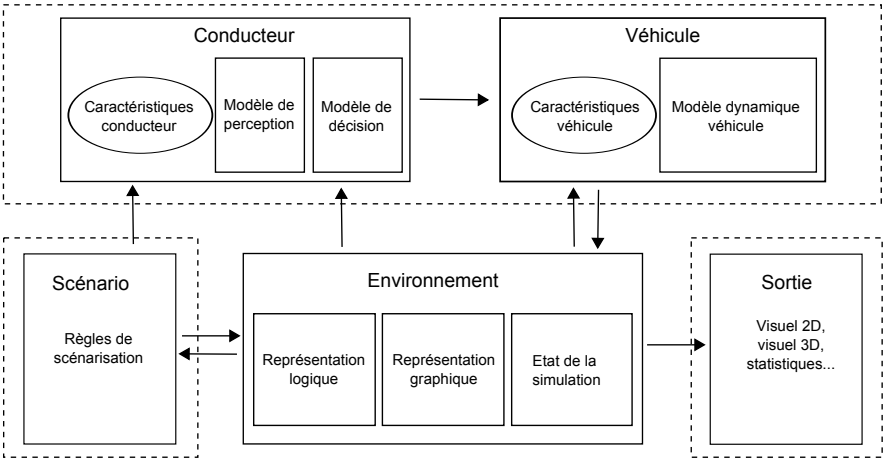


FIGURE 1.7 – Architecture d'un modèle de simulation de trafic dans le contexte de la simulation de conduite

1.1.4 Exemples d'applications existantes

De nombreux outils de simulation de conduite sont disponibles, développés par des universités, des instituts de recherche ou des industriels. Nous présentons ici quelques uns d'entre eux, choisis pour donner un panorama des différentes possibilités exploitées dans le domaine en terme de simulation de trafic.

Hank, développé par l'Université d'Iowa, utilise un modèle de trafic basé sur une approche originale pour la prise de décision : les automates concurrents hiérarchisés [Cremer et al., 1995, Wang et al., 2005]. Ceux-ci permettent d'introduire une hiérarchie entre automates, chaque état pouvant être lui-même un automate. Une parallélisation des calculs permet à plusieurs états d'être actifs simultanément. Cette approche permet d'améliorer à la fois la modularité du modèle, et les comportements des véhicules. Hank présente par ailleurs la particularité d'être utilisé sur un simulateur de conduite à vélo, ce qui permet de réaliser des études de sécurité impliquant des cyclistes [Kearney et al., 2006].

VTISim est l'application développée par l'institut national de recherche sur la route et les transports suédois (VTI) pour leurs simulateurs de conduite. Son modèle de trafic se base sur une décomposition en sous-modèles de la tâche de trafic, comme présenté dans la Section 1.1.3.2. Cette application intègre des développements spécifiques au cas des routes en milieu rural [Olstam, 2005], et utilise un modèle mésoscopique afin de gérer les véhicules qui ne sont pas présents dans le voisinage du conducteur [Olstam, 2002]. L'objectif est de réduire la charge de calcul, et de ne simuler que les véhicules les plus significatifs.

Archisim est l'application de simulation de conduite développée par l'INRETS. Elle est basée sur une architecture multi-agent, gérant des agents conducteurs, des agents infrastructures qui transmettent les informations sur l'environnement aux conducteurs, et des agents contrôlant l'équipement routier, comme les feux de signalisation. Le modèle de trafic utilise des études réalisées en psychologie de la conduite [Espié et al., 1994]. Il vise à reproduire le comportement des conducteurs, plutôt que de suivre des lois de flux ou de suivi : le principe mis en œuvre est la minimisation des interactions avec les autres usagers de la route, comme nous l'avons présenté dans la Section 1.1.3.2. Le modèle intègre par ailleurs des solutions basées sur la coordination [Champion, 2003, Mandiau et al., 2007] et l'anticipation [Doniec et al., 2008] afin

d'améliorer le comportement des véhicules dans les intersections.

SCANeR™, enfin, est une suite logicielle développée et utilisée par Renault, qui est aujourd'hui co-développée et distribuée par la société Oktal [Kemeny, 1993]. Son modèle de décision s'articule suivant une boucle « perception – décision – action », au sein d'un système multi-agents [Champion et al., 1999]. Lors de la phase de perception, chaque véhicule acquiert les connaissances sur son environnement. Le processus de décision utilise ensuite ces informations afin de déterminer le prochain objectif, par l'intermédiaire d'un automate à états. Pour finir, la prochaine position du véhicule dans l'environnement est calculée. SCANeR™ est utilisé à la fois sur des simulateurs de conduite, sur des postes de conception, et pour de la formation. L'application et son fonctionnement seront présentés plus en détails dans le Chapitre 4.

1.1.5 Le cas des jeux vidéos

Bien que les deux domaines soient en réalité très éloignés, nous pensons qu'il est intéressant de faire un parallèle entre les outils de simulation de conduite et les jeux vidéos, en particulier les jeux de courses. En effet, une personne non initiée se retrouvant pour la première fois face à un logiciel de simulation de conduite se pose rapidement la question : mais quelle est la différence avec un jeu vidéo ? Comme nous allons le voir, les différences sont en fait multiples.

Tout d'abord, l'objectif final est fondamentalement différent. Dans le cas des jeux vidéos, l'important est que l'utilisateur prenne plaisir à jouer : la problématique centrale est l'expérience de jeu, le « gameplay ». Pour les simulateurs de conduite, le point crucial est le réalisme de la simulation : il faut que l'immersion du conducteur soit de la meilleure qualité possible.

Les contraintes portent donc sur des points différents. Par exemple, les utilisateurs attendent d'un jeu vidéo qu'il soit particulièrement performant. Les bases de données seront donc simples d'un point de vue logique, afin d'optimiser les temps de calcul, mais le plus riche possible graphiquement. En effet, dans les jeux de courses les circuits sont souvent des boucles sans intersections, qu'il est possible de représenter de manière optimisée [Darby, 2004]. En simulation de conduite, cela n'est pas envisageable car les bases représentent souvent des environnements réels, dans toute leur complexité. Bien souvent, on ne souhaite d'ailleurs pas les simplifier, afin de permettre la réalisation de comparaisons de conduites réelles et de conduites sur simulateur. De plus, dans les jeux vidéos, la totalité des bases utilisables est fournie avec le jeu, alors que dans les simulateurs de conduite, les utilisateurs doivent pouvoir les créer eux-même. Elles ne peuvent donc pas être optimisées de la même manière, car elles sont susceptibles d'être modifiées.

En ce qui concerne le trafic, les problématiques sont là encore bien différentes. Le contexte des jeux vidéos a conduit au développement de modèles de gestion des véhicules autonomes très spécifiques, destinés à offrir la meilleure expérience de jeu possible et très orientés performances. Augmenter l'expérience de jeu conduit les concepteurs à biaiser sciemment la simulation : par exemple, si le joueur est en tête dans une course, la vitesse des véhicules suiveurs est augmentée afin de rendre le challenge plus intéressant [Biasillo, 2002]. Une autre possibilité est de rendre les données d'entrées des modèles imprécises, afin que les véhicules n'aient pas un comportement trop parfait [Funge, 2004]. D'autre part, l'orientation performances conduit au développement de modèles ne prenant en compte que les éléments indispensables. Cela se fait au prix de la généricité : par exemple, le plus souvent les intersections ne sont pas gérées et la signalisation n'est pas intégrée. Ces choix, tout à fait justifiés dans le cas des jeux, ne sont pas envisageables pour les simulateurs de conduite. En effet, le réalisme étant la problématique centrale, les modèles sont plus complexes afin de prendre en compte un maximum de paramètres des environnements

réels. Ils nécessitent donc plus de temps de calcul. Le besoin de scénarisation des véhicules ajoute aussi à cette complexité, car les contraintes impliquées doivent être prises en compte. Les modèles doivent également être flexibles et génériques, afin de s'adapter aux besoins des utilisateurs, qui seront différents suivant le type d'étude réalisé ou le pays où elle a lieu.

Enfin, une différence fondamentale entre les deux domaines se situe au niveau des moyens mis en œuvre pour le développement. Par exemple, d'un point de vue financier, la valeur d'un logiciel comme SCANeR™ est évaluée à environ 8 millions d'euros, pour un développement s'étalant sur une période de 13 ans (1996-2009) : cela représente donc un budget d'environ 600 000 € par an. Dans le cas des jeux vidéos, les coûts de développement ne sont pas communiqués, mais il est possible d'estimer leur ordre de grandeur. Ainsi, d'après le directeur d'Electronics Arts Montréal, pour un titre de la série Need for Speed, les équipes sont constituées de 35 à 85 personnes, et les projets ont une durée de 12 à 24 mois [Gameindustry.biz, 2006, Gameindustry.biz, 2008]. On arrive donc à un coût par projet de 8.5 M€, en ne prenant en compte que les ressources humaines (hors coûts de licences, de communication, de publicité...) [4]. Ramené à un coût annuel, le budget moyen pour un titre est donc de plus de 4 millions d'euros par an, près de 7 fois plus que celui de SCANeR™. Comparer les résultats reste donc périlleux, d'autant plus que les marchés, les besoins et les enjeux sont presque totalement disjoints.

Les deux domaines peuvent difficilement être comparés, car les objectifs et les moyens diffèrent trop, mais ils possèdent cependant des points de convergence qui méritent d'être notés. Le domaine de la simulation de conduite s'enrichit ainsi d'avancées et de techniques provenant du monde du jeu vidéo. Par exemple, les visuels sont développés en utilisant des formats et des techniques similaires. Des optimisations de calcul pour les modèles véhicules s'appuient également sur des innovations réutilisables.

1.2 Approche centrée individu et systèmes multi-agents

Comme nous l'avons vu, certains simulateurs de trafic sont conçus suivant une architecture multi-agents. C'est en particulier le cas de l'application développée et utilisée chez Renault, SCANeR™. Dans cette seconde partie, nous nous intéressons donc à l'approche centrée individus. Dans un premier temps, nous rappelons les principales propriétés et caractéristiques des systèmes multi-agents. Nous précisons ensuite les différents éléments les constituant, avant d'illustrer l'approche par des exemples d'application.

1.2.1 Les systèmes multi-agents

Cette première section rappelle les principales caractéristiques des systèmes multi-agents, et présente les différentes utilisations qui peuvent en être faites.

1.2.1.1 Caractéristiques

Afin de présenter les différents aspects d'un système multi-agents, nous considérons l'approche VOYELLES [Demazeau, 1995]. Dans cette approche, le système s'articule autour de quatre dimensions : l'Agent, l'Environnement, l'Interaction et l'Organisation. Le système multi-agents peut alors être défini par un ensemble A d'entités autonomes dotées de capacités de perception et

4. Pour une équipe de 60 personnes pendant 2 ans, avec une rémunération moyenne de 70 k€ annuel.

d'action, évoluant dans un environnement E et Interagissant dans le cadre d'une organisation O. Nous reprendrons ces différentes dimensions dans la Section 1.2.2 afin de préciser les propriétés des éléments constitutifs du système.

Un système multi-agents est caractérisé par l'absence de contrôle global. La prise de décision est décentralisée, et chaque agent possède un certain degré d'autonomie dans ses actions. Une des conséquences directes de l'aspect décentralisé de la prise de décision est l'émergence : cette notion caractérise l'apparition de phénomènes au niveau du système, alors que les caractéristiques individuelles des agents ne contiennent aucune référence à ce phénomène. L'émergence représente donc le passage du niveau local au niveau global, ou encore du niveau microscopique au macroscopique. Notons que la notion d'émergence est largement discutée dans la littérature, en particulier en fonction des éléments sur lesquels elle s'appuie. Nous ne détaillerons cependant pas ici ces discussions, qui n'ont pas d'impact sur notre problématique.

1.2.1.2 Catégorisation

Différentes catégories de systèmes multi-agents peuvent être distinguées, en fonction du type de problème ciblé. La première catégorie concerne la résolution de problèmes distribués. Dans ce cadre, l'objectif est que les agents coopèrent et se coordonnent pour résoudre ensemble un problème global. Chaque agent se voit typiquement confier une sous-tâche de l'objectif, sans nécessairement connaître le but final. Les problématiques soulevées sont liées à la communication, la coordination, ou encore l'allocation des tâches.

La seconde catégorie contient les applications dans lesquelles les agents sont conçus pour jouer un rôle similaire à celui d'un être humain. Cela peut concerner par exemple la conception d'assistants virtuels, auxquels est délégué la tâche de maintenir l'agenda de leur correspondant humain, ou des agents destinés à enchérir sur des plateformes comme eBay lors de l'absence du propriétaire du compte. Les problématiques concernent ici en particulier la sécurité et les communications.

Enfin, la dernière catégorie regroupe les applications dont le but est de simuler des phénomènes issus du monde réel, l'objectif étant de les reproduire pour les comprendre ou les expliquer. Les systèmes multi-agents ont ainsi rencontré un fort intérêt dans des domaines comme la simulation sociale ou l'éthologie, où l'apparition de phénomènes macroscopiques n'est pas expliquée simplement par les compétences des agents. La simulation centrée individus permet de comprendre quels éléments ou quelles interactions conduisent à l'apparition de ces propriétés globales.

Notre application se situe dans cette dernière catégorie. Nous reviendrons donc plus loin sur l'intérêt de la simulation centrée individu.

1.2.2 Composition d'un système multi-agents

Intéressons nous maintenant aux différents éléments constituant un système multi-agent selon l'approche VOYELLES (cf. Sect 1.2.1.1) : les agents, l'environnement, les interactions et l'organisation.

1.2.2.1 Les agents

Diverses définitions ont été proposées pour la notion d'agent. En effet, ce concept est utilisé dans des domaines variés : agents conversationnels animés, applications réparties, simulation informatique, résolution de problème par émergence, etc. Ces contextes présentent des particularités qui conduisent à adapter la définition utilisée. Ferber propose la définition suivante [Ferber, 1995] :

> « On appelle agent une entité physique ou virtuelle :
>
> 1. qui est capable d'agir dans un environnement,
> 2. qui peut communiquer directement avec d'autres agents,
> 3. qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle chercher à optimiser),
> 4. qui possède de ressources propres,
> 5. qui est capable de percevoir (mais de manière limitée) son environnement,
> 6. qui possède des compétence et offre des services,
> 7. qui peut éventuellement se reproduire,
> 8. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit. »

Cette définition présente l'intérêt de présenter l'ensemble des éléments clés relatifs au concept d'agent. Suivant l'accent mis sur les différents points de la définition, différentes catégories d'agents peuvent être distinguées [Wooldridge, 2002] : les agents réactifs, les agents cognitifs, et les agents hybrides.

Pour les agents réactifs, la perception entraîne l'action par un couplage direct. La phase de cognition est réduite au minimum, voire est inexistante : elle est restreinte au déclenchement de règles comportementales. L'utilisation d'agents réactifs repose sur le fait que l'émergence d'un comportement au niveau global ne s'appuie pas nécessairement sur des comportements individuels complexes. C'est le cas par exemple dans les colonies de fourmis : le dépôt de phéromones par les agents fourmis sur leur trajet vers les sources de nourriture, couplé à l'action de suivre les pistes les plus odorantes, suffit à faire émerger un comportement global [Dorigo et al., 1996].

Les agents cognitifs au contraire utilisent des raisonnements complexes lors de leur phase de décision. Ce sera le cas par exemple d'agents planifiant leurs but : devant une porte fermée, l'agent peut réorganiser ses objectifs afin de chercher la clef correspondante dans l'environnement. Ces agents peuvent disposer d'une représentation de l'environnement, afin de raisonner sur celui-ci et de déterminer les actions à exécuter. Ils peuvent par ailleurs disposer de processus élaborés de communication, par exemple s'ils doivent collaborer pour d'atteindre leurs buts.

Cependant, la frontière entre agents réactifs et cognitifs reste floue. De nombreuses architectures d'agents peuvent être considérées comme hybrides, mettant en application des comportements réactifs ou cognitifs suivant les cas. Le cas des conducteurs peut en présenter un exemple typique : si l'agent est en mesure de planifier son itinéraire en fonction de sa destination, il possède une dimension cognitive, et s'il est capable de réagir immédiatement lors d'une situation critique comme un freinage d'urgence, il possède également une dimension réactive.

1.2.2.2 L'environnement

L'environnement décrit le contexte dans lequel les agents évoluent. En fonction de l'application, il correspond à des ensembles très différents : par exemple, dans le cadre d'agents communicationnels, il est constitué par le réseau entre les agents ; pour des agents géographiquement situés, il contient la description du contexte, comme la base de données, les éléments logiques... Il peut par ailleurs être utilisé comme vecteur de communication entre les agents : c'est le cas par exemple dans les approches utilisant un tableau noir [Corkill, 1991].

L'environnement possède différentes propriétés [Russell & Norvig, 1995] :

- il peut être accessible ou inaccessible : dans un environnement accessible, il est possible d'obtenir des informations complètes, précises et à jour sur l'état de l'environnement. Les agents reposent sur la précision des informations à leur disposition pour prendre leurs décisions : si ces informations sont incomplètes ou imprécises, leurs décisions peuvent être erronées. Notons que la plupart des environnements réels sont inaccessibles,
- l'environnement peut être déterministe ou non-déterministe : dans un environnement déterministe, toutes les actions ont un effet garanti et reproductible. Si l'environnement est non déterministe, un agent ne peut pas être sûr que l'action qu'il a réalisé a été prise en compte dans l'environnement : il doit considérer qu'elle peut avoir échoué,
- l'environnement peut être statique, ou dynamique : un environnement statique reste inchangé tant que l'agent ne réalise pas d'action. Dans le cadre multi-agent, l'environnement est le plus souvent dynamique : les autres agents pouvant eux-même agir sur le monde, aucun agent ne peut supposer son environnement statique,
- enfin l'environnement peut être discret ou continu : un environnement est discret s'il existe un nombre fini d'actions possibles. Par exemple, un jeu d'échec est un environnement discret : le nombre d'états et d'actions possibles dans l'environnement est fini, même s'il est très élevé.

Notons que dans la plupart des applications multi-agent, l'environnement est inaccessible, non-déterministe, dynamique et continu. C'est en particulier le cas pour la simulation de trafic dans SCANeR™.

En ce qui concerne les environnements géographiquement situés, il est intéressant de distinguer les représentations discrètes et continues de l'espace. Lors de la conception d'un système multi-agent, ces deux possibilités sont en effet envisageables. Dans le cas d'une représentation discrète de l'espace, l'environnement est le plus souvent représenté sous forme de grille. Chaque case de la grille possède des coordonnées entières, et chaque agent est situé sur une case. La vitesse d'un agent est un nombre de cases à franchir en un pas de temps. Dans le cas d'une représentation continue de l'espace, l'environnement est représenté sous forme de coordonnées continues. La position d'un agent est décrite par des coordonnées réelles (x, y), ou (x, y, z) en environnement à trois dimensions. La vitesse d'un agent est un vecteur $\vec{v}$ de composantes réelles.

Chacune des approches possède ses avantages et ses inconvénients. Dans le cas de la représentation discrète, les calculs de collision sont très simples : la case de la grille vers laquelle l'agent se dirige est vide, ou non. Les capacités de représentation de l'environnement restent cependant limitées : la vitesse est nécessairement un nombre entier de cases, et les agents ne doivent occuper qu'une case ou un ensemble rectangulaire de cases à la fois. Cela limite la variété des comportements et la finesse de représentation des agents.

Notons qu'en introduisant une dimension temporelle dans la simulation, en plus de la dimension spatiale, certaines de ces contraintes peuvent être contournées : par exemple, la vitesse peut être vue non comme un nombre de cases de la grille à franchir, mais comme un nombre de

pas de temps à attendre entre chaque franchissement de case : un agent de vitesse 3 se déplace trois fois plus souvent qu'un agent de vitesse 1. Cette représentation reste cependant moins intuitive, et impose la prise en compte d'autres problématiques : par exemple, un agent de vitesse 1 applique-t-il son processus de décision à chaque pas de temps ? Si oui, alors il prends 3 fois plus de décisions que l'agent de vitesse 3, qui doit attendre avant d'effectuer son action...

Dans le cas d'une représentation continue, les calculs sont plus complexes, que ce soit pour les collisions, les déplacements ou l'arrivée en bordure de l'environnement. De plus, les questions d'approximations impliquées par des calculs avec des réels causent fréquemment des problèmes dans les simulations, s'ils ne sont pas correctement pris en charge. En contrepartie de ces complexités, la description de la position et du déplacement des agents est plus riche : la vitesse d'un agent peut avoir une direction et une valeur quelconques, les agents peuvent s'approcher sans limitation de taille de case de grille, etc.

Il ne faut donc pas négliger de considérer systématiquement la représentation discrète, qui permet de s'affranchir de nombreuses problématiques de calcul, le choix final reposant principalement sur les critères de représentation.

1.2.2.3 Les interactions

Les interactions correspondent aux relations possibles entre les agents, et sont capitales pour les systèmes multi-agents. En effet, elles sont la source de l'émergence d'un comportement global.

Certaines approches les placent ainsi au centre du système. Par exemple, IODA [5] définit les interactions indépendamment des agents, en les séparant au niveau logiciel [Kubera et al., 2008a, Mathieu et al., 2001]. Les agents peuvent subir ou effectuer un nombre quelconque d'interactions, définies de manière générique : une interaction comme « ouvrir » a la même signification quel que soit le type d'agent qui l'exécute, et peut donc être réutilisée indépendamment de celui-ci. Les interactions sont constituées de séquences d'actions s'appliquant à des agents et soumises à des conditions d'activation. Ils peuvent y jouer différents rôles : être l'agent source, qui effectue l'interaction, ou l'agent cible, qui la subit. Par exemple, pour l'interaction « ouvrir », un agent « humain » pourra être la source et un agent « porte » être la cible. La condition d'activation concerne la portée de l'interaction : dans le cas d'« ouvrir », les agents doivent être voisins pour qu'elle puisse se produire. L'ensemble des interactions est ainsi défini en dehors des agents, et une matrice décrit les assignations possibles entre sources et cibles pour une simulation (Tab. 1.1). En concrétisant les interactions entre les agents, cette méthodologie vise à faciliter la conception et à améliorer la réutilisabilité du code.

Les interactions peuvent aussi prendre la forme de communications entre les agents, qu'elles soient directes ou indirectes. Les communications directes reposent sur l'envoi de messages, et utilisent les différents modes de communication existant en informatique. Des protocoles dédiés, comme FIPA-ACL [6], ont été introduits afin de proposer des standards destinés aux systèmes multi-agents. Les communications indirectes reposent sur l'utilisation de l'environnement comme support de communication : les agents y laissent des traces qui pourront être utilisées par les autres agents. C'est par exemple le cas pour les colonies de fourmis, où les phéronomes déposés par les fourmis constituent un moyen de communication indirecte.

5. Pour *Interaction-Oriented Design of Agent simulations.*
6. *Foundation for Intelligent Physical Agents - Agents Communication Langage* [FIPA, 1996].

source \ cible	∅	Herbe	Mouton	Chèvre	Loup
Herbe	se reproduire				
Mouton	mourir se déplacer	manger	se reproduire		
Chèvre	mourir se déplacer	manger		se reproduire	
Loup	mourir se déplacer		manger	manger	se reproduire

TABLE 1.1 – Exemple de matrice d'interaction pour un modèle proie / prédateur avec quatre espèces (d'après [Kubera et al., 2008b]).

1.2.2.4 L'organisation

Enfin, la dernière dimension du système multi-agents est l'organisation. Elle permet de spécifier un cadre global de fonctionnement pour le système, afin de guider son comportement.

Par exemple, l'utilisation de rôles par les agents fournit une structure organisationnelle qui spécifie certains aspects statiques du fonctionnement du système [Ferber et al., 2004]. Ces structures de haut niveau présentent l'avantage de préserver l'autonomie des agents, afin de conserver les propriétés et les avantages de l'approche centrée individus. Les systèmes multi-agents normatifs [Boella et al., 2006], sur lesquels nous reviendrons dans la Section 2.1, font également partie de cette démarche : l'objectif est de fournir des structures organisationnelles, ici sous forme de normes définissant des obligations et des interdictions, afin de faciliter le bon fonctionnement de la société d'agents. Enfin, certaines approches visent à intégrer ces différents aspects [Hübner et al., 2002]. En spécifiant indépendamment la structure et le fonctionnement de l'organisation et en les reliant par des aspects normatifs, les agents peuvent profiter de plans pré-conçus lorsque ceux-ci sont adaptés à la tâche en cours, et être guidés par la structure de l'organisation s'ils ne le sont pas.

L'auto-organisation des systèmes représente une autre approche de cette problématique. L'objectif est alors de fournir au système la capacité de construire une organisation lui permettant de répondre au but fixé. Par exemple, cela permet d'obtenir des systèmes s'auto-organisant à partir d'agents représentant des instructions élémentaires, comme des instructions de langages de programmation. A partir de spécifications de plus haut niveau, le programme s'auto-conçoit de manière à fournir les fonctionnalités souhaitées [Georgé & Gleizes, 2005]. Dans une autre optique, des méthodologies de développement reposant sur ces principes visent à pallier les difficultés de spécification et de conception des applications, en utilisant des systèmes multi-agents auto-adaptatifs par auto-organisation coopérative [Picard et al., 2004].

Enfin, l'intérêt grandissant pour les systèmes multi-agents ouverts, que les agents sont susceptibles de rejoindre ou de quitter à tout moment, renforce ce type de besoin organisationnel [Hübner et al., 2009]. Les agents doivent en effet être capables d'apprendre les règles du système lorsqu'ils arrivent, et le système doit être capable de s'adapter lorsque des agents le quittent. Rôles et normes peuvent être combinés pour spécifier l'organisation, afin de définir des langages et des sémantiques opérationnelles pour son implémentation [Tinnemeier et al., 2009].

1.2.3 La simulation centrée individus

Dans cette partie, nous présentons l'intérêt de l'approche centrée individus, et l'illustrons par deux exemples : l'utilisation de comportements réactifs pour l'animation d'agents et la simulation de foules.

1.2.3.1 Intérêt de l'approche

L'approche centrée individus rencontre un intérêt grandissant dans le domaine de la simulation. Elle est en effet bien adaptée à la modélisation de systèmes massivement distribués, dont le comportement global est issu des interactions entre ses parties.

En particulier, elle présente l'intérêt d'obtenir un modèle explicatif du phénomène simulé. Dans les simulations qui ne sont pas multi-agents, le modèle reproduit le phénomène observé dans le monde réel, le plus souvent à partir d'équations, mais ne permet pas d'expliquer les raisons à l'origine de son apparition. Dans le cas de l'approche centrée individus, le modèle est explicatif, car ce sont les interactions entre les agents qui entraînent l'apparition du phénomène recherché.

Par exemple, dans le modèle « proie – prédateur » de Lotka et Volterra [Volterra, 1931], la dynamique de l'environnement peut être décrite par un système d'équations différentielles. Les solutions reproduisent des observations réelles, comme la disparition d'une population si le taux de natalité est insuffisant. Cependant, les résultats ne permettent pas d'expliquer comment ou pourquoi les phénomènes se produisent : ils s'appuient sur des paramètres globaux, et non sur le comportement des individus. En réalisant le même type de simulation avec un modèle centré individu, des explications plus précises peuvent être apportées.

De plus, les simulations centrées individus présentent l'avantage d'utiliser le même vocabulaire que celui du domaine simulé. Les agents sont en effet conçus de manière analogue aux éléments du monde réel qu'ils modélisent. Cela simplifie la communication entre les experts du domaine simulé et les informaticiens concevant la simulation. Par exemple, dans la simulation biologique, l'approche centrée individus est utilisée afin de reproduire les mécanismes cellulaires : les interactions entre cellules et entre molécules sont transposées directement dans le monde des agents, ce qui facilite fortement le dialogue avec l'utilisateur final.

L'approche est ainsi utilisée dans des domaines variés, comme les sociétés humaines, les sociétés animales, ou le fonctionnement des écosystèmes [Schelling, 1972, Drogoul, 1993]. Nous détaillons ci-dessous deux exemples : les *steering behaviors* et la simulation de foules.

1.2.3.2 Les *steering behaviors*

Dans le cadre de ses recherches sur l'animation de personnages autonomes, Craig Reynolds a développé des techniques permettant de créer des comportements réalistes à partir de briques élémentaires simples [Reynolds, 1999]. Il se place dans le contexte d'agents virtuels situés, personnifiés et réactifs : les agents évoluent dans un environnement virtuel, dans lequel ils possèdent une représentation physique et réagissent « instinctivement » aux évènements extérieurs. L'objectif est que les agents produisent des comportements qui aient l'air naturels et réels. Le domaine d'application ciblé est l'animation et les médias interactifs comme les jeux ou la réalité virtuelle (par exemple, les personnages non-joueurs des jeux vidéos).

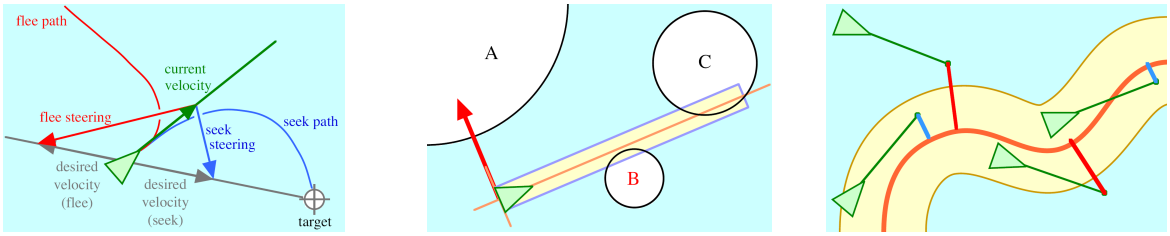


FIGURE 1.8 – De gauche à droite (d'après [Reynolds, 1999]) : le comportement de recherche, l'évitement d'obstacle et le suivi de chemin.

Les comportements proposés se situent à un niveau intermédiaire de la prise de décision de l'agent : ils ne concernent pas le choix stratégique de réalisation d'une action, ni la mise en œuvre finale du déplacement proprement dit. Ces niveaux sont laissés à la convenance de l'utilisateur : l'approche proposée est ainsi générale, et ne dépend pas de la « personnification » de l'agent. Elle fonctionne de manière identique pour un humain, une voiture, un vaisseau spatial, un sous-marin... La résultante d'un comportement est un vecteur de direction, qui est ensuite combiné à la vitesse actuelle pour déterminer le prochain point cible. Parmi ces comportements élémentaires, on trouve par exemple :

1. La *recherche*, qui permet de diriger le personnage vers une position spécifique de l'espace (Fig. 1.8). Ce comportement ajuste la vitesse de l'agent afin qu'elle soit dirigée radialement vers l'objectif : un vecteur « vitesse désirée », dirigé vers la cible, est construit ; la différence entre la vitesse actuelle et la vitesse désirée donne le vecteur direction résultat. À partir de ce comportement se déclinent ceux de *fuite* (la vitesse désirée est dirigée à l'opposé de la cible), de *poursuite* (le comportement de *recherche* est appliqué à un objectif mobile dont on estime la position future), et d'*évasion* (identique à *poursuite*, mais en appliquant la *fuite* au lieu de la *recherche*).
2. Le comportement d'*évitement d'obstacles*, qui permet à l'agent d'évoluer dans un environnement encombré. L'agent utilise un cylindre virtuel, dirigé vers l'avant : si un obstacle est détecté dans ce cylindre, cela déclenche une réaction. La direction correctrice est calculée en prenant l'opposé de de la projection du centre de l'obstacle sur la perpendiculaire à la direction de l'agent (Fig. 1.8). Cette correction permet à l'agent d'éviter les obstacles de manière fluide.
3. Le *suivi de chemin*, qui permet à un agent de suivre un chemin prédéterminé, sans toutefois le contraindre de manière rigide sur une trajectoire. L'agent évolue ainsi au voisinage du chemin, comme un humain marche sur un trottoir sans forcément rester systématiquement exactement au milieu. Pour ce faire, il se base sur une estimation de sa position future : si elle est en dehors de la zone de variation autorisée, une correction est appliquée. Le comportement *recherche* est appliqué vers la projection de la position estimée sur le chemin (Fig. 1.8).

En plus de ces comportements individuels, des comportements de groupe peuvent être utilisés. Ils reposent sur l'observation des agents dans le voisinage. Par exemple, la *séparation* exerce une force de répulsion pour chaque agent du voisinage, proportionnellement à la distance. Elle permet d'éviter que les agents ne se regroupent trop. La *cohésion* pousse les agents à se regrouper : elle exerce une force vers le centre de gravité des agents présents dans le voisinage. Enfin, l'*alignement* applique à l'agent une correction correspondant à la moyenne des vecteurs vitesse de tous les autres agents. Notons qu'utilisés ensemble, ces trois comportements constituent le modèle *boids* [Reynolds, 1987].

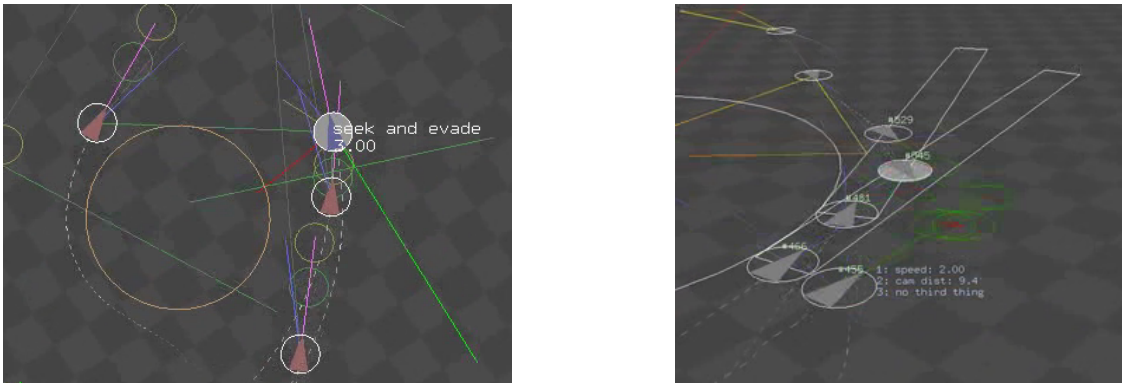


FIGURE 1.9 – À gauche, un exemple de poursuite. À droite, une foule suivant un chemin commun.

Enfin, les comportements élémentaires peuvent être combinés afin de créer des comportements plus complexes. Les combinaisons peuvent consister en l'utilisation alternée de certains d'entre eux : si je cherche de la nourriture avec *recherche*, et qu'un prédateur apparaît, j'utilise l'*évasion* et abandonne complètement l'autre comportement. Toutefois, je peux avoir envie de combiner l'*évitement d'obstacle* avec mon *évasion* afin de ne pas rentrer dans un arbre. Ces combinaisons se font typiquement soit en réalisant une somme pondérée des vecteurs directions résultants de chacun des comportements, soit en privilégiant un comportement particulier, afin d'éviter que les effets ne s'annulent mutuellement. L'agrégation permet ainsi de reproduire des situations comme la poursuite d'un attaquant ou le déplacement simultané de nombreux agents dans un environnement complexe (Fig. 1.9).

Ces différentes méthodes sont appliquées dans le monde du jeu vidéo et de la vie artificielle, et présentent un intérêt certain pour introduire du réalisme dans les simulations. De plus, elles sont peu coûteuses en temps de calcul, car conçues pour être très efficaces. L'introduction de comportements naturellement variés, en limitant les contraintes afin que chaque agent agisse de manière légèrement différente, peut être obtenu par ce type de technique.

1.2.3.3 La simulation de foules

Un autre domaine qui utilise largement les approches centrées individus est celui de la simulation de foules. Le champ d'application couvre le design architectural (mise en situation d'un nouveau bâtiment pour des besoins d'illustration, estimation des niveaux de service), la planification urbaine (évaluation des impacts des modifications de l'environnement), la sécurité (simulation d'évacuation), la thérapie de troubles sociaux par la réalité virtuelle (pour l'agoraphobie par exemple)...

Dans les applications développées, les problématiques concernent principalement deux dimensions : le comportement des agents d'une part, et le rendu graphique d'autre part. Elles sous tendent par ailleurs des questions relatives à la représentation de l'environnement et aux performances.

La simulation du comportement des agents est envisagée suivant deux approches. La première est d'utiliser des modèles visant à reproduire le comportement de piétons réels, afin de tester des hypothèses et de prédire le déroulement de phénomènes globaux. L'objectif est alors de pouvoir valider les résultats de la simulation, par exemple avec des mesures de flux macroscopiques [Teknomo, 2002, Berrou et al., 2005]. La seconde approche est d'utiliser des modèles qui

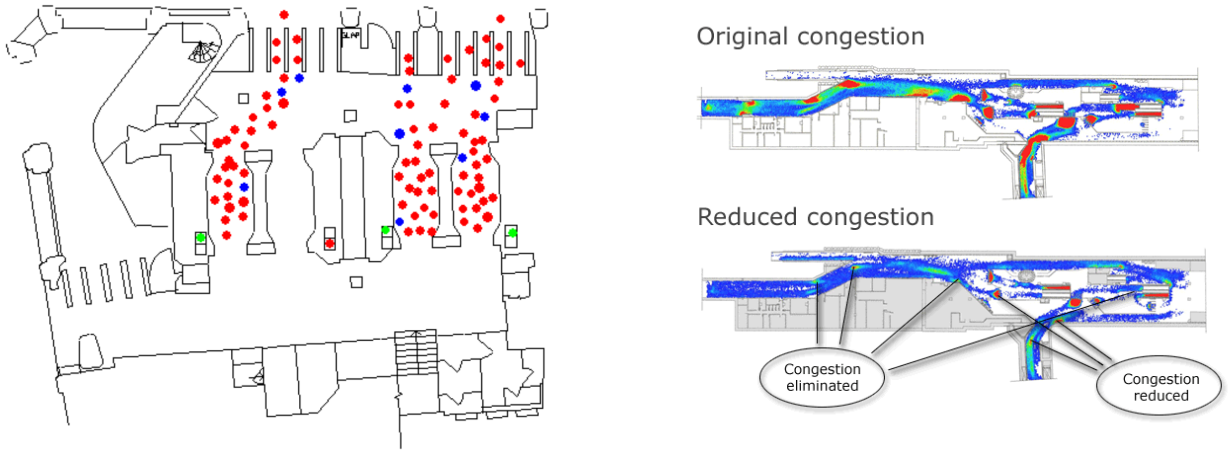


FIGURE 1.10 – Deux exemples de simulation de foules de piétons avec l'application Legion (d'après [Legion, 2009]) : l'objectif ici est d'optimiser les flux et de réduire les points de congestion dans des stations de métro.

permettent de donner une apparence réaliste à la simulation, afin qu'elle paraisse vraisemblable, sans s'attacher toutefois à la validité des résultats.

Des modèles de simulation basés sur des analogies avec les forces, comme le Social Force Model [Helbing & Molnar, 1995] ou le Magnetic Force Model [Okazaki, 1979] trouvent leur place dans la première catégorie. Ils permettent par exemple de reproduire avec succès des phénomènes de panique [Helbing et al., 2000]. Dans la seconde catégorie on trouvera des modèles plus proches du monde de l'animation, parfois inspirés des travaux de Reynolds [Reynolds, 1999] présentés plus haut. Par exemple, Shao & Terzopoulos développent des techniques d'un type comparable pour animer des foules dans des simulations de vie artificielle [Shao & Terzopoulos, 2007]. D'autres travaux enrichissent encore ces comportements en ajoutant des comportements à des niveaux de description plus élevés, comme les groupes d'individus [Musse & Thalmann, 2001].

Les questions de performance impliquent des problématiques complexes au niveau graphique. Dès lors qu'un utilisateur est immergé dans la simulation, les agents doivent être en mesure de réagir à sa présence : les scènes sont donc calculées en temps réel. Afin d'optimiser les calculs, des représentations de l'environnement spécifiques sont utilisées. Au contraire de la simulation de conduite, où l'environnement est particulièrement contraint (voies et sens de circulation, signalisation...), les piétons évoluent dans un espace très ouvert. Diverses approches sont ainsi utilisées pour le simplifier, en utilisant par exemple des représentations de plus haut niveau de l'espace d'évolution [Lamarche & Donikian, 2004, Paris et al., 2006]. Au niveau graphique, des techniques spécifiques sont développées, comme par exemple faire varier le niveau de détails des modèles 3D des piétons en fonction de la proximité de la caméra [Maim, 2009].

Nombre d'outils commerciaux dédiés à ces différentes problématiques ont vu le jour. Dans le domaine de la simulation de piétons on peut par exemple citer Legion, de Legion Limited, ou MASA, de MASA Group ; pour la simulation d'évacuation, Exodus, de l'Université de Greenwich ; dans le monde du jeu vidéo, Massive, de Massive Software, qui a servi à l'animation des scènes du Seigneur des Anneaux. La Figure 1.10 illustre l'utilisation de Legion pour l'amélioration des flux dans des bâtiments existants.

Dans ce type d'application, les simulations centrées individus présentent de nombreux points forts : elles permettent d'évaluer les impacts microscopiques des modifications (ajout d'une sortie

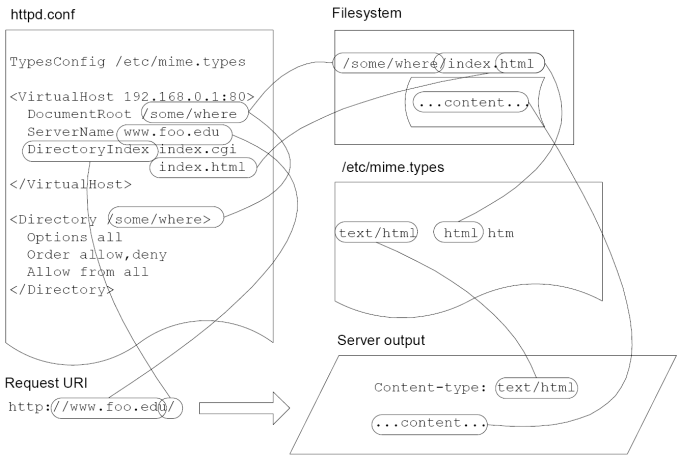


FIGURE 1.11 – Les contraintes de configuration permettant de répondre correctement à une requète http (d'après [Schwartberg & Couch, 2004]).

de secours), d'intégrer la présence d'agents de différents types (handicapés, personnes âgées, jeunes enfants...), et sont interactives (les agents peuvent réagir à la présence d'un humain dans la simulation).

1.3 Travaux connexes

L'objectif de ce travail est d'améliorer le réalisme de la simulation à travers la prise en compte simultanée de la variété et de la cohérence du comportement des agents. Dans la dernière partie de ce chapitre, nous présentons différents travaux qui s'intéressent à ces aspects. Les deux premières sections concernent l'automatisation de la gestion de configurations ; les deux dernières sont ciblées sur la simulation centrée individus.

1.3.1 Gestion de configuration

Le premier exemple que nous considérons est celui du maintien de la cohérence de la configuration en administration système [Couch et al., 2003]. Une problématique importante dans ce domaine est de garantir un temps de disponibilité maximal, ce qui se traduit en particulier au niveau administration par la maîtrise de la configuration des services mis à disposition sur une machine ou un cluster de machines. Ces services peuvent être administrés par des personnes différentes, et leur paramétrage est souvent interconnecté (Fig. 1.11). Par exemple, le fonctionnement d'un serveur web repose sur l'attribution de droits d'accès corrects dans le système de fichier, et ces droits d'accès sont généralement modifiables non seulement par l'administrateur du serveur, mais aussi par d'autres utilisateurs. Afin d'éviter les erreurs, des méthodes de maîtrise de la cohérence de la configuration ont été développées. Une des approches utilisée est d'abstraire la configuration bas niveau par une couche intermédiaire ajoutant un contrôle supplémentaire [Burgess & Couch, 2006]. Cela permet en outre de diminuer l'expertise nécessaire à l'administration.

L'approche repose sur la construction de domaines dits de « sémantique prédictible », dans lesquels le lien entre entrées et sorties est garanti. Ces domaines sont appelés des *closures*. Leur

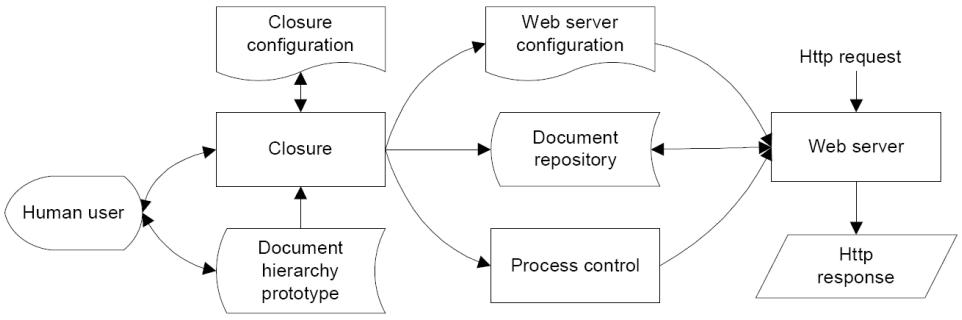


FIGURE 1.12 – Une *closure* sert d'intermédiaire entre l'utilisateur et la configuration du serveur, réduisant la complexité du processus de configuration (d'après [Schwartberg & Couch, 2004]).

principale propriété est qu'elles ne peuvent pas produire d'effets inconnus, leur comportement étant totalement déterminé par leurs interactions avec le monde extérieur. Notons que dans le contexte considéré, le mot comportement a une signification particulière : un comportement est représenté par un ensemble de tests qui peuvent être vrai ou faux. Par exemple, la question « Est-ce que le système A fait fonctionner un serveur http sur le port 80 ? » définit un comportement.

Les *closures* se définissent formellement de la façon suivante [Burgess & Couch, 2006] :

Définition 1 *Une* closure D *est un service* $\langle C_D, F_D \rangle$ *où* C_D *décrit des contraintes sur les entrées et* F_D *est une fonction faisant correspondre des réponses aux entrées.* F_D *mappe chaque séquence* S *d'entrées, chacune d'entre elles obéissant à* C_D*, à un sortie unique* $F_D(S)$ *(qui peut être vide).*

Une *closure* agit comme une fonction de ses entrées. Elle est une unité indépendante dans une configuration, en ce qu'elle ne fait que réagir aux entrées qu'elle reçoit. Elle peut représenter un nœud de configuration et être combinée en des *closures* de plus haut niveau, contenant plusieurs services.

L'ensemble des *closures* d'un système doit être cohérent. La cohérence est définie comme le fait que si deux *closures* possèdent un ou plusieurs paramètres communs, les valeurs de ces paramètres doivent être égaux :

Définition 2 *Deux configurations* $c \in \mathcal{D}_\mathcal{A}$ *et* $c' \in \mathcal{D}_\mathcal{B}$ *sont cohérentes si pour chaque paramètre* $p \in \mathcal{V}_\mathcal{A} \cap \mathcal{V}_\mathcal{B}$, $c(p) = c'(p)$*, où les* $\mathcal{D}_i$ *sont les configurations possibles, et les* $\mathcal{V}_i$ *les domaines de définition des paramètres.*

Les *closures* sont complétées par les *aspects*, qui spécifient des comportements du système indépendamment des paramètres externes. Les composants doivent coopérer pour atteindre un état qui leur convient, la négociation se faisant par l'intermédiaire de *promises*, qui représentent la spécification d'un état ou d'un comportement futur d'un agent. Les *agents* sont des entités qui peuvent faire ou recevoir des *promises*, et qui prennent leurs décisions de manière autonome. Pour finir, les *aspects* sont des ensembles de *promises*, et les *promises* avec leurs agents peuvent former des *closures*.

Ces concepts permettent d'abstraire la configuration du système. L'utilisateur spécifie les *aspects*, qui sont un concept haut niveau indépendant des paramètres externes de configuration,

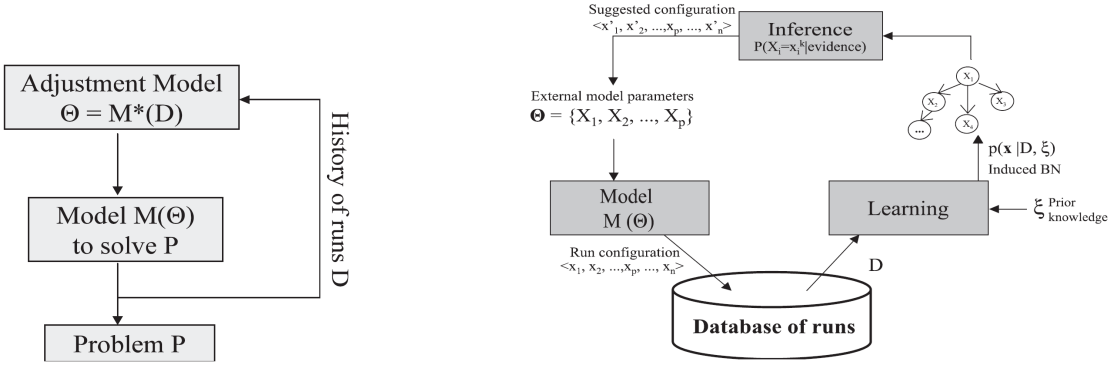


FIGURE 1.13 – À gauche, le méta-modèle M^* permettant l'ajustement des paramètres du modèle M. À droite, le schéma général du modèle proposé (d'après [Pavón et al., 2008]).

et une configuration bas niveau est automatiquement construite à partir des *promises* correspondant aux *aspects* spécifiés. Ces travaux ont été appliqués à la gestion de configuration de serveur webs (Fig. 1.12), pour lesquels ils permettent d'apporter une première solution pour maîtriser la complexité de configuration [Schwartberg & Couch, 2004].

La problématique centrale de ces travaux est de garantir la cohérence de la configuration en fournissant des outils permettant de limiter les erreurs d'administration. L'approche est centrée sur l'application cible. Cependant, comme nous le verrons dans la section 2.2, la démarche que nous utilisons présente des similarités avec celle-ci : définir le comportement du système à travers l'application de contraintes sur ses entrées. Notons que les notions d'agents et de comportements que nous utilisons restent éloignées, et que les questions de variété, centrales dans notre cas, sont ici absentes. En effet, les configurations du système doivent impérativement rester dans un état stable.

1.3.2 Génération de paramétrage de modèles complexes

Dans des travaux récents, Pavón et al. abordent la question de la génération automatique de paramétrage [Pavón et al., 2008]. Leur objectif est de fournir un outil d'aide au paramétrage de modèles complexes, comme des réseaux de neurones ou des algorithmes génétiques, qui fournisse un résultat au moins aussi performant que celui qui pourrait être réalisé par l'utilisateur, tout en lui évitant cette tâche. Pour ce faire, ils proposent l'utilisation d'un méta-modèle qui génère des recommandations sur le meilleur paramétrage à utiliser pour le modèle considéré. Leur approche se veut générale, dans le sens où elle peut s'appliquer à tout modèle, et automatisée, car son exécution ne nécessite pas d'intervention de l'utilisateur.

Le méta-modèle réalise une estimation du paramétrage ciblé à partir d'une base contenant les résultats antérieurs du modèle (Fig. 1.13). Le calcul des nouvelles solutions repose sur l'utilisation de réseaux bayésiens, du fait de leur capacité à d'extraire facilement la connaissance depuis des bases de données existantes. Dans le réseau utilisé, les nœuds correspondent aux paramètres que l'on souhaite mettre à jour, et les arcs représentent des relations de probabilité entre eux. La manière dont ces relations sont obtenus n'est toutefois pas précisée. Une fois ces éléments établis, le réseau infère une nouvelle solution qui constituera le paramétrage du modèle cible lors de sa prochaine itération.

Cette approche est appliquée aux solveurs de contraintes géométriques. Dans ce contexte particulier, l'utilisation d'algorithmes génétiques est une solution possible pour accélérer la résolution du problème. Cependant, le paramétrage de ces algorithmes est complexe : il faut définir la fonction objectif, la taille de la population, le mécanisme de sélection, ou encore le taux de mutation. Les valeurs de ces paramètres influençant fortement la qualité de la solution, il est intéressant de les optimiser. Le méta-modèle proposé est appliqué dans ce but, et permet d'obtenir des résultats de qualité comparable à la référence du domaine. Il présente de plus l'avantage de ne pas nécessiter de supervision, et devient de plus en plus précis avec la taille de la base de données.

L'approche proposée par les auteurs est particulièrement intéressante par les aspects d'automatisation et de généricité qu'elle vise. Cependant, la conception du mécanisme reste influencée par le champ d'application, et la démarche est peu transposable à des approches ne mettant pas en jeu spécifiquement des algorithmes complexes. Le paramétrage d'agents par exemple ne peut pas être pris en charge directement. Il faudrait pour cela traduire les relations de probabilités entre les différents paramètres de l'agent et utiliser un méta-modèle par agent. Cela limite l'intérêt de l'approche pour ce type d'application.

1.3.3 Variété et simulation de foules

Comme nous l'avons vu plus haut, la simulation de foules d'humains virtuels est utilisée pour la formation, la thérapie de phobies ou la conception architecturale. Afin d'augmenter l'immersion des utilisateurs et la validité des résultats, le réalisme de la simulation est primordial. La variété des agents présents contribuant largement à ce réalisme, des techniques destinées à l'augmenter ont été développées.

Ce contexte est cependant soumis à des problématiques importantes en terme de performances. En réalité virtuelle, le système doit être temps réel : afin de préserver des temps de calcul acceptables, d'au minimum 25 images par seconde, le calcul des images doit être suffisamment rapide. Par ailleurs, des limitations de mémoire entrent rapidement en jeu pour les calculs graphiques. Même si les cartes embarquent aujourd'hui jusqu'à 768 Mo de mémoire, cette limite est très vite atteinte si le nombre et la précision des modèles 3D d'agents présents dans la simulation sont importants. Enfin, la conception des modèles 3D eux-même est une limite : au vu du volume de travail demandé pour leur création, il n'est pas envisageable de créer des centaines de modèles différents pour intégrer de la variété dans la simulation.

Différentes techniques sont donc utilisées afin de faire varier automatiquement l'apparence d'humains virtuels, tout en prenant en compte ces problématiques [Maim, 2009]. Elles mettent en jeu l'utilisation de variations automatisée de couleurs, d'accessoires, et de formes (Fig. 1.14). Elles permettent de créer des foules cosmopolites à partir de peu d'éléments, permettant ainsi de limiter les temps de calcul et la charge mémoire.

La première technique utilisée repose sur la variation des couleurs des textures appliquées sur les modèles d'agents. Elle nécessite une étape préalable de définition des parties physiques de l'agent dont les couleurs peuvent varier : vêtements, visage, cheveux... Un jeu de couleur est ensuite sélectionné aléatoirement, puis appliqué sur le modèle. Cela permet de créer une diversité visuelle pour un coût de calcul et de mémoire restreint : une palette couleur occupe peu de mémoire, et peut être utilisée sur plusieurs templates visuels différents (un homme et un enfant par exemple), et même sur des accessoires. Les couleurs étant tirées aléatoirement, certains des coloris et des compositions de couleur manquent cependant nettement de « goût ». Pour



FIGURE 1.14 – À partir du même template graphique, l'application de couleurs de peau et d'accessoires différents (vêtements, lunettes, cheveux...) permet d'obtenir une grande diversité visuelle pour un coût de calcul et de mémoire restreint (d'après [Maim, 2009]).

contourner cette limitation, la palette de couleur est restreinte dans des intervalles de variation. Par exemple, dans les dimensions TSV [7], la teinte est restreinte de 20 à 250, la saturation de 30 à 80 %, et la luminosité de 40 à 100 %.

L'autre type de variation concerne la forme des agents, et passe par deux vecteurs : l'ajout d'accessoires sur les agents, et la modification de leur corpulence. Les accessoires peuvent être simples, i.e. ne pas nécessiter de modification de l'animation de l'agent (un chapeau, des lunettes par exemple), ou complexes (le port d'un sac modifie la dynamique du bras, voire la posture du fait du poids du sac). Les accessoires sont répartis en différentes catégories, afin d'empêcher l'instanciation d'objets similaires sur un même agent (deux chapeaux par exemple). Pour chaque agent, le modèle contient, pour chacune des catégories, la liste des accessoires qui peuvent être attachés et le point d'attache. Lors de la création de l'agent, les accessoires sont attribués aléatoirement aux différents emplacements.

La modification de la corpulence requiert une modification du modèle 3D de l'agent, en partie intégrée dans les outils de modélisation graphique comme 3DSMax par les *FatMap*. Elle repose ensuite sur des algorithmes faisant croître ou diminuer le modèle en fonction de certaines dimensions (les épaules et la tête par exemple ne changent pas dans les même proportions), et permet de générer de nombreuses variations à partir d'un unique modèle 3D .

L'utilisation de techniques simples en apparence permet donc d'augmenter énormément la variété des agents dans la simulation, à travers la dimension visuelle. En ce qui concerne le comportement, seule l'intégration d'animations liées au port d'un accessoire particulier sera cependant modifiée par ces éléments. Le modèle comportemental proprement dit ne bénéficie pas de telles améliorations. La variété y est prise en compte en forçant les agents à ne pas suivre exactement le même chemin (génération aléatoire de waypoints sur le trajet suivi), et en les faisant se déplacer en petits groupes de 2 à 5 personnes utilisant un paramétrage similaire. Notons finalement que les mécanismes de génération de variété pourraient profiter d'une approche unifiée : par exemple, la gestion des accessoires est intégrée directement dans le modèle 3D de

7. TSV : Teinte (le type de la couleur, de 0 à 360), Saturation (l'« intensité » de la couleur, de 0 à 100 %), Valeur (la « brillance » de la couleur, de 0 à 100 %)

l'agent. Le contrôle de la cohérence (par exemple dans les palettes de couleurs) est intégré en dur, tout comme celui des accessoires (une liste fixé d'accessoires est définie pour chaque modèle 3D). Un mécanisme complémentaire gérant ces aspects pourrait apporter plus de flexibilité à ce niveau.

1.3.4 Des personnalités virtuelles pour les conducteurs

Wright et al. [Wright et al., 2002] se sont intéressés à l'introduction de personnalités virtuelles pour les conducteurs simulés, afin d'améliorer l'immersion des conducteurs réels en simulation de conduite. À la différence de notre objectif, ils ne s'intéressent cependant pas à la facilité de scénarisation ou à la transposition dans des contextes culturels variés. L'approche est appliquée à la modélisation du choix de la vitesse des véhicules, mais pourrait selon eux être étendue facilement à d'autres sous-modèles de la tâche de conduite, comme les changements de voie ou le suivi de véhicule.

Le modèle proposé est construit en quatre étape successives, permettant de créer des véhicules autonomes ayant des caractéristiques comportementales diverses, associées à des personnalités virtuelles. La première étape sélectionne une distribution de probabilité offrant une approximation raisonnable des choix de vitesse rencontrés dans la réalité. Les auteurs se basent sur des données provenant de la littérature, ce qui aboutit à l'utilisation d'une distribution normale. Cette distribution est combinée à une fonction introduisant de la variabilité : la variance de la distribution est fonction de la vitesse moyenne, ce qui élargit la courbe aux basses vitesses.

La seconde étape permet de prendre en compte les éléments contextuels comme les limites de vitesse, la courbure de la route et la distance à l'intersection, éléments qui influencent le choix de la vitesse.

La troisième étape intègre les paramètres caractérisant la personnalité virtuelle des conducteurs, à partir de données extrapolées depuis des observations expérimentales. Cinq paramètres comportementaux sont pris en compte : le sexe, l'âge, l'agressivité, l'intoxication par l'alcool et la fatigue. Ils sont utilisés comme des correcteurs sur la distribution normale définie à la première étape. Par exemple, les hommes ont tendance à conduire plus vite que la moyenne, ce qui se traduit par l'utilisation d'un correcteur sur la moyenne de la distribution normale d'un conducteur masculin : $\mu = 1.072\,\mu$. Notons que les différents paramètres peuvent se compenser : après l'application des correcteurs, un homme prudent et une femme agressive utiliseront peut-être des paramètres similaires.

Enfin la dernière étape génère la valeur utilisée par le conducteur, à partir de la distribution normale construite dans les étapes précédentes. Afin de conserver une certaine homogénéité dans le choix de la vitesse, le nouveau choix tiré aléatoirement est combiné aux vitesses des pas de temps précédents (Fig. 1.15).

Le modèle proposé est appliqué sur le simulateur de conduite de l'Université de Leeds. Il permet de générer des valeurs différentes suivant les profils conducteurs choisis, et répond donc à l'objectif d'introduire de la variabilité et des personnalités virtuelles dans la simulation. Une expérimentation est par ailleurs menée afin d'évaluer si les utilisateurs du simulateur perçoivent les différences de personnalité. Des utilisateurs sont ainsi invités à observer le déroulement de la simulation, sans être en situation de conduite. Il sont capables de distinguer des comportements différents, mais les qualificatifs utilisés ne correspondent pas aux paramètres introduits. Les observateurs reconnaissent en fait uniquement trois types de comportement : prudent, normal et

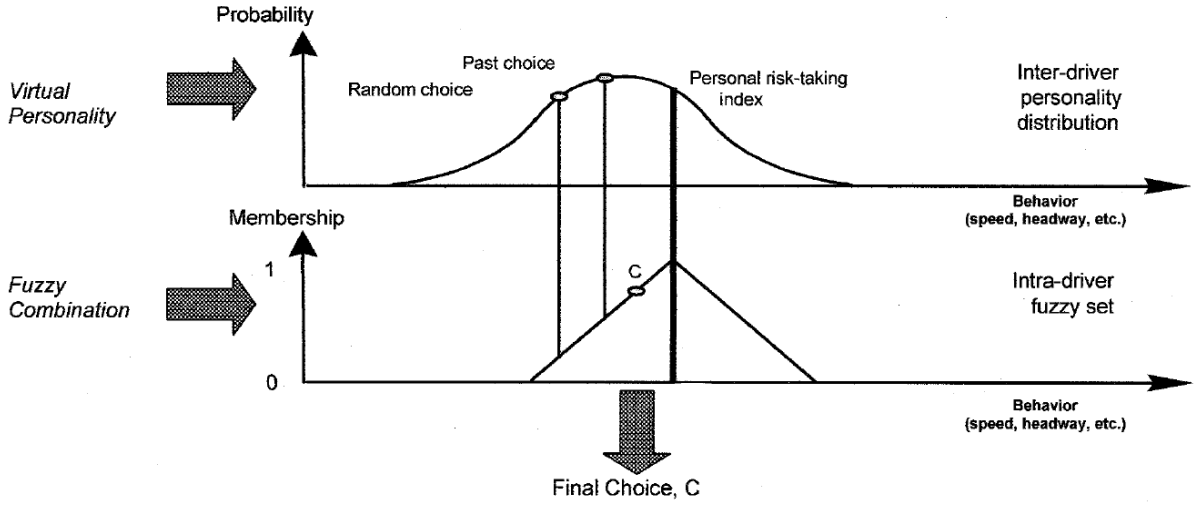


FIGURE 1.15 – Les distributions utilisées dans le modèle de choix de la vitesse, introduisant des caractéristiques comportementales et de la variabilité (d'après [Wright et al., 2002]).

agressif. En situation de conduite, le nouveau trafic est par ailleurs qualifié de « plus réaliste » par les conducteurs.

L'approche proposée permet donc d'introduire avec succès des personnalités virtuelles pour les conducteurs, ce qui améliore le réalisme perçu par les observateurs de la simulation. Cependant, la technique utilisée présente plusieurs limites. Tout d'abord, elle nécessite de remettre en cause l'intégralité du modèle de trafic existant, car chaque sous-modèle de la tâche de conduite doit être modifié. Cela suppose d'avoir un contrôle total sur le code source de l'outil, ainsi que de maîtriser les risques de régression logicielle. Par ailleurs, l'approche est spécifique à l'application ciblée, et ne peut être transposée dans d'autres contextes. Enfin, elle n'est pas flexible : toute modification du paramétrage doit passer par une modification du modèle, ce qui ne permet pas de prendre en compte des contextes de trafic variés. Par exemple, des utilisateurs italiens devraient reprendre l'approche dans sa totalité pour bénéficier d'apports similaires.

1.4 Conclusion

Dans ce premier chapitre, nous nous sommes intéressés au contexte de notre étude. Nous avons tout d'abord présenté le domaine de notre application principale, la simulation de trafic dans les simulateurs de conduite. Si la simulation de trafic est abordée dans des domaines variés, comme l'ingénierie du trafic ou la psychologie de la conduite, le cadre de la simulation de conduite entraîne des contraintes spécifiques. Il est en effet nécessaire de prendre en compte les aspects liés à la scénarisation et au réalisme comportemental des véhicules. Nous avons également présenté le fonctionnement général d'un simulateur de trafic appliqué aux simulateurs de conduite, de la représentation de l'environnement aux caractéristiques conducteur prises en compte. Plusieurs outils existants ont été décrits ; dans le Chapitre 4, nous reviendrons sur la suite logicielle développée par Renault, SCANeR™.

SCANeR™ étant conçu suivant une architecture multi-agents, nous avons ensuite présenté les spécificité de ce domaine et de l'approche centrée individus. Les caractéristiques de ces sys-

tèmes, ainsi que leurs domaines d'application ont été rappelés, ainsi que les principaux éléments les constituant : les agents, l'environnement, les interactions et l'organisation. L'intérêt de la simulation centrée individus a également été rappelé, en particulier ses capacités explicatives et sa flexibilité. Deux exemples d'application sont présentés plus en détails : l'utilisation de comportements réactifs pour l'animation d'agents, et la simulation de foules.

Enfin, la dernière partie de ce chapitre décrit différents travaux connexes à notre approche. La génération de comportements variés et cohérents a en effet été abordée suivant plusieurs axes : gestion automatisée de configurations systèmes, génération automatisée du paramétrage de modèle complexes, introduction de variété dans les simulations de foules, ou encore utilisation de personnalités virtuelles pour augmenter le réalisme du trafic en simulation de conduite. Cependant, ces différentes approches ne permettent pas de répondre à notre besoin. Suivant les cas elles sont très spécifiques à leurs domaine, ou n'offrent pas les facilités de configuration que nous recherchons. Aucune ne prend en compte simultanément les questions de variété et de cohérence, tout en offrant la flexibilité recherchée.

Chapitre 2

Une description normative

Ce second chapitre présente les systèmes normatifs et l'application que nous en faisons dans ce travail. Dans un premier temps, nous nous intéressons à la notion de norme, et présentons son origine et ses applications dans les systèmes multi-agents. Nous présentons ensuite le cadre formel de notre approche : décrire le comportement des agents par une sémantique basée sur les normes. Cette utilisation permet de définir à la fois les comportements de référence et les écarts tolérés par rapport à eux, la conformité étant contrôlée à travers la notion de violation. Enfin, nous décrivons les algorithmes utilisés pour la création et le contrôle de ces comportements, et montrons comment ils nous permettent de répondre aux besoins en obtenant à la fois variété et cohérence.

2.1 Les approches normatives

Cette première partie introduit les notions de norme et d'approche normative. Dans un premier temps, nous nous plaçons dans une perspective historique, en décrivant l'origine de ces concepts ainsi que l'utilisation qui en est faite dans différents domaines. Ensuite, nous présentons pourquoi et comment ils ont été transposés naturellement dans le contexte des systèmes multi-agents. Nous terminons enfin en illustrant cette démarche par des cas d'application concrets.

2.1.1 Introduction

Suivant le contexte, la notion de norme prend différentes significations. Nous présentons ici la définition retenue dans notre cas, puis les approches historiques qui ont conduit à sa formalisation.

2.1.1.1 Définitions

Le concept de norme est utilisé dans différents domaines. Communément, il répond aux définitions suivantes [8] :

1. Type concret ou formule abstraite de ce qui doit être, en tout ce qui admet un jugement de valeur : idéal, règle, but, modèle suivant les cas.

8. Le Grand Robert de la Langue Française, Édition 2007.

2. État habituel, ordinaire, régulier, conforme à la majorité des cas.
3. Formule qui définit un type d'objet, un produit, un procédé technique en vue de simplifier, de rendre plus efficace et plus rationnelle la production dans un secteur économique donné.
4. Ce qui dans la parole, le discours, correspond à l'usage général (sens objectif) ; usage d'une langue valorisé et considéré comme préférable (sens prescriptif).
5. Manière de faire, de se comporter ou de penser, souvent majoritaire, socialement définie et sanctionnée, selon un système de référence implicite (idéologie, valeur) ou explicite (règle, droit, loi).
6. Norme d'un vecteur, nombre réel correspondant à sa mesure.

Dans notre cadre, la définition adaptée est la définition 5, qui correspond à l'acceptation communément utilisée dans le cadre des systèmes multi-agents [Boella et al., 2007]. Celle-ci s'appuie sur le comportement au sens de « manière de faire, de se comporter ou de penser ». Notons que nous considérons ce terme au sens large : la norme pourra concerner tout élément lié à l'expression finale du comportement, du bas niveau (la valeur d'un paramètre, comme « vitesse = 100 km/h ») au haut niveau (avec une sémantique plus riche, comme « aller plutôt vite »). Par ailleurs, le comportement que nous considérons n'est pas nécessairement majoritaire : cela permet de prendre en compte des comportements communs à un nombre restreint d'individus que l'on souhaite pourtant intégrer en tant que norme sociale. La norme doit également avoir une définition sociale, les autres membres de la société sont donc en mesure de la percevoir. Enfin, l'obéissance ou la désobéissance à la norme doivent être sanctionnés, soit implicitement, soit explicitement, et la société inclut donc des systèmes de régulation réalisant cette fonction.

Notons finalement que deux sources de représentations normatives sont généralement considérées : les premières correspondent aux coutumes, les secondes à la législation, qui distinguent les normes sociales et les normes légales. La principale différence entre ces deux groupes est la manière dont les normes sont appliquées : le respect des premières n'est pas basé sur la punition, contrairement aux secondes [Conte et al., 1999b].

2.1.1.2 Approches historiques

La formalisation de la notion de norme trouve son origine dans différents domaines, en particulier les sciences juridiques et les sciences sociales. Dans le cadre des sciences juridiques, les lois ont pour but principal de fournir des règles destinées à restreindre ou coordonner le comportement des humains : des agents autonomes. Ces règles, sous forme d'obligations, d'interdictions ou de permissions, peuvent être décrites sous forme de normes.

Les premières définitions de la notion de norme remontent ainsi au début du vingtième siècle, lors de la formalisation des concepts juridiques [Hohfeld, 1913]. La démarche visait à construire une théorie du droit, en particulier sur les aspects de jurisprudence. Dans les années cinquante apparaît la logique déontique [von Wright, 1951] : son objectif est d'étudier les relations logiques entre les obligations et les permissions. Elle peut donc être utilisée pour représenter les normes et raisonner sur elles. Sur cette base, Kanger [Kanger, 1971], puis Lindhal [Lindhal, 1977], construisent la théorie des positions normatives, qui permet d'obtenir une spécification complète des relations juridiques existant entre deux personnes à partir d'un ensemble incomplet de besoins. Certains auteurs s'intéressent par ailleurs aux processus d'adhésion et de propagation des normes, et cherchent à expliquer les mécanismes d'acceptation par les agents, c'est-à-dire les éléments qui les conduisent à considérer la norme comme légitime [Raz, 1975].

Si les différentes écoles de théorie juridique utilisent une définition commune de la norme, elles ne s'accordent cependant pas sur les raisons qui conduisent à l'acceptation et l'obéissance. Différentes raisons sont ainsi proposées [Conte et al., 1999b] :

- les normes sont acceptées par peur de l'autorité,
- les normes sont acceptées car elles sont rationnelles,
- les normes sont acceptées par sens du devoir,
- les normes sont acceptées car elles résolvent des problèmes de coordination et de coopération.

Notons que le champ juridique reste aujourd'hui un terrain d'application largement utilisé par les théories normatives, en particulier du fait de la formalisation intrinsèque du domaine et de la possibilité de s'appuyer fortement sur la logique pour décrire les problématiques traitées.

Dans le contexte des sciences sociales, les normes sont utilisées pour expliquer les relations entre les membres de la communauté, ainsi que les comportements attendus dans la société. Par exemple, Tuomela [Tuomela, 1995] distingue les règles et les normes sociales (r-norms et s-norms). Les règles sont basées sur la notion d'autorité et sont associées à des sanctions, formelles ou informelles. Les normes sociales concernent les conventions qui apparaissent au sein des communautés. D'autres types de normes permettent de décrire des normes auxquelles les agents obéissent pour des raisons personnelles, comme les normes morales et les normes de prudence (m-norms et p-norms). Finalement, les agents obéissent à ces différents types de normes pour des raisons distinctes : les règles car elles sont acceptées, les normes sociales parce que les autres attendent qu'on s'y conforme, les normes morales à cause de la conscience individuelle, et les normes de prudence car c'est la chose rationnelle à faire. En sociologie, la distinction entre un ordre normatif non-institutionnalisé (les normes sociales, morales, personnelles dans la vie quotidienne) et les institutions est ainsi formalisé dans les approches de recherche [Therborn, 2002].

L'utilisation des normes dans les contextes juridiques et sociaux a ainsi conduit à des formalisations des concepts dans le cadre de la logique. Ces travaux, ainsi que l'intérêt intrinsèque de la notion, ont conduit à son extension dans d'autres domaines. L'un d'entre eux est celui des systèmes multi-agents.

2.1.2 Normes et systèmes multi-agents

Nous présentons d'abord l'intérêt des normes dans ce contexte, avant d'aborder deux de leurs applications : les systèmes multi-agents normatifs, et l'intégration d'obligations dans certains modèles d'agents. Enfin, nous présentons les questions relatives aux violations, aux déviations et aux conflits.

2.1.2.1 Intérêt des normes dans les approches centrées individus

L'utilisation des normes dans les systèmes multi-agents provient originellement de problématiques de coordination entre les agents, au niveau des actions ou au niveau de la communication [Verhagen, 2000]. Pour répondre à ce type de question, un système purement centralisé peut être utilisé : d'architecture simple, il présente le désavantage d'être soumis à des coûts de calculs prohibitifs dès que sa taille augmente. Une autre possibilité est d'utiliser une approche purement décentralisée, dans laquelle les agents résolvent les conflits au fil de l'eau. Dans ce second cas, c'est le coût de résolution des conflits qui peut être limitant : leur fréquence est potentiellement très élevée si aucun mécanisme de coordination n'est présent. Afin de limiter le

nombre de conflits sans centraliser l'ensemble du processus, des approches intermédiaires ont été proposées.

L'une de ces approches repose sur l'utilisation des normes. En effet, les systèmes multi-agents et la sociologie cherchent tous deux à expliquer le lien entre les comportements microscopiques et les effets macroscopiques qui en résultent. Les normes, en tant que clé du comportement humain en société, ont donc naturellement inspiré les concepteurs d'agents artificiels. Elles présentent une solution intéressante pour des problématiques comme la régulation de sociétés artificielles [Savarimuthu et al., 2008], l'organisation dans les systèmes multi-agents [Vázquez-Salceda et al., 2005], l'amélioration de la coordination, les institutions électroniques [García-Camino et al., 2009] ou encore les communications entre agents [Esteva et al., 2002].

Leur principal atout face à d'autres méthodes destinées à gouverner le comportement des agents est qu'elles permettent de réguler non seulement leurs interactions, mais aussi leurs actions. De plus, elles présentent un champ d'application plus étendu car elles ne sont pas limitées à deux agents seulement : elles peuvent s'appliquer à un agent, un groupe d'agents ou même tous les agents.

Les normes peuvent être appliquées de différentes manières dans les systèmes multi-agents. L'une d'entre elles est de les utiliser comme contraintes sur les actions des agents : ceux-ci sont alors conçus pour agir individuellement, mais leurs possibilités d'action sont restreintes afin de limiter l'apparition de conflits [Moses & Tennenholtz, 1995]. Le mécanisme de coordination, i.e. la contrainte des actions, est intégré dans les agents lors de la conception de la simulation. La modification des contraintes ne peut être réalisée qu'off-line.

La représentation des normes a évolué suivant différents niveaux, qui peuvent être classifiés de la façon suivante [Boella et al., 2008]. Au niveau 1, que nous venons de voir, les normes sont imposées par le concepteur et automatiquement respectées. Le niveau 2 désigne un système possédant une représentation explicite des normes ; celles-ci peuvent être utilisées dans la communication ou la négociation entre les agents, et des formes simples d'organisations et d'institutions peuvent être créées. Au niveau 3, les normes peuvent être manipulées : les agents peuvent en ajouter ou en supprimer suivant les règles du système normatif. Au niveau 4, les normes définissent la réalité sociale, en tant qu'éléments structurants de la société des agents elle-même. Enfin au niveau 5, les normes créent une nouvelle réalité morale, en évoluant du droit vers des problématiques éthiques. Aujourd'hui, les systèmes existants font partie des niveaux 1 à 3, et les recherches s'intéressent au niveau 4. Le niveau 5 est le sujet de travaux à plus long terme.

2.1.2.2 Systèmes multi-agents normatifs

L'utilisation avancée des normes nécessite en fait que les agents en aient une représentation mentale, sur laquelle ils puissent raisonner [Conte et al., 1999b]. Pour ce faire, les normes doivent exister en tant qu'entités manipulables, afin de permettre aux agents de communiquer sur elles, de percevoir ce qu'est l'autorité normative, et de résoudre des conflits potentiels entre les normes. C'est la prise en compte de ces éléments qui a conduit au développement des systèmes d'agents normatifs.

La représentation des normes est donc externalisée, et elles sont utilisées pour réguler les actions des agents afin de gouverner leurs comportements : les actions peuvent être interdites, permises ou encore obligatoires. Deux aspects importants doivent être pris en compte : le premier concerne la manière dont les agents peuvent acquérir les normes ; le second celle dont ils peuvent

les violer. Avec l'acquisition des normes, les agents deviennent capables de constituer leur propre référence. Le système, pourvu de mécanismes permettant de créer, d'adhérer et de refuser des normes, limite les aspects de conception. Il devient adaptatif, de nouvelles normes pouvant apparaître et être prises en compte par les agents. Le mécanisme de violation permet aux agents de choisir parmi les normes du système celles qu'ils vont respecter. Ils peuvent ainsi gérer eux-même leur système normatif, et le faire évoluer en fonctions de leurs besoins [Conte et al., 1999a]. La violation leur permet également de résoudre les conflits entre les normes auxquelles ils adhèrent. Les agents ont ainsi le choix de rejeter les standards proposés par les normes, afin de préserver leur autonomie ; ce rejet repose sur le modèle interne de l'agent, en fonction de son raisonnement, de ses objectifs et de ses attentes propres.

De nombreuses études se focalisent sur les aspects théoriques et formels des normes, en particulier dans le contexte de la logique déontique [Dignum, 1999, Verhagen, 2000, Grossi, 2007]. Plus récemment, d'autres travaux se focalisent sur l'implémentation pratique de ces systèmes [Vázquez-Salceda et al., 2004, García-Camino et al., 2005, Cranefield, 2007]. Spécification et implémentation des normes peuvent par ailleurs être associées et reliées par un mécanisme de génération automatique [Torres da Silva, 2008]. Le concepteur spécifie les normes dans le langage proposé, l'implémentation associée étant générée automatiquement. Cette implémentation est alors utilisable par les agents et par un système de gouvernance, qui est en mesure d'activer et de désactiver les normes, de déterminer les violations et d'informer les agents des punitions et récompenses.

Comme nous l'avons vu, les normes peuvent être abordées suivant différentes perspectives. La vue légaliste est une approche « top-down », qui considère que le système normatif est un instrument de régulation du comportement des agents sans contrainte directe sur les agents. Cependant, les agents sont souvent motivés par des sanctions, plutôt que par le réel partage de normes communes. La vue interactioniste qui émerge aujourd'hui, est une vue « bottom-up ». Les normes y sont perçues comme des régularités de comportement qui émergent sans contrainte du système, simplement parce que les agents s'y conforment sur la base de buts ou de valeurs communs, ou par un sentiment d'appartenance au groupe. Les sanctions formelles ne sont pas toujours nécessaires, le risque d'exclusion ou de mise à l'écart du groupe étant suffisant pour permettre l'adhérence aux normes.

L'évolution rapide du domaine se reflète dans la définition du « système multi-agents normatif ». Ainsi, en 2006, une définition adoptée par la communauté [9] [Boella et al., 2006] était :

> « A normative multiagent system is a multiagent system together with normative systems in which agents on the one hand can decide whether to follow the explictly represented norms, and on the other the normative systems specify how and in which extent the agents can modify the norms »

En 2008, lors du même workshop, la définition adoptée était [Boella et al., 2008] :

> « A normative multiagent system is a multiagent system organized by means of mechanisms to represent, communicate, distribute, detect, create, modify, and enforce norms, and mechanisms to deliberate about norms and detect norm violation and fulfilment »

La problématique centrale s'est déplacée de la représentation des normes vers les mécanismes utilisés par les agents pour se coordonner, et d'une manière plus générale pour organiser le

9. Définition issue des discussions du workshop Norms in Multi-Agent Systems, NorMAS [Boella et al., 2006, Boella et al., 2008].

système. Cela ouvre en particulier la voie à l'utilisation des normes dans les systèmes ouverts, où les agents peuvent entrer dans un environnement dont ils ne connaissent pas les règles, ou à l'émergence des normes, où les agents doivent les découvrir afin de communiquer pour atteindre leurs buts.

2.1.2.3 Normes et agents

Dans une autre perspective, des évolutions de modèles classiques d'agents ont été proposées afin de prendre en compte les apports des normes. Le modèle BDI [10] est un modèle d'agents rationnels distinguant les attitudes mentales de l'agent [Rao & Georgeff, 1995] : les croyances (Beliefs) représentent l'état des informations disponible pour l'agent, c'est-à-dire ses croyances sur l'état du monde ; les Désirs représentent l'état motivationnel de l'agent, c'est-à-dire les objectifs ou les situations qu'il souhaiterait atteindre ; enfin les Intentions représentent l'état délibératif de l'agent, c'est-à-dire ce qu'il a décidé de faire. Ce modèle est en particulier utilisé lorsqu'il est important de pouvoir distinguer l'état motivationnel de l'agent. Le processus de délibération d'un agent BDI implique différentes étapes successives : surveiller les évènements extérieurs, identifier les changements de l'environnement auxquels il faut répondre, générer des buts, sélectionner un ou plusieurs buts comme intentions, mettre en œuvre le plan choisi, puis poursuivre la surveillance de l'environnement.

Devant l'intérêt des normes et des obligations comme mécanisme de coordination des agents, Dignum et al. [Dignum et al., 2000] ont proposé une extension du modèle BDI intégrant les obligations. Pour ce faire, ils ont étendu le processus de délibération originel des agents BDI : une étape supplémentaire est ajoutée entre l'identification des évènements extérieurs et la génération des buts. Cette étape est destinée à analyser les évènements observés au regard des normes et des obligations auxquelles est soumis l'agent. Des actions particulières peuvent y être associées, et l'agent a ainsi la possibilité de répondre ou non à ces éléments, en fonction de ses préférences.

Dans [Broersen et al., 2001], Broersen et al. introduisent une architecture d'agent intégrant la notion de normes. Dans ce but, un composant complémentaire est introduit dans le modèle BDI : les Obligations. Celles-ci permettent de prendre en compte tous les aspects normatifs au niveau de l'architecture, et intégrent ainsi des règles sociales au sein des agents. Les violations sont possibles, ce qui permet de respecter l'autonomie des agents : par exemple, un désir plus fort qu'une obligation conduira à une violation. Les violations permettent également de résoudre les conflits entre obligations : si j'ai l'obligation d'être honnête, et que j'ai l'obligation d'être poli, un conflit est possible (être honnête peut me conduire à être impoli...). L'architecture BOID a été développée pour aider les agents à résoudre ce type de conflit.

Deux apports distincts caractérisent l'approche proposée. La première est de donner une priorité différente à chacun des composants B, O, I, D. En fonction de la valeur de la priorité attachée à un composant, une certaine attitude mentale est privilégiée. Différents types de comportements peuvent ainsi être obtenus : un agent sera réaliste si les croyances sont privilégiées (B > O, I, D), il sera égoïste si les désirs sont prioritaires (D > B, O, I), ou encore social s'il privilégie les obligations (O > B, I, D)... Par exemple un agent BOID sera ainsi qualifié de réaliste social, les croyances étant prioritaires sur tous les autres composants, et les obligations étant prioritaires sur les désirs (B > O > I > D). Enfin, le second apport principal du modèle est l'utilisation de boucles de contrôle dynamique, qui renvoient les éléments dérivés en tant que nouvelles entrées au BOID, et peuvent déclencher de nouvelles obligations, croyances, désirs,

10. Beliefs – Desires – Intentions

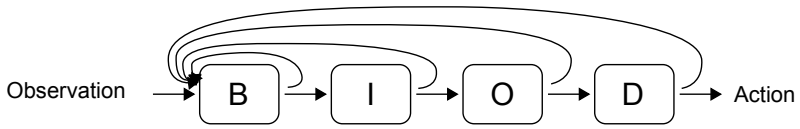


FIGURE 2.1 – Un agent BOID « social simple d'esprit » : social, car les obligations passent avant les désirs ; simple d'esprit, car les intentions passent avant les obligations et les désirs.

ou même intentions (Fig 2.1). Par exemple, l'obligation d'assister une personne en danger peut induire l'obligation d'alerter les secours.

2.1.2.4 Déviations, violations, conflits

La notion de norme est par ailleurs intrinsèquement liée à celle de violation. En effet, la norme étant le comportement de référence, les agents ne se conformant pas à cette référence sont violateurs. Notons que la norme peut être restreinte à un petit groupe d'agents : il reste vrai que tous les autres sont en violation par rapport à la norme considérée. Cette identification permet aux autres membres de mon groupe, ou à la société, de réagir à ces écarts de comportement.

A ce stade, il est important de distinguer deux notions : le comportement déviant, et le comportement violateur. Un comportement violateur est un comportement qui ne respecte pas la norme. Un comportement déviant est un comportement dont certaines caractéristiques peuvent le conduire à être en violation. Autrement dit, un comportement déviant est potentiellement violateur. Dans le monde réel, tout agent humain est déviant : sans même parler de règle sociale, chacun peut être amené à enfreindre une loi par choix ou par inadvertance. Dans le monde informatique, les choses sont différentes. Suivant la manière dont le système est conçu, il est possible que certaines lois ou certaines normes soient inviolables, simplement parce que la simulation, telle qu'implémentée, ne le permet pas. On parle dans ce cas de normes régimentées [Jones & Sergot, 1993]. Un agent obéissant uniquement à des normes régimentées n'est pas déviant.

Les violations constituent un point important des systèmes normatifs, pour plusieurs raisons. Tout d'abord, elles permettent de résoudre les conflits entre normes. Il est en effet possible que les agents se retrouvent confrontés à un ensemble d'obligations et d'interdictions qui leur rendent impossible toute action s'ils ne violent pas certaines d'entre elles. La violation permet à l'agent de ne pas respecter certaines d'entre elles, afin de ne pas se retrouver bloqué. La sélection des normes pouvant être violées est d'ailleurs une problématique à part entière. Il est par exemple possible de résoudre ce type de problème en associant des coûts aux violations : les agents satisfont une fonction d'utilité qui les conduit à violer les règles les moins importantes [Barbuceanu, 1998]. Une telle approche permet de faire en sorte que les obligations et les interdictions les plus importantes soient toujours respectées, même s'il faut pour cela en violer certaines autres.

D'autre part la possibilité de violation est garante de l'autonomie des agents. En effet, si les agents doivent systématiquement respecter toutes les règles du système, ils n'ont plus de marge de manœuvre leur permettant de s'adapter à des situations nouvelles. Les violations, ponctuelles ou systématiques, sont donc tout à fait pertinentes dans les simulations afin d'enrichir les possibilités d'action [Castelfranchi et al., 2000]. L'utilisation d'agent violant délibérément les normes, ou mentant afin de ne pas se faire punir, est ainsi parfois envisagée [Torres da Silva, 2008].

2.1.3 Exemples d'applications

Les systèmes normatifs sont utilisés dans le cadre d'applications variées, comme la description des mécanismes de marchés financiers [Michael et al., 2005], ou encore les ventes aux enchères [García-Camino et al., 2005]. Dans cette partie, nous nous intéressons à deux exemples particuliers, illustrant les types d'application des normes introduits plus haut : la conception hors-ligne d'une part, et l'utilisation dans le cadre de systèmes multi-agents ouverts et adaptatifs.

2.1.3.1 Des règles sociales pour optimiser la navigation

Shoham & Tennenholz [Shoham & Tennenholtz, 1995] s'intéressent à l'utilisation de lois sociales pour améliorer la navigation de robots en milieu ouvert. Le problème est posé de la manière suivante : un nombre m de robots évoluent dans un grille de taille $n.n$. Chacun d'entre eux occupe une case de la grille, et le système doit interdire les collisions. A chaque pas de temps, les robots peuvent soit se déplacer d'une case en utilisant de l'énergie, soit rester immobile. Ils doivent atteindre la destination qui leur est fixée en minimisant le temps et l'énergie consommée. Ils peuvent avoir plusieurs destinations à visiter.

Un système centralisé est considéré comme ne pouvant monter en charge suffisamment pour gérer l'ensemble du système. Il s'agit donc ici de déterminer un bon chemin pour chacun des agents en se basant sur des mécanismes décentralisés utilisant uniquement les communications d'agent à agent.

On suppose que les agents possèdent des perceptions limitées, mais peuvent déterminer si un autre agent est présent sur une case voisine. La solution triviale est d'assigner un identifiant unique à chaque robot, puis de les faire avancer tour à tour sur la prochaine case libre. Cependant, cette méthode présente plusieurs inconvénients : des robots peuvent se trouver bloqués ; chaque robot a une solution en $O(m.n)$ ou $O(m.t)$, où t est la meilleure solution lorsque le robot est seul sur la grille, alors qu'une solution linéaire en n ou en t serait préférable ; enfin il est nécessaire de réaliser toute la planification a priori, ce qui est peu efficace si des robots sont amenés à entrer ou sortir du système en cours d'exécution.

L'utilisation de règles sociales permet de résoudre ces problématiques. Elles reposent sur l'utilisation d'une grille à plus gros grain, superposée à la grille originale (Fig. 2.2). En se basant sur cette grille complémentaire, des règles de circulation sont définies, permettant d'améliorer la coordination entre les agents. Par exemple, des sens de circulation sont imposés : dans les lignes paires, les agents doivent se déplacer de gauche à droite, et dans les colonnes impaires ils ne peuvent qu'aller de bas en haut. La preuve de non-collision repose sur des règles complexes de transitions entre les cases et de déplacement dans la case cible, que nous ne détaillons pas ici.

Elles permettent de prouver qu'aucune collision ne peut se produire, et qu'un but peut toujours être atteint en temps et énergie linéaires : $t + 2n + o(n)$ unités de temps, et $t + o(t)$ unités d'énergie (avec $m = o(t)$). Les agents peuvent rejoindre leur destination efficacement, et surtout avec un temps et des ressources très peu dépendants des plans des autres agents.

Le système, présenté comme une preuve de concept, peut être amélioré et complexifié, mais il permet d'illustrer l'intérêt d'introduire des mécanismes de coordination entre les agents. L'approche repose sur le choix des auteurs de concevoir le comportement des agents en amont de la simulation. L'idée est que le temps consacré à la conception, même s'il est important, deviendra négligeable par rapport au temps gagné lors de la simulation. Comme nous l'avons vu plus haut, ce type de démarche ne permet cependant pas d'adaptation simple aux changements de

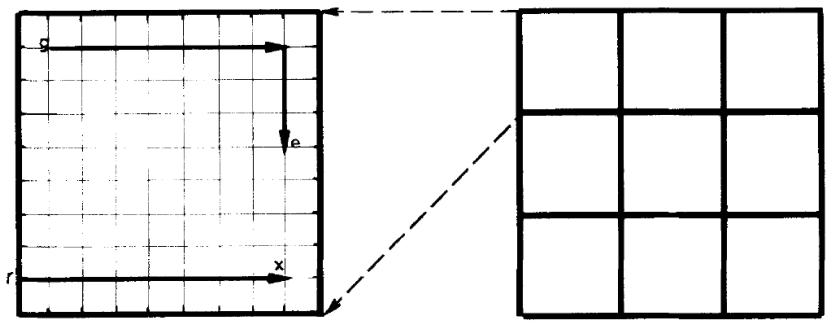


FIGURE 2.2 – Utilisation de lois sociales pour le déplacement de robots (d'après [Shoham & Tennenholtz, 1995]) : à gauche, la grille initiale, à droite la grille superposée permettant de coordonner le déplacement des robots.

comportement des agents, et ne laisse que peu de place à l'autonomie. Ici, les agents peuvent planifier eux-même une partie de leur déplacement, mais le champ des possibles est fortement contraint.

2.1.3.2 Adapter les normes au comportement des agents

Ce second exemple concerne la construction de systèmes normatifs adaptatifs. Bou et al. s'intéressent aux possibilités d'adaptation des normes dans le cadre des systèmes multi-agents ouverts [Bou et al., 2007]. En effet, dans le cas de populations d'agents égoïstes, intéressés uniquement par la réalisation de leurs buts individuels, il est intéressant de fournir à l'institution des mécanismes lui permettant de tenir compte des changement de buts des agents. L'institution s'adapte ainsi dynamiquement aux circonstances, permettant au système de devenir plus autonome. Cette forme d'auto-configuration est illustrée par une application dans le domaine du trafic routier.

Les possibilités d'évolution du système s'appuient sur deux éléments. Le premier est un ensemble d'objectifs qu'il doit satisfaire : dans le cas du trafic, cela pourrait être de minimiser le nombre d'accidents, tout en équilibrant le nombre de policiers et le montant des amendes. Chaque objectif est défini par un intervalle de valeurs réelles : si l'institution utilise une valeur dans l'intervalle, elle atteint le but. Le second élément concerne l'évolution des normes. Celles-ci sont utilisées pour contraindre le comportement des agents et définir les conséquences de leurs actions. Afin de rendre l'institution adaptative, les normes peuvent évoluer à travers un paramétrage spécifique : l'évolution des valeurs des paramètres modifie le comportement de la norme. Par exemple, le montant de la sanction associée à une violation est un de ces paramètres.

Les mécanismes d'adaptation ainsi ajoutés sont illustrés dans le cadre de la gestion du trafic. Le cas considéré est celui d'intersections sans signalisation, où les seules règles concernent la priorité à droite et le virage à gauche. Par ailleurs, les véhicules se voient associer un permis de conduire avec un nombre limité de points : si ce nombre atteint zéro, ils sont exclus de la simulation. L'environnement est modélisé sous la forme d'une grille, dont certaines cellules définissent les routes sur lesquelles les agents peuvent se déplacer (Fig. 2.3).

Le but fixé pour l'institution est de minimiser le nombre d'accidents, le nombre de violations, le nombre de véhicules bloqués, le nombre de permis retirés et le nombre d'agents de police. La fonction objectif favorise la satisfaction de l'ensemble des buts et pénalise un trop grand écart

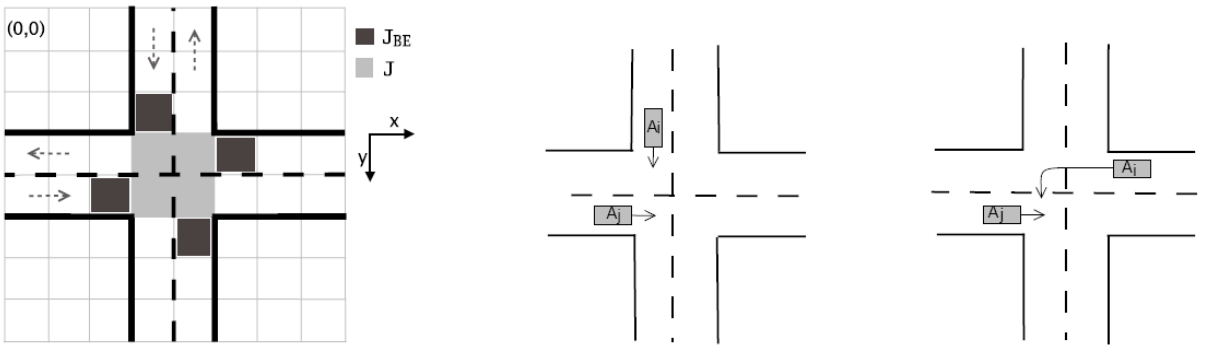


FIGURE 2.3 – Gestion d'intersection par des normes adaptatives (d'après [Bou et al., 2007]) : à gauche la modélisation de l'environnement, à droite les deux cas considérés et sanctionnés par les normes, la priorité à droite et le virage à gauche.

entre eux. Le système utilise deux normes différentes afin de gouverner le comportement des agents. La première concerne la priorité à droite, la seconde la priorité lorsque deux véhicules se trouvent face à face et que l'un tourne à gauche. Les normes sont constituées d'une action, d'une pré-condition et d'une sanction. Par exemple, dans le cas de la priorité à droite :

1. Action : au prochain pas de temps j'entre dans une intersection et je n'ai pas mis mon clignotant à droite (i.e. je ne tourne pas).
2. Pré-condition : il y a un véhicule à ma droite qui entrera dans le carrefour au prochain pas de temps.
3. Sanction : mon nombre de points de permis est décrémenté de la valeur associée à l'infraction. Cette valeur est une variable de la norme.

La norme est utilisée de la manière suivante : si j'effectue l'action (entrer dans le carrefour) alors que la pré-condition est vraie (il y a un autre véhicule, j'aurais dû céder la priorité), alors je suis sanctionné. Le choix de violer ou non la norme dépend d'une fonction aléatoire prenant en paramètre le montant de l'amende et le nombre de policiers présents. Finalement, un algorithme génétique est utilisé afin d'obtenir la meilleure répartition entre le montant des amendes et le nombre de policiers en fonction de la composition de la population, qui varie en nombre et en caractéristiques.

Du fait de la simplicité du protocole expérimental et du modèle de déplacement des agents, les résultats obtenus sont peu transposables dans le monde réel. En particulier, la fonction objectif choisie conduit à un nombre élevé d'agents de police après optimisation : il faut presque un agent de police pour chaque véhicule présent dans la simulation. Cependant, la démarche utilisée ainsi que le mode de description sont intéressants, en particulier les aspects de paramétrisation des normes et la définition d'objectifs pour l'institution.

2.2 Description normative des comportements

Cette seconde partie est consacrée à la présentation du premier axe du modèle que nous proposons : décrire le comportement des agents en s'appuyant sur des normes. Tout d'abord, nous introduisons les objectifs de notre approche, puis nous présentons la sémantique de notre modèle. Enfin, les violations sont abordées en tant que moyen pour créer de la variété.

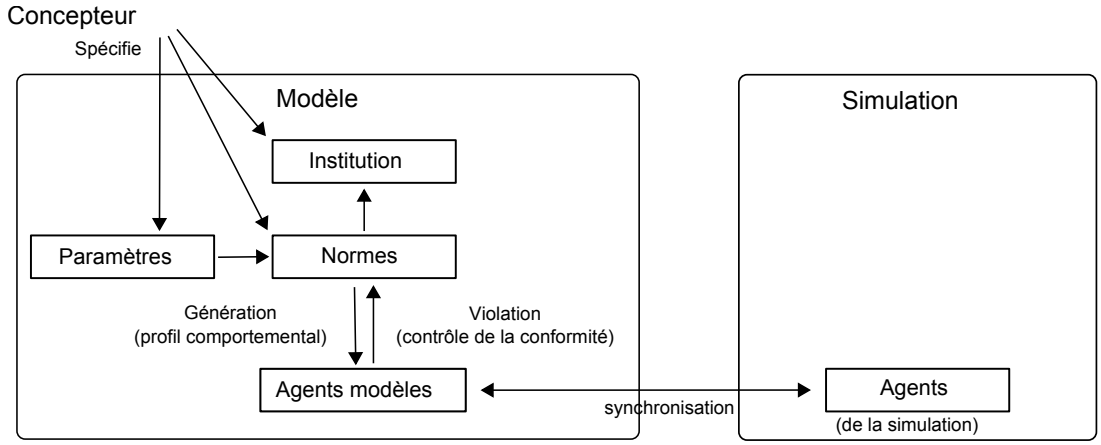


FIGURE 2.4 – Les différentes phases de notre approche : lors de la conception, le concepteur spécifie les normes par l'intermédiaire de leur paramètres ; lors de l'exécution, les agents modèles sont synchronisés avec les agents de la simulation afin de contrôler la conformité du paramétrage vis-à-vis de la norme.

2.2.1 Notre approche

Comme nous l'avons vu, la notion de norme est le plus souvent utilisée de manière prescriptive, en tant que contrainte sur les actions des agents. C'est par exemple le cas dans les approches présentées ci-dessus (Sect. 2.1.3), comme celle de Bou et al. [Bou et al., 2007] ou de Shoham & Tennenholz [Shoham & Tennenholtz, 1995], où l'objectif est de créer un ensemble de normes régissant de manière directe ou indirecte les actions des agents.

Notre approche s'inscrit dans une autre voie : mettre à profit les capacités descriptives des normes. En les abordant sous cet angle, elles permettent de spécifier le comportement des agents dans la simulation à travers leur paramétrage, puis de contrôler le respect de ces spécifications [Lacroix et al., 2008b]. Dans notre approche, le modèle est utilisé différemment suivant la phase en cours de la simulation (Fig. 2.4).

Lors de la phase de conception, le concepteur définit des normes. Celles-ci peuvent être vues comme des « profils comportementaux », notre utilisation des normes se rapprochant du sens « norme sociale ». Ces normes sont basées sur le paramétrage des agents de la simulation : à travers lui, le concepteur spécifie les comportements qu'il souhaite leur voir adopter.

Par ailleurs, des agents modèles encapsulent le paramétrage des agents de la simulation. Cela permet d'implémenter le modèle et la simulation dans des processus distincts et asynchrones, afin de faire face à d'éventuelles contraintes techniques et de permettre une utilisation non-intrusive du modèle. Les agents modèles constituent ainsi une représentation pivot entre les normes et les agents de la simulation.

Lors de l'exécution, les agents de la simulation évoluent et leur paramétrage peut changer, par exemple s'il est basé sur des paramètres dynamiques. L'agent modèle permet alors de contrôler ce paramétrage vis-à-vis de la norme adoptée, afin d'assurer sa conformité si nécessaire. Différentes réactions sont envisageables si l'agent de la simulation viole la norme, en fonction des souhaits du concepteur. Il est ainsi possible de tolérer les violations, de les limiter, ou encore de les interdire.

Décrire les comportements en utilisant une sémantique basée sur les normes nous paraît une approche à la fois originale et pertinente. Celles-ci permettent en effet de fournir des points

d'entrée adaptés aux concepteurs de simulations, qui peuvent spécifier les comportements qu'ils souhaitent obtenir en traduisant les profils comportementaux sous forme de normes. La variété désirée est obtenue grâce aux vastes possibilités de définition des normes et à la déviance. Ensuite, la possibilité de détecter les violations permet de contrôler les spécifications à l'exécution, et donc de garantir la conformité des comportements observés en fonction du niveau de tolérance spécifié par le concepteur.

Exemple Dans la suite de ce chapitre, nous avons choisi d'illustrer l'introduction des différentes notions sur un exemple commun, celui d'une simulation de piétons. Chaque piéton est caractérisé par trois paramètres comportementaux, qui sont pris en compte dans son modèle de décision. La vitesse désirée v_d traduit la vitesse à laquelle le piéton souhaiterait se déplacer. La vitesse maximale $v_{\max}$ représente la vitesse qu'il est physiquement capable d'atteindre. Finalement, l'espace personnel s accepté par le piéton représente la zone privée que chaque individu tente de préserver autour de soi ; lorsqu'elle n'est pas respectée, cela conduit par exemple à une augmentation du stress [Sommer, 1969]. Dans le modèle de décision, ce paramètre ne peut prendre que deux valeurs discrètes : s peut être soit « petit », soit « normal ».

Ces différents paramètres pourraient correspondre au paramétrage existant d'une simulation, telle que celles présentées dans la Section 1.2 (p. 19). Sans accès interne au modèle de décision, il s'agit des seuls éléments qui vont permettre d'influer sur le comportement des agents.

Par ailleurs, dans la Section 2.3 (p. 60) nous illustrons le fonctionnement de la génération du comportement des agents sur un exemple concernant le comportement de tir au but de joueurs de football.

2.2.2 Sémantique

Nous proposons un modèle de données basé sur la notion de normes. Dans cette section nous définissons la sémantique du modèle, avant de décrire ses propriétés puis la dynamique de son fonctionnement.

2.2.2.1 Cadre

On considère un système multi-agents constitué d'un ensemble d'agents $\mathcal{A} = \{a_1, a_2, \cdots, a_n\}$ définis dans un environnement $\mathcal{E}$. Chacun des agents $a_i \in \mathcal{A}$ possède un ensemble de propriétés $\mathcal{P}^{a_i}$ regroupant en particulier les éléments suivants :

- un ensemble de paramètres externes, statiques $\mathcal{P}^{a_i}_{e,s}$ ou dynamiques $\mathcal{P}^{a_i}_{e,d}$,
- un ensemble de paramètres internes, statiques $\mathcal{P}^{a_i}_{i,s}$ ou dynamiques $\mathcal{P}^{a_i}_{i,d}$.

On appelle paramètre statique un paramètre qui ne peut pas évoluer sous l'influence de la simulation seule. Il doit être fixé à l'initialisation ou modifié par intervention directe du concepteur ou de l'utilisateur. Au contraire, un paramètre dynamique est susceptible d'être modifié par le moteur de simulation lors de l'exécution. On appelle paramètre externe un paramètre rendu disponible à l'extérieur de la simulation, alors qu'un paramètre interne n'est pas accessible et ne pourra être modifié que par la simulation elle-même.

Typiquement les paramètres externes sont ceux disponibles dans les interfaces de configuration des simulations. Ils sont le plus souvent statiques, mais le cas de paramètres dynamiques se

rencontre également. Les paramètres internes servent au fonctionnement du moteur de simulation. Par exemple, un paramètre numérique inscrit directement dans le code (« codé en dur ») dont la valeur n'est jamais modifiée est un paramètre interne statique.

Exemple 1 *Dans la simulation de piétons introduite plus haut, $\mathcal{A}$ est l'ensemble des piétons. Tous les piétons possèdent le même paramétrage, et on a donc $\forall a_i \in \mathcal{A},\ \mathcal{P}^{a_i} = \{v_d, v_{\max}, s\} = \mathcal{P}$. En ce qui concerne la caractérisation des paramètres, $v_{\max}$ et s sont des paramètres externes statiques, alors que v_d est externe, mais dynamique. Suivant les notations introduites, $\mathcal{P}_{e,s} = \{v_{\max}, s\}$ et $\mathcal{P}_{e,d} = \{v_d\}$.*

Dans toute la suite nous nous focaliserons sur l'utilisation du modèle avec les paramètres externes de la simulation, $\mathcal{P}_{e,s}^{a_i}$ et $\mathcal{P}_{e,d}^{a_i}$. Le modèle est cependant applicable aux paramètres internes, comme nous le décrivons dans la Section 3.3.1 (p. 87). Enfin nous ne nous intéressons pas ici au fonctionnement du moteur de simulation lui-même. En effet, le modèle proposé en est indépendant lorsqu'il s'applique sur les paramètres externes.

Notons que le modèle ne s'applique que pour un sous-ensemble des agents de $\mathcal{A}$ possédant un paramétrage identique. Par exemple, si la simulation contient des piétons et des véhicules, le modèle devra être appliqué indépendamment sur chacun des types d'agents.

2.2.2.2 Définitions

Intéressons nous maintenant à la sémantique du modèle.

Définition 3 *Un* Paramètre p *est un sextuplet $(l_p, p_{\text{ref}}, \mathcal{D}_p, v_{d_p}, g_p, f_p)$ défini par :*
- *l_p un label unique,*
- *p_{ref} un paramètre de référence,*
- *$\mathcal{D}_p$ un domaine de définition fini, avec si $p_{\text{ref}} \neq 0$, alors $\mathcal{D}_p \subseteq \mathcal{D}_{p_{\text{ref}}}$,*
- *$v_{d_p} \in \mathcal{D}_p$ une valeur par défaut,*
- *g_p une distribution de probabilité sur $\mathcal{D}_p$,*
- *$f_p : \mathcal{D}_{p_{\text{ref}}} \mapsto [0,1]$ une fonction distance. Si $p_{\text{ref}} = 0$, alors $f_p : \mathcal{D}_p \mapsto [0,1]$.*

Un label permet d'identifier le paramètre de manière unique. Le paramètre de référence correspond soit à un autre paramètre, soit à la valeur nulle. Dans le cas où $p_{\text{ref}}(p) = 0$, p est dit paramètre « racine ». Le domaine de définition définit le typage du paramètre ainsi que sa plage de valeurs. Un paramètre pourra être réel, entier, constitué de chaînes de caractères, de règles d'action, et le domaine être discret, continu, discontinu... Notons que dans les applications présentées, les paramètres sont le plus souvent réels et définis sur des domaines soit discrets, soit continus. Si le paramètre de référence p_{ref} n'est pas nul, le domaine de définition doit être un sous-domaine du domaine de p_{ref}. Un paramètre contient une valeur par défaut, incluse dans le domaine de définition. Par défaut, sur un domaine réel discret ou discontinu cette valeur est la valeur la plus proche de la moyenne des valeurs du domaine ; sur un domaine réel continu il s'agit de la moyenne de l'intervalle.

Exemple 2 *Le paramètre v_{max} décrivant la vitesse maximale des piétons peut être défini par : $l_p = v_{max}$, $p_{\text{ref}} = 0$, $\mathcal{D}_p = [0, 20]$ km/h, $v_{d_p} = 5$ km/h, g_p et f_p les fonctions par défaut. Pour un paramètre discret comme s, la définition peut être : $l_p = \text{s}$, $p_{\text{ref}} = 0$, $\mathcal{D}_p = \{petit, normal\}$, $v_{d_p} = normal$, g_p et f_p les fonctions par défaut.*

Ces définitions utilisent des domaines de définition assez larges. Ces paramètres sont des paramètres « racine » car $p_{\text{ref}} = 0$*. Leur but fonctionnel est de définir les limites maximales des domaines de définition dans l'application.*

Un paramètre inclut également une distribution de probabilité sur son domaine de définition $\mathcal{D}_p$, utilisée pour associer des valeurs au paramètre. Par défaut cette distribution est uniforme. Le paramètre contient enfin une fonction distance, qui permet de déterminer l'écart entre la valeur d'un paramètre et le domaine de définition. Elles est destinée à quantifier d'éventuelles violations (cf. Sect. 2.2.3 p. 55). Par défaut, les fonctions suivantes sont utilisées :

– sur un domaine discret ou discontinu :

$$\forall x \in \mathcal{D}_{p_{\text{ref}}},\ f_p(x) = \begin{cases} 0 & \text{si } x \in \mathcal{D}_p \\ 1 & \text{sinon} \end{cases}$$

– sur un domaine continu :

$$\forall x \in \mathcal{D}_{p_{\text{ref}}},\ f_p(x) = \begin{cases} 0 & \text{si } x \in \mathcal{D}_p \\ \dfrac{\mid x - mean(\mathcal{D}_p) \mid}{max(\mathcal{D}_{p_{\text{ref}}}) - min(\mathcal{D}_{p_{\text{ref}}})} & \text{sinon} \end{cases}$$

Ces fonctions peuvent être remplacées par des fonctions fournies par l'utilisateur, la seule contrainte étant qu'elles doivent prendre leur valeur entre 0 et 1.

Définition 4 *Une* Norme N *est un ensemble* $\{l_N, N_{\text{ref}}, \mathcal{P}_N, \mathcal{Q}_N, \tau_N, \delta_{\max_N}\}$ *tel que :*

– l_N *un label unique,*

– N_{ref} *une norme de référence non nulle (*$N_{\text{ref}} \neq 0$*),*

– $\mathcal{P}_N$ *un ensemble fini de paramètres p, vérifiant :*

$$\begin{cases} \forall p \in \mathcal{P}_N,\ p_{\text{ref}}(p) \neq 0 \\ \forall p_1 \in \mathcal{P}_N,\ \exists p_2 \in \mathcal{P}_{N_{\text{ref}}},\ p_{\text{ref}}(p_1) = p_2 \\ \forall p_1 \in \mathcal{P}_N,\ \forall p_2 \in \mathcal{P}_N - \{p_1\},\ p_{\text{ref}}(p_1) \neq p_{\text{ref}}(p_2) \end{cases}$$

– $\mathcal{Q}_N$ *un ensemble de propriétés (institutionnelles ou environnementales),*

– τ_N *un taux de violation de la norme de référence, avec* $\tau_N \in [0, 1]$*,*

– $\delta_{\max_N}$ *un écart maximal à la norme, avec* $\delta_{\max_N} \in [0, 1]$*.*

Une norme est identifiée par un label unique. Elle contient une référence à une autre norme : la notion de hiérarchie induite par cette référence ainsi que son initialisation (cas $N_{\text{ref}} = 0$) sont décrits dans la Section 2.2.2.3. Une norme contient également un ensemble fini de paramètres, dont le paramètre de référence doit être non nul. Il doit provenir de la norme de référence, et chaque paramètre de la norme de référence ne peut être inclus qu'une seule fois. Par contre tous les paramètres de la norme de référence ne sont pas nécessairement inclus. On notera par ailleurs que pour chacun des paramètres inclus, le domaine de définition du paramètre dans N est inclus dans celui du paramètre de la norme de référence, par construction des paramètres.

La norme contient par ailleurs un ensemble de propriétés permettant de décrire des caractéristiques institutionnelles ou environnementales. Elles permettent par exemple de préciser le contexte d'application d'une norme, comme la zone géographique dans laquelle elle s'applique. Ces propriétés pouvant ultérieurement être exploitées par le modèle de simulation. La spécialisation des normes suivant le type d'agents, évoquée plus haut, peut être réalisée par leur intermédiaire.

Exemple 3 *Avant de définir un exemple de norme, nous allons enrichir l'ensemble des paramètres disponibles. Aux deux paramètres déjà décrits, nous ajoutons (i) la vitesse maximale normale, et (ii) l'espace personnel normal :*

(i) $l_p = v_{max,normal}$, $p_{\text{ref}} = v_{max}$, $\mathcal{D}_p = [4, 6]$ km/h, $v_{d_p} = 5$ km/h, g_p *distribution normale de moyenne* v_{d_p} *et de variance 1 tronquée aux limites de* $\mathcal{D}_p$, *et* f_p *la fonction distance par défaut.*

(ii) $l_p = s_{normal}$, $p_{\text{ref}} = \text{s}$, $\mathcal{D}_p = \{normal\}$, $v_{d_p} = normal$, g_p *et* f_p *les fonctions par défaut.*

On suppose que le système contient déjà une norme N_0 *avec* $\mathcal{P}_{N_0} = \{v_{max},\ \text{s}\}$. *La norme* N_{normal} *peut alors être définie par* $l_N = N_{\text{normal}}$, $N_{\text{ref}} = N_0$, $\mathcal{P}_N = \{v_{max,normal},\ s_{normal}\}$ *et* $\mathcal{Q}_N = \emptyset$.

Les paramètres de N_{normal} *sont non nuls, leurs paramètres de référence aussi et leurs domaines de définition sont bien inclus dans ceux de leurs paramètres de référence.*

Le taux de violation τ_N est exploité lors de la génération des comportements, comme nous le verrons dans la Section 2.3. Il permet à l'utilisateur de spécifier s'il souhaite permettre l'apparition de comportements violateurs, et quelle proportion de tels comportements doit apparaître. Enfin l'écart maximal à la norme spécifie la tolérance par rapport à la violation de la norme. Une norme permissive peut ainsi avoir un $\delta_{\max}$ élevé, afin d'élargir le spectre du paramétrage disponible.

Exemple 4 *Définissons* $\tau_{N_{\text{normal}}} = 0.05$, *et* $\delta_{\max_{N_{\text{normal}}}} = 0.1$. *Les agents auront 5 % de chances d'être en violation lors de leur création et l'écart maximal à la norme pourra atteindre 10 %. Les piétons possèdent ainsi un paramètrage créant dans la simulation un comportement qui paraîtra normal, les possibilités de violation autorisant l'apparition de comportements légèrement différents, qui augmentent la variété.*

Dans la suite des exemples, on prendra $\tau_{N_{\text{normal}}} = 0$ *et* $\delta_{\max_{N_{\text{normal}}}} = 1$ *: ainsi, aucune violation n'est autorisée par le modèle et l'écart à la norme n'est pas une contrainte.*

Notons par ailleurs que les normes conflictuelles sont autorisées. Le modèle permet en effet que des normes différentes puissent utiliser des paramètres identiques, ou possèdent des paramètres ayant des domaines de définition se chevauchant.

Définition 5 *Un* Agent modèle s *est un ensemble* $\{N_s, \mathcal{C}_s\}$ *tel que :*

- N_s *une norme de référence,*
- $\mathcal{C}_s = \{(l_p, v_{p_s}),\ p \in \mathcal{P}_{N_s}\}$ *un ensemble de couples de labels de paramètres et de valeurs associées, vérifiant :*

$$\begin{cases} \forall c \in \mathcal{C}_s,\ \exists p_1 \in \mathcal{P}_{N_s},\ c = (l_{p_2}, v_{p_{2_s}}) \wedge l_{p_2} = l_{p_1} \\ \forall p \in \mathcal{P}_{N_s},\ \exists c \in \mathcal{C}_s,\ c = (l_p, v_{p_s}) \end{cases}$$

L'agent modèle contient d'une part une norme de référence, et d'autre part un ensemble de couples de labels de paramètres et de valeurs associées. Les paramètres référencés par ce biais doivent provenir de la norme de référence, et tous les paramètres définis dans la norme doivent être inclus. À ce stade, aucune contrainte n'est imposée sur la valeur associée au paramètre. En particulier, nous n'imposons pas qu'elle soit incluse dans le domaine de définition $\mathcal{D}_p$ tel que défini dans la norme de référence. Un Agent modèle est l'« instantiation » d'une norme : il contient l'ensemble des paramètres, associés à une valeur particulière. Le processus d'instanciation sera décrit dans la Section 2.3.

Exemple 5 *Un Agent modèle s_1 issu de la norme N_{normal} peut ainsi avoir pour caractéristiques $N_{s_1} = N_{\text{normal}}$ et $\mathcal{C}_{s_1} = \{(v_{max,normal},\ 5.2), (s_{normal},\ normal)\}$. L'agent fait correspondre une valeur à chacun des paramètres inclus dans sa norme de référence.*

L'agent modèle introduit une couche supplémentaire entre la norme et l'agent de la simulation. Il joue le rôle de représentation pivot entre ces deux éléments, afin de permettre une implémentation du modèle dans un processus distinct de la simulation. De cette façon, le modèle peut être utilisé de manière non-intrusive, comme nous le verrons dans la Section 3.3 (p. 86). Les agents modèles permettent ainsi d'encapsuler le paramétrage des agents de la simulation, tout en associant à ce paramétrage une norme de référence.

Définition 6 *Une* Institution I *est un ensemble $\{l_I, \mathcal{N}_I, \sigma_I\}$ tel que :*
- *l_I un label unique,*
- *$\mathcal{N}_I$ un ensemble fini de normes,*
- *σ_I une valeur décrivant le critère global de déterminisme de la simulation, avec $\sigma_I \in [0, 1]$.*

Dans le cadre de ce travail, l'institution est constituée d'un label permettant de l'identifier de manière unique, ainsi que d'un ensemble fini de normes. Elle permet de ne référencer qu'une partie des normes définies dans l'environnement. Finalement, une valeur réelle σ_I permet de spécifier le niveau global de déterminisme de la simulation. Cette valeur est utilisée lors de la génération des agents modèle.

Exemple 6 *L'ensemble $\{i_{ville},\ \{N_{\text{normal}}\},\ 0.01\}$ constitue une institution. Celle-ci définit l'utilisation de la norme piéton normal en ville, avec un critère global de déterminisme de 1 %.*

L'ensemble des éléments introduits peut être synthétisé sous forme d'un schéma UML (Fig. 2.5).

2.2.2.3 Hiérarchie des normes

Chaque norme possède une norme de référence, ce qui induit une hiérarchie entre elles. Cette hiérarchie permet d'introduire des caractéristiques communes pour un ensemble de normes, du fait de l'inclusion du domaine de définition de chaque paramètre dans celui de son paramètre de référence. Ainsi, la distinction entre un ensemble de normes pour des piétons en ville et des piétons à la campagne peut se faire par son intermédiaire (Fig. 2.6). Par exemple, en définissant une borne maximale de vitesse plus faible dans N_{campagne} que dans N_{ville}, cette limite est respectée par toutes les normes de la branche correspondante.

On impose cependant que cette hiérarchie possède une racine unique N_{init}. La particularité de cette racine est que sa norme de référence est nulle : $N_{\text{ref}}(N_{\text{init}}) = 0$. N_{init} est définie de la manière suivante :

Définition 7 *La norme racine N_{init} est l'ensemble $\{l_{N_{\text{init}}}, N_{\text{ref}_{\text{init}}}, \mathcal{P}_{N_{\text{init}}}, \mathcal{Q}_{N_{\text{init}}}\}$ tel que :*
- *$l_{N_{\text{init}}}$ un label unique,*
- *$N_{\text{ref}_{\text{init}}} = 0$,*
- *$\mathcal{P}_{N_{\text{init}}}$ un ensemble fini de paramètres p, vérifiant $\forall p \in \mathcal{P}_{N_{\text{init}}},\ p_{\text{ref}}(p) = 0$,*
- *$\mathcal{Q}_{N_{\text{init}}}$ un ensemble de propriétés (institutionnelles ou environnementales).*

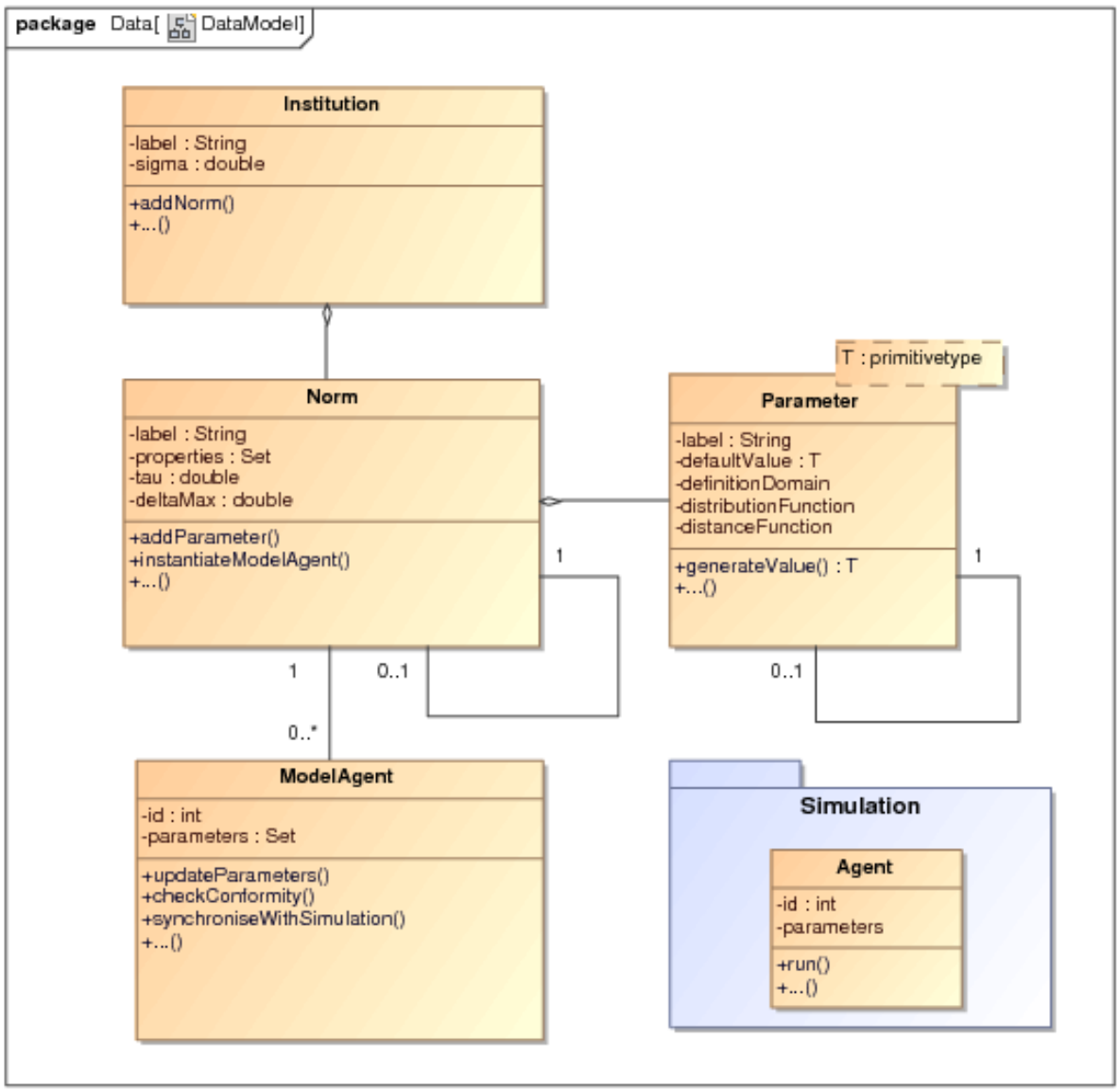


FIGURE 2.5 – Représentation synthétique du modèle de données proposé

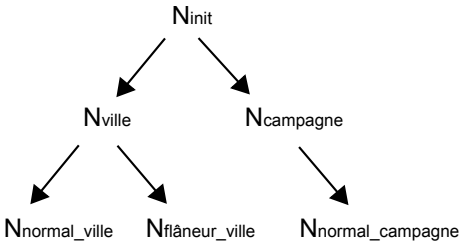


FIGURE 2.6 – Hiérarchie de normes, permettant d'introduire des caractéristiques communes pour un ensemble de normes. La norme racine N_{init} est unique et régimentée.

Les paramètres contenus dans la norme racine doivent donc également être des paramètres « racine », c'est-à-dire dont le paramètre de référence est nul. On notera que pour les paramètres de $p \in \mathcal{P}_{N_{\text{init}}}$, $f_p = 0$ sur tout le domaine : ces fonctions n'ont d'intérêt que pour les autres normes. Par ailleurs, comme nous l'avons vu dans la définition générale des normes (Définition 4), pour toutes les normes $N \neq N_{\text{init}}$, $N_{\text{ref}}(N) \neq 0$, ce qui garantit l'unicité de la racine de la hiérarchie. Enfin, on remarque que par définition, pour toute norme $N \neq N_{\text{init}}$, $\forall p \in \mathcal{P}_N$, $p_{\text{ref}}(p) \neq 0$: il ne peut donc pas y avoir de paramètres « racine » ailleurs que dans N_{init}.

N_{init} contient alors l'ensemble des paramètres qui pourront être utilisés dans les normes, associés à leurs domaines de définition dans leur forme la plus large. En effet :

1. N_{init} est la seule norme qui peut inclure des paramètres « racine » (Définition 7),
2. les domaines de définition des paramètres doivent être des sous-domaines de celui de leur paramètre de référence (Définition 3),
3. les normes $N \neq N_{\text{init}}$ ne contiennent donc nécessairement que des paramètres ayant un domaine au plus aussi grand que celui de N_{init}.

De plus, N_{init} est implémentée dans le système en tant que norme régimentée (cf. Sect. 2.1.2 p. 43). Il n'est donc pas possible de la violer, et par conséquent les domaines de définitions spécifiés pour ses paramètres seront les limites absolues que ceux-ci pourront prendre dans le système.

Exemple 7 *On définit la norme N_{pietons}, qui sera la norme racine dans notre modèle : $l_N = N_{\text{pietons}}$, $N_{\text{ref}} = 0$, $\mathcal{P} = \{v_{max}, \text{s}\}$ (tels que définis dans l'exemple 2 p. 49) et $\mathcal{Q}_N = \emptyset$. N_{pietons} ne contient bien que des paramètres « racine », et sa norme de référence est nulle.*

On peut alors modifier la norme N_{normal} telle que définie dans l'exemple 3 afin qu'elle se base sur N_{pietons}. Les paramètres de N_{normal} ayant pour référence ceux utilisés dans N_{pietons}, il suffit de modifier $N_{\text{ref}}(N_{\text{normal}})$ et lui faire prendre la valeur N_{pietons}.

N_{init} fournit un cadre global pour la future exécution des normes, en garantissant pour l'ensemble des paramètres une limite qui ne pourra pas être outrepassée par le système.

2.2.2.4 Affectation des normes

La finalité du modèle est de générer un paramétrage pour les agents présents dans la simulation. Nous décrivons ici la méthode utilisée pour réaliser cette génération. Notons que les valeurs

proprement dites sont créées par un mécanisme spécifique, qui est détaillé dans la Section 2.3. Nous ne nous y intéresserons donc pas ici.

Le processus est déclenché lors de la création d'un agent de la simulation, ou sur requête externe (une action utilisateur par exemple). La méthode de sélection de la norme fait partie de la configuration : par exemple, chaque agent peut se voir affecter la même norme, l'attribution peut se faire en fonction de distributions statistiques [11], etc. L'algorithme 1 présente l'ensemble des opérations réalisées pour l'affectation d'un paramétrage correspondant à la norme n à l'agent de la simulation $a \in \mathcal{A}$.

Tout d'abord, la norme ciblée est instanciée sous la forme d'un agent modèle, en fonction de ses propres spécifications et en utilisant le moteur de génération décrit dans la Section 2.3. L'agent modèle ainsi créé contient l'identifiant de l'agent de la simulation lui correspondant, la norme de référence, ainsi qu'un ensemble de couples (paramètre, valeur). Au minimum, tous les paramètres de $\mathcal{P}_{e,s}$ et de $\mathcal{P}_{e,d}$ sont présents dans l'agent modèle. Pour tous les paramètres spécifiés dans la norme (i.e. tous les $p \in \mathcal{P}_{N_{G_s}}$), la valeur créée dans l'agent modèle est affectée au paramètre correspondant. Pour tous les autres paramètres, une valeur par défaut est utilisée. La valeur choisie est celle de la valeur par défaut du paramètre dans la première norme de référence contenant ce paramètre. L'agent modèle créé envoie alors l'ensemble de son paramétrage à l'agent de la simulation lui correspondant.

Cette méthode permet de conserver une indépendance complète du modèle avec la simulation. Conserver une copie de l'agent sous forme d'agent modèle n'est pas indispensable si on souhaite uniquement générer un paramétrage initial pour les agents de la simulation. Si par contre on souhaite observer son évolution (en s'intéressant par exemple aux violations), connaître la norme de référence de l'agent de la simulation est nécessaire. Celui-ci ne pouvant a priori pas être modifié, il nous faut donc une copie externe où conserver la norme de référence. C'est le but de l'agent modèle créé, qui permet en outre de stocker le paramétrage initial. De cette façon il sera possible de faire intervenir un contrôle par les normes de manière totalement externe, dans un module séparé.

2.2.3 Violations de la norme

Les violations sont une notion intrinsèque au concept de normes. Dans notre approche, nous les exploitons afin d'augmenter la variété des comportements créés. Nous présentons ici la définition que nous utilisons, le mécanisme de quantification qui nous permet de contrôler la variété, puis les différentes réactions que le modèle proposé peut avoir face à elles.

2.2.3.1 Définition

Comme nous l'avons vu plus haut, l'utilisation du modèle de données n'impose pas de contrainte sur le modèle de décision des agents. Cela signifie en particulier que leurs paramètres peuvent prendre des valeurs en dehors des domaines définis dans les normes. En effet, la valeur des paramètres dynamiques $p \in \mathcal{P}_{e,d}^{a_i}$ est a priori modifiée lors de l'exécution de la simulation, sans limitation particulière. En ce qui concerne les paramètres statiques $p \in \mathcal{P}_{e,s}^{a_i}$, rien ne garantit que des modules externes ou des actions utilisateur ne vont pas leur imposer de changements les conduisant en dehors de la norme, indépendamment de notre propre action.

11. Ces méthodes sont approfondies dans le Chapitre 4.

Algorithme 1 Procédure d'affectation des normes : **affectation_norme**(a, n).

Entrées : un agent $a \in \mathcal{A}$ de la simulation ;
$\mathcal{A}_m$ l'ensemble des agents modèle ;
une norme $N = \{N,\ N_{\text{ref}},\ \mathcal{P}_N,\ \mathcal{Q}_N,\ \tau_N,\ \delta_{\max_N}\}$;
σ le critère global de déterminisme de l'institution.

Sorties : a affecté d'un paramétrage ;
a_m un agent local au modèle contenant un paramétrage identique à a

initialisation : instanciation d'un agent modèle et de son paramétrage à partir de la norme N (cf. Alg. 6) : $a_m^{temp} \leftarrow$ **instantiate_norm**(N, σ) ;
création d'un agent modèle a_m ayant pour référence a
pour tout $p \in \mathcal{P}_e^a$ **faire**
    si $p \in \mathcal{P}_N$ **alors**
        récupérer la valeur de ce paramètre depuis l'agent modèle temporaire : $v_p \leftarrow v_p(a_m^{temp})$
    sinon
        remonter la hiérarchie des normes pour trouver une valeur par défaut : $N' \leftarrow N$
        tant que $N' \neq N_{\text{init}}$ **faire**
            remonter la hiérarchie : $N' \leftarrow N_{\text{ref}}(N)$
            si $p \in \mathcal{P}_{N'}$ **alors**
                récupérer la valeur par défaut dans la norme : $v_p \leftarrow v_{d_p}(N')$
                break ;
            fin si
        fin tant que
    fin si
    sauvegarder la valeur obtenue dans l'agent modèle et dans l'agent de la simulation : $v_p(a_m),\ v_p(a) \leftarrow v_p$
fin pour

La notion de norme contient intrinsèquement celle de violation. Dans le cadre de notre approche, nous nous basons sur celles-ci afin de contrôler l'évolution du paramétrage des agents, ainsi que la conformité aux spécifications fournies initialement. Les violations sont définies de la manière suivante :

Définition 8 *Un Agent modèle* s *est en violation vis-à-vis de sa norme de référence N si et seulement si $\exists c = (l_p, v_{p_s}) \in \mathcal{C}_s,\ v_{p_s} \notin \mathcal{D}_p$ où $p \in \mathcal{P}_{N_s}$ (par définition p existe et est unique).*

Une violation est donc définie vis-à-vis de la conformité à la norme de référence. Un agent modèle est violateur si et seulement si au moins une valeur associée à un paramètre qu'il inclut n'appartient pas au domaine défini pour ce paramètre dans la norme de référence. Il s'agit d'un critère simple permettant de déterminer à tout instant le respect de la norme.

Exemple 8 *Soit un agent modèle s_2 issu de la norme N_{normal}. s_2 est défini par $N_{s_2} = N_{\text{normal}}$ et $\mathcal{C}_{s_2} = \{(v_{max,normal},\ 7), (s_{normal},\ normal)\}$. La norme de référence spécifie $\mathcal{D}_{v_{max,normal}} = [4, 6]$ km/h. On a $7 \notin \mathcal{D}_{v_{max,normal}}$, donc s_2 est en violation.*

Les violations permettent d'augmenter la variété des comportements obtenus en s'éloignant des spécifications fournies pas le concepteur. Obtenir ces violations peut en effet faire partie des objectifs de la simulation, si par exemple on souhaite voir apparaître des comportements non spécifiés. Le modèle lui même est donc en mesure de générer des agents en violation, cela n'étant pas en contradiction avec les définitions fournies (ce point sera abordé plus en détails dans la Section 3.2). Cependant, le modèle ne présente pas d'intérêt s'il n'est pas possible de garantir une certaine conformité aux spécifications. Il est donc nécessaire d'être en mesure de quantifier les violations, afin de pouvoir en contrôler l'impact.

2.2.3.2 Quantification des violations

En considérant les définitions proposées, la quantification des violations peut s'aborder suivant deux axes complémentaires :

1. Par le nombre de paramètres qui sont en dehors du domaine de définition spécifié dans la norme. Plus ce nombre est important, plus le comportement résultant sera éloigné de celui spécifié par le concepteur.
2. Par l'écart entre la valeur prise par un paramètre et le domaine de définition défini dans la norme. Si cet écart est faible, le comportement produit sera généralement proche de celui recherché.

Cela nous conduit à définir une grandeur permettant de quantifier les violations :

Définition 9 *On note q_s la grandeur définie par :*

$$q_s = \frac{\sum_{c \in \mathcal{C}_s} f_p(v_{s_c})}{Card(\mathcal{P}_{N_s})}$$

Algorithme 2 Procédure de quantification des violations : **quantify_violation(s)**.

Entrées : un agent modèle $s = \{N_s, \mathcal{C}_s\}$
Sorties : un taux de violation q_s, relatif à la norme de référence de s

$q_s, i \leftarrow 0$
pour tout paramètre $p \in N_s$ **faire**
    incrémenter i : $i \leftarrow i + 1$
    récupérer la valeur de p dans l'agent modèle : $v_p \leftarrow v_p(s)$
    récupérer le domaine de définition de p dans la norme de référence de s : $\mathcal{D}_p \leftarrow \mathcal{D}_p(N_s)$
    si $v_p \notin \mathcal{D}_p$ **alors**
        ajouter la distance au domaine à q_s : $q_s \leftarrow q_s + f_p(v_p)$
    fin si
fin pour
diviser la valeur finale par le cardinal de N_s : $q_s \leftarrow q_s / i$
retourner q_s

Elle représente une quantification de la violation de l'agent s. Plus le nombre de paramètres en violation est grand et plus l'écart au domaine de définition du paramètre est important (quantifié par la fonction f_p), plus la valeur de q_s sera élevée. Cette valeur permet ainsi de qualifier le paramétrage des agents par rapport à leur écart à la norme. Par ailleurs, par définition $\forall c \in \mathcal{C}_s,\ f_p(v_{s_c}) \in [0, 1]$ et $Card(\mathcal{C}_s) \leq Card(\mathcal{P}_N)$: q_s est toujours compris entre 0 et 1. Enfin on peut remarquer que si $q_s \neq 0$, alors s est en violation. On dispose donc d'un second critère permettant de détecter celle-ci, outre la Définition 8.

Le calcul de q_s se fait suivant l'algorithme 2. Pour chaque paramètre p de la norme de référence de l'agent modèle s, on récupère d'une part la valeur actuelle v_p stockée dans l'agent modèle, et d'autre part le domaine de définition stocké dans la norme. Si la valeur n'est pas incluse dans le domaine, alors on quantifie l'écart en utilisant la fonction distance f_p, et on ajoute cette valeur à q_s. Finalement on divise cette valeur par le nombre de paramètres définis dans la norme.

2.2.3.3 Réaction aux violations

Losqu'une violation est détectée, le modèle peut réagir soit en l'autorisant, soit en l'interdisant. Cette réaction est spécifiée par l'utilisateur, en utilisant l'écart maximal à la norme $\delta_{\max_N}$. Ce paramètre correspond à la valeur maximale que q_s peut prendre sans déclencher de réaction.

L'algorithme 3 décrit ce fonctionnement. Il est déclenché à chaque pas de temps. Pour chaque agent de la simulation, on quantifie l'écart q_s entre son paramétrage actuel et sa norme initiale N, obtenue à partir de l'agent modèle correspondant. Si $q_s \leq \delta_{\max_N}$, les limites sont respectées, l'agent modèle est mis à jour avec les paramètres provenant de l'agent de la simulation. Si $q_s > \delta_{\max_N}$ la violation est trop importante, on l'interdit : les paramètres de l'agent modèle sont recopiés dans l'agent, écrasant les données existantes. De cette façon, l'agent de la simulation retourne à son dernier état conforme.

Nous avons fait le choix de quantifier la violation de manière globale pour une norme : q_s prend en compte simultanément tous les paramètres. L'objectif est de simplifier la configuration, en évitant d'avoir à fixer un seuil pour chacun d'entre eux. Cependant, il n'est par conséquent pas possible de réagir indépendamment à leurs valeurs : un paramétrage dont la violation dépasse

Algorithme 3 Réaction aux violations.

Entrées : $\mathcal{A}_m$ l'ensemble des agents modèle ; $\mathcal{A}$ l'ensemble des agents de la simulation

1: **pour tout** $a_m \in \mathcal{A}_m$ **faire**
2: créer une copie temporaire de a_m pour le calcul : $a_{temp} \leftarrow a_m$
3: recopier toutes les valeurs de l'agent de la simulation a correspondant à a_m dans a_{temp} : $\forall p \in \mathcal{P}_a,\ v_p(a_{temp}) \leftarrow v_p(a)$
4: quantifier les violations de a_{temp} (correspondant au paramétrage de a) : $q_{a_{temp}} \leftarrow$ **quantify_violation**(a_{temp})
5: **si** $q_{a_{temp}} \leq \delta_{\max_{N_{a_m}}}$ **alors**
6: l'écart maximal à la norme n'est pas dépassé, la simulation se poursuit : $\forall p \in \mathcal{P}_a,\ v_p(a_m) \leftarrow v_p(a)$
7: **sinon**
8: la violation est interdite, on force les paramètres de a à leur ancienne valeur : $\forall p \in \mathcal{P}_{a_m},\ v_p(a) \leftarrow v_p(a_m)$
9: **fin si**
10: **fin pour**

l'écart maximal doit être refusé dans sa globalité. Par exemple, si un seul paramètre est très fortement en violation, alors que tous les autres sont dans leur plage de validité, tous seront pourtant fixés : on ne peut pas contraindre uniquement le paramètre causant la violation et laisser évoluer les autres librement.

Notons que :
- si $\delta_{\max_N} = 0$, tous les paramètres de l'agent seront toujours compris dans le domaine de définition de la norme,
- si $\delta_{\max_N} = 1$, le modèle ne contraindra jamais aucun paramètre de la simulation ($q_s \leq 1$).

Cette flexibilité permet de contrôler finement la simulation : dans le cas d'expérimentations où les comportements doivent être sous contrôle, on limitera ou on interdira les violations, tout en fournissant des normes strictes. Dans le cas d'expérimentations où des comportements nouveaux ou originaux peuvent apporter des résultats intéressants – afin d'expérimenter des comportements émergents ou adaptatifs –, la tolérance aux limites des normes pourra être plus grande. Les violations, qui peuvent mener à des comportements atypiques, font alors partie des aspects recherchés.

Plusieurs éléments doivent cependant être considérés attentivement lorsque l'on souhaite utiliser cette méthode pour interdire les violations. L'effet de l'algorithme est que si la violation est trop importante, on empêche le modèle de décision de modifier certains paramètres des agents, ce qui peut avoir plusieurs conséquences. Tout d'abord, cela peut empêcher le bon déroulement de la simulation, qui n'a a priori pas été conçue pour tenir compte de telles perturbations extérieures. Par exemple, si le paramètre que nous fixons était lié à une action d'urgence, des comportements anormaux peuvent apparaître (des piétons entrant en collision car ils ne peuvent pas ralentir assez vite...). Ensuite, cette correction se fait avec un pas de temps de retard sur la simulation : comme c'est le modèle de décision qui met à jour les agents, il faut attendre qu'il ait agit avant de pouvoir réaliser d'éventuelles correction. En fonction de la fréquence de rafraichissement, ce point peut causer des artefacts : par exemple, si la vitesse d'un agent avait fortement augmenté, et que la réaction aux violations la diminue brusquement, le comportement paraîtra anormal (alors même que la violation peut avoir été causée par un autre paramètre...).

Il est donc nécessaire d'utiliser cette fonctionnalité avec prudence. Une possibilité est de se

restreindre aux normes ne contraignant que des paramètres qui ne produiront pas ce type de problème. Pour éviter un maximum d'erreurs, la valeur par défaut de $\delta_{\max_N}$ est par ailleurs fixée à 1. Notons enfin qu'en pratique cette limite n'est pas très contraignante, car pour la plupart des usages les violations sont soit toutes autorisées, soit toutes interdites.

2.3 Génération des comportements

Nous décrivons dans cette section un algorithme générique permettant de contrôler le degré d'aléatoire introduit dans un processus d'instanciation. Comme nous l'avons vu dans la section précédente, notre approche décrit les comportements attendus des agents sous forme normative, en s'appuyant sur le paramétrage de la simulation. L'affectation d'un paramétrage à un agent nécessite l'instanciation d'une norme en un agent du modèle. L'algorithme proposé s'applique particulièrement bien à ce cadre, et nous présentons son utilisation dans ce contexte.

2.3.1 Présentation

Cette première partie présente les objectifs poursuivis lors de la conception de l'algorithme. Nous décrivons ensuite son fonctionnement, puis de quelle façon il répond au besoin ciblé.

2.3.1.1 Objectifs

L'algorithme présenté dans cette partie a pour but de prendre en compte différents besoins utilisateurs : généricité, flexibilité et reproductibilité.

Les premiers critères de généricité et de flexibilité sont liés au besoin d'utiliser un algorithme indépendant du domaine étudié. Il pourra être appliqué à l'identique dans différents cas d'utilisation, avec une simple redéfinition de ses éléments de base et sans modification de sa structure. Outre cette indépendance, il s'agit de limiter les changements de configuration entre différentes simulations. L'intervention sur un algorithme n'est en effet pas envisageable pour tous les utilisateurs, car elle nécessite des connaissances poussées, et ce même si des points d'entrée adéquats sont fournis. Nous cherchons donc à rendre cette configuration la plus limitée possible. Introduire dans l'algorithme des éléments de contrôle de haut niveau permet de répondre à cette problématique.

Le second point concerne la reproductibilité des résultats. Suivant les besoins de l'utilisateur, celui-ci peut souhaiter avoir une génération totalement déterministe, afin d'être en mesure de reproduire la configuration initiale demandée, ou introduire différents degrés d'aléatoire. Si l'objectif est d'expérimenter des comportements émergents, il peut en effet être intéressant d'introduire des comportements non spécifiés, voire même potentiellement aberrants, afin d'étudier leur influence sur les résultats de la simulation. L'algorithme prend donc en compte ces besoins variés [12].

L'approche utilisée par Reynolds [Reynolds, 1999] pour le suivi de chemin flou, que nous avons vu dans le Chapitre 1, présente une certaine similitude avec les résultats que nous cherchons à obtenir. Le modèle qu'il introduit permet de contraindre faiblement la trajectoire des

12. Notons qu'en théorie il suffit de stocker les graines des générateurs aléatoires utilisés afin de reproduire les données à l'identique. En pratique cela reste très difficile : les générateurs et leur fonctionnement sont souvent dépendants de la plateforme, et la maîtrise du processus est insuffisante pour répondre au besoin dès lors que plusieurs générateurs aléatoires influent les uns sur les autres.

agents autour d'un segment spécifié par l'utilisateur, alors que nous cherchons à contraindre un paramétrage dans un espace donné. Cependant son objectif est d'améliorer le réalisme comportemental des agents en environnement ouvert en utilisant des briques comportementales simples. Son approche ne s'intéresse pas à l'écart au segment, aux éventuelles « sorties de route » ou aux contraintes sur plusieurs paramètres. Ainsi si l'approche globale peut sembler adéquate, les objectifs poursuivis ne permettent pas de répondre aux besoins que nous rencontrons.

Comme nous allons le voir nous nous inspirons cependant de l'approche utilisé par Reynolds, en introduisant un degré contrôlable d'aléatoire à chacune des étapes du processus.

2.3.1.2 Présentation de l'algorithme

Afin de répondre aux besoins présentés ci-dessus, nous avons construit un algorithme indépendant du domaine étudié, permettant de spécifier globalement le déterminisme et offrant la possibilité d'être contrôlé de manière fine [Lacroix et al., 2008a].

L'algorithme s'applique sur un élément de l'environnement $e \in \mathcal{E}$ associé à un ensemble $\mathcal{O}$ d'objets. e peut par exemple être un agent et $\mathcal{O}$ représenter ses paramètres, ou e peut être une règle de comportement et $\mathcal{O}$ contenir ses préconditions... Chaque élément $o \in \mathcal{O}$ est associé à un facteur de probabilité $p_o \in [0, 1]$, ainsi qu'à deux fonctions distinctes. La première, d_o, est un processus déterministe sur o et la seconde, nd_o, est une fonction quelconque, non nécessairement déterministe. A chaque pas de temps t, $\mathcal{O}_t \subset \mathcal{O}$ est l'ensemble des objets qui peuvent être sélectionnés. Enfin un facteur global de déterminisme $\gamma \in [0, 1]$ est spécifié : plus le facteur γ sera grand, plus il y aura de chances que du non-déterminisme soit utilisé au sein de l'algorithme.

L'algorithme fonctionne de la manière suivante (Alg. 4). Soit α le résultat du tirage d'une variable aléatoire réelle suivant la loi uniforme sur $[0, 1]$. Si α est inférieur à γ, le facteur global de déterminisme, alors du non-déterminisme peut être appliqué. Dans ce cas, pour chaque objet de e on tire un second nombre pseudo-aléatoire β depuis une distribution réelle uniforme sur $[0, 1]$, puis on compare β et le facteur de probabilité p_o de l'objet considéré. Si $\beta < p_o$ on applique la procédure non déterministe nd_o, sinon on applique la procédure déterministe. Finalement, dans le cas où α est supérieur à γ, l'algorithme est purement déterministe : la procédure d_o est utilisée pour chacun des objets de e.

Exemple 9 *Le fonctionnement de l'algorithme peut être illustré sur un exemple simple, celui du comportement de joueurs de football*[13]*. Pour ce faire, considérons que e est un agent footballeur. Dans le cadre de cet exemple, $\mathcal{O}$ ne contient qu'un élément o_1, « tirer », associé au facteur de probabilité p_{o_1}. La procédure déterministe d_{o_1} est la séquence $(droite, gauche)$ répétée : aux pas de temps pairs l'agent tire à droite, aux pas de temps impairs il tire à gauche. La procédure non-déterministe nd_{o_1} est le tirage aléatoire d'une valeur dans la séquence $(droite, gauche)$ avec les coefficients $(\frac{3}{4}, \frac{1}{4})$.*

Observons l'influence de γ et de p_o sur les probabilités de tirer à droite et à gauche. À un pas de temps impair d_{o_1} sélectionne la gauche, et on a alors :

$$\begin{cases} p_{\text{gauche}} &= \gamma \cdot (p_{o_1} \cdot \frac{3}{4} + (1 - p_{o_1}) \cdot 1) + (1 - \gamma) \cdot 1 \\ p_{\text{droite}} &= \gamma \cdot (p_{o_1} \cdot \frac{1}{4} + (1 - p_{o_1}) \cdot 0) + (1 - \gamma) \cdot 0 \end{cases}$$

13. Voir l'applet disponible sur `http://www2.lifl.fr/SMAC/projects/cocoa/football.html`

Algorithme 4 Processus générique d'instanciation d'un élément quelconque associé à des objets.

Entrées : e associé à $\mathcal{O}$. $p_o \in [0,1]$ facteur de probabilité de o, d_o et nd_o process sur o respectivement déterministe ou non. $\gamma \in [0,1]$ facteur global de déterminisme.

au pas de temps t : $\mathcal{O}_t \subset \mathcal{O}$ l'ensemble des objets disponibles
$\alpha \leftarrow$ uniform_random($[0,1]$)
si $\alpha < \gamma$ **alors**
    pour tout $o \in \mathcal{O}_t$ **faire**
        $\beta \leftarrow$ uniform_random($[0,1]$)
        si $\beta < p_o$ **alors**
            exécuter nd_o
        sinon
            exécuter d_o
        fin si
    fin pour
sinon
    pour tout $o \in \mathcal{O}$ **faire**
        exécuter d_o
    fin pour
fin si

$$\Leftrightarrow \begin{cases} p_{\text{gauche}} & = 1 - \gamma \cdot p_{o_1} \cdot \frac{1}{4} \\ p_{\text{droite}} & = \gamma \cdot p_{o_1} \cdot \frac{1}{4} \end{cases}$$

Si l'utilisateur fixe $\gamma = 0$, $p_{\text{gauche}} = 1$ *et* $p_{\text{droite}} = 0$ *aux pas de temps impairs, et l'inverse aux pas de temps pairs. Le joueur tirera toujours à gauche, puis à droite, et ainsi de suite : l'algorithme est déterministe. Par ailleurs, comme on peut le voir dans les équations, plus* γ *augmente et plus la probabilité de tirer à droite augmente. Finalement,* p_{o_1} *influence les probabilités de manière similaire, même si il joue un rôle limité dans un exemple simple comme celui-ci.*

Comme l'illustre l'exemple ci dessus, les paramètres γ et les p_{o_i} influencent la probabilité d'utilisation de la fonction non-déterministe associée à l'objet en cours. Avec plusieurs paramètres, les facteurs p_{o_i} permettent de diminuer la probabilité d'utilisation de la fonction non-déterministe indépendamment de γ, ce qui fournit des possibilités de contrôle fines.

2.3.1.3 Réponse aux besoins

Analysons maintenant dans quelle mesure cet algorithme permet de répondre aux différents critères spécifiés, et intéressons nous tout d'abord à la généricité et à la flexibilité. Nous avions déterminé qu'il était nécessaire que l'algorithme soit indépendant du domaine étudié, et qu'il soit facilement configurable. L'application à l'exemple des joueurs de football, puis au modèle de données (illustré dans la Section 2.3.2), permet d'établir la généricité du processus. Il est bien utilisable sur différents domaines d'application [14], et l'indépendance au domaine étudié est respectée.

Cette indépendance se fait cependant au détriment de la facilité de configuration, car le paramétrage rendu nécessaire par le critère de généricité reste relativement important. Il est

14. Voir la Section 3.4 et le Chapitre 4 pour d'autres applications de l'algorithme.

en effet nécessaire de spécifier γ, e et l'ensemble des éléments de $\mathcal{O}$, avec pour chacun une probabilité p_o et deux fonctions. En fait, comme nous allons le voir ci-dessous, une fois réalisé le couplage au modèle de données proposé, la configuration est immédiate : chacun de ces éléments est directement basé sur un des éléments du modèle. Dans les autres cas elle peut être adaptée largement en fonction des désirs et besoins des utilisateurs, ou même simplifiée. Par exemple en fixant $\forall o \in \mathcal{O},\ p_o = 1$ on se ramène à un algorithme plus simple, reposant uniquement sur γ.

En ce qui concerne la maîtrise du déterminisme, l'algorithme offre des possibilités variées. Tout d'abord, le facteur γ permet aux utilisateurs de spécifier de manière globale le comportement de l'algorithme. Ainsi, si $\gamma = 0$, il est purement déterministe : à chaque exécution, d_o, procédure elle même déterministe, est appliquée pour chacun des objets. Si $\gamma \neq 0$, on introduit la possibilité qu'à certaines exécutions de l'algorithme des procédures non-déterministes soient utilisées. Par ailleurs, plus γ augmente, et plus la probabilité d'utiliser les fonctions non-déterministes est élevée (α étant tiré uniformément entre 0 et 1). Ce critère introduit donc un premier élément de contrôle.

Le second élément intervient dès lors que la possibilité d'introduire du non-déterminisme est validée. Dans ce cas, chacun des objets de $\mathcal{O}$ a une chance d'appliquer sa procédure non déterministe nd_o. Cette chance est limitée par le paramètre p_o. Il n'est donc pas possible de savoir à l'avance quels sont les objets qui vont appliquer leur procédure non-déterministe, ou combien d'entre eux vont le faire. Ce second niveau d'aléatoire induit par le tirage de β implique que même si le choix du non-déterminisme a été fait à la première étape, seuls certains des objets, sélectionnés eux aussi aléatoirement, appliqueront la procédure non-déterministe fournie par le concepteur. Ce mécanisme permet d'obtenir des tirage très variés, tout en contrôlant la « quantité » de non-déterminisme qui va être introduite : par exemple, si les p_o sont faibles il y aura peu de chance que leur procédure nd_o soit utilisée. Le mécanisme permet en outre à l'utilisateur d'équilibrer finement (avec les p_o) entre des objets risquant de produire des résultats inattendus ou non souhaités (à cause de nd_o), et d'autres qu'au contraire il souhaite voir varier.

Ce mécanisme peut ainsi être vu comme un « générateur aléatoire de comportements aléatoires », si on assimile l'expression des objets de $\mathcal{O}$ au comportement de l'élément e.

2.3.2 Application à la structure de données

L'utilisation du modèle de données présenté dans la Section 2.2 nécessite l'instanciation des normes en agents du modèle, qui sont utilisés pour affecter un paramétrage aux agents de la simulation. Nous allons voir comment l'algorithme proposé permet de réaliser cette instanciation, et quelles possibilités sont ainsi offertes au modèle.

2.3.2.1 Instanciation des normes

L'objectif est d'obtenir des agents variés, mais utilisant des valeurs conformes. Pour ce faire, nous nous appuyons sur les spécifications du modèle de données.

La génération des agents se base sur leur norme de référence. Chacune de ces normes est associée à un ensemble de paramètres, et ces paramètres spécifient un domaine de définition associé à une distribution de probabilité sur ce domaine. Ces éléments sont utilisés afin de générer les agents. Notons que l'ensemble de valeurs obtenu ne sera cohérent que dans la mesure où la spécification elle-même est cohérente : on ne sera en mesure de garantir que la conformité de l'instanciation (cf. Sect. 3.2 p. 82).

Par ailleurs, afin d'augmenter la variété des agents, nous souhaitons être en mesure de générer certains des paramètres en dehors de leur domaine de définition. Pouvoir garantir une certaine conformité par rapport au spécifications est toutefois souhaitable, afin de ne pas créer de jeux de valeurs inadéquats, voire absurdes. Les paramètres de référence spécifiés seront utilisés à cet effet.

Le mécanisme souhaité est donc le suivant : pour chaque paramètre p de la norme N instanciée, on utilise le domaine de définition $\mathcal{D}_p$ afin de générer une valeur en utilisant la distribution statistique g_p. Cependant, afin d'augmenter la variété, le non-respect de cette règle est parfois autorisée. Dans ce cas, le tirage de la valeur est réalisé non pas en utilisant les éléments définis dans p, mais ceux définis dans p_{ref}, qui englobent ceux de p. Le facteur σ contenu dans l'institution, ainsi que τ_N et $\delta_{\max_N}$ contenus dans la norme sont utilisés afin de contrôler la fréquence et l'amplitude de ces écarts.

2.3.2.2 Utilisation de l'algorithme

L'application de l'algorithme générique au modèle de données présenté se fait en l'appliquant au niveau de la norme. Les paramètres de l'algorithme sont identifiés à ceux du modèle de données de la manière suivante :

- $e \leftarrow N$: l'élément auquel l'algorithme s'applique est une norme,
- $\mathcal{O} \leftarrow \mathcal{P}_N$: les objets associés à l'élément sont les paramètres contenus dans la norme. Par ailleurs, aucun filtrage temporel n'est effectué sur les paramètres : à chaque instant $\mathcal{O}_t = \mathcal{O}$,
- $\forall o \in \mathcal{O},\ p_o \leftarrow \tau_N$: tous les paramètres utilisent le même facteur de probabilité, qui est le taux de violation de la norme, afin de simplifier la configuration. Il serait également possible de générer automatiquement une distribution de p_o différents, en utilisant par exemple σ comme graine du générateur aléatoire. Ce processus pourrait cependant être contre intuitif pour les utilisateurs : nous ne l'avons pas appliqué afin de préserver la simplicité du mécanisme,
- $d_o \leftarrow g_p$: le processus déterministe associé à chaque objet est la distribution g_p. Le processus est donc en fait pseudo-aléatoire, mais nous utilisons cette analogie afin d'exprimer le fait que le résultat provenant de g_p est « attendu » par l'utilisateur. En effet il l'a configuré lui même et les valeurs obtenues restent dans les limites du domaine de définition spécifié dans la norme,
- $nd_o \leftarrow g_{p_{\text{ref}}}$: le processus non-déterministe associé aux objets est la distribution $g_{p_{\text{ref}}}$, qui correspond donc au paramètre de référence de p. Il s'agit également d'un processus pseudo-aléatoire, mais les résultats obtenus peuvent cette fois sortir du domaine de la norme, et donc produire des résultats inattendus,
- $\gamma \leftarrow \sigma$: le critère global de déterminisme défini au niveau de l'institution est utilisé pour contrôler le déterminisme de l'algorithme.

Cette identification des paramètres conduit à l'algorithme 5. Ici les critères déclenchant l'utilisation des processus non-déterministes sont d'une part σ au niveau global, et d'autre part τ_N au niveau de chacun des paramètres. Si τ_N est élevé, le concepteur souhaite que la norme ait plus de chances d'être violée. En effet, si $\tau_N > \alpha$ la valeur du paramètre est tirée en utilisant la fonction $g_{p_{\text{ref}}}$, qui a valeur dans $\mathcal{D}_{p_{\text{ref}}}$ et non $\mathcal{D}_p$. Comme $\mathcal{D}_p \subset \mathcal{D}_{p_{\text{ref}}}$, il est possible que la valeur tirée ne soit pas dans $\mathcal{D}_p$: on peut avoir des violations. Si $\tau_N \leq \alpha$, la valeur est tirée en utilisant g_p, et donc $v_p \in \mathcal{D}_p$: ce paramètre ne sera pas la source d'une violation.

Algorithme 5 Algorithme générique appliqué à la structure de données.

Entrées : une norme $N = \{N,\ N_{\text{ref}},\ \mathcal{P}_N,\ \mathcal{Q}_N,\ \tau_N,\ \delta_{\max_N}\}$;
σ le critère global de déterminisme de l'institution.

```
1: tirer une valeur α depuis une distribution aléatoire sur [0,1]
2: si α < σ alors
3:     pour tout p ∈ P_N faire
4:         tirer une valeur β depuis une distribution aléatoire sur [0,1]
5:         si β < τ_N alors
6:             exécuter g_{p_ref}
7:         sinon
8:             exécuter g_p
9:         fin si
10:    fin pour
11: sinon
12:    pour tout p ∈ P_N faire
13:        exécuter g_p
14:    fin pour
15: fin si
```

Cette façon de générer les valeurs des paramètres fait donc intervenir différents degrés d'aléatoire, qui permettent de créer des ensembles variés à partir d'un jeu de configuration minimal. De plus le contrôle est très fin : on contrôle tout d'abord la chance que le paramètre en cours de tirage viole la norme, puis lors du tirage de la valeur il est encore possible d'échapper à la violation, car $\mathcal{D}_p \subset \mathcal{D}_{p_{\text{ref}}}$. Les valeurs sont donc variées, mais sans s'éloigner du profil défini, représenté ici par la norme de référence. De cette façon les comportements sont lissés autour de la norme, tout en multipliant les combinaisons disponibles.

Exemple 10 *Reprenons l'exemple 9 afin d'illustrer l'utilisation de l'algorithme. Les joueurs de football sont des agents, dont le modèle de décision s'appuie sur un paramètre décrivant la propension à tirer à gauche ou à droite, et qui est pris en compte dans la prise de décision (en plus d'autres facteurs comme par exemple la présence de défenseurs, du gardien, la position sur le terrain...). Ce paramètre, que nous noterons d (pour direction), est fixé par l'utilisateur et peut aller de la valeur* -1 *(complètement à gauche) à la valeur* 1 *(complètement à droite). Nous souhaitons le générer à l'aide du modèle proposé afin d'obtenir automatiquement des joueurs ayant un comportement de tir varié.*

Dans ce cadre, en utilisant les notations introduites dans la Section 2.2, $\mathcal{A} = \{\text{joueurs}\}$*, et* $\mathcal{P} = \mathcal{P}_{e,s} = \{d\}$*. On définit* d_{init} *et* d_1 *deux paramètres (Tab. 2.1), puis la norme de référence ainsi qu'une norme « tireur à gauche » :*

$$N_{\text{init}} = \{N_{\text{init}},\ \emptyset,\ d_{\text{init}},\ \emptyset\}, \qquad N_{\text{gauche}} = \{N_{\text{gauche}},\ N_{\text{init}},\ d_{\text{gauche}},\ \emptyset,\ \tau_{N_{\text{gauche}}},\ \delta_{\max_{N_{\text{gauche}}}}\}$$

L'algorithme s'applique ainsi naturellement pour générer toute une variété d'agents ayant tendance à tirer à gauche.

Exemple 11 *De la même façon, avec la simulation de piéton introduite à la Section 2.2, la norme* N_{normal} *permet de générer des agents dont la vitesse maximale est comprise entre 4 et*

p	d_{init}	d_{gauche}
l_p	d_{init}	d_{gauche}
p_{ref}	0 (d est le paramètre racine)	d_{init}
$\mathcal{D}_p$	$[-1, 1]$	$[-1, 0]$
v_d	0	$-\frac{1}{2}$
g_p	distribution uniforme sur $[-1, 1]$	distribution uniforme sur $[-1, 0]$
f_p	fonction par défaut	fonction par défaut

TABLE 2.1 – Définition de deux paramètres dans la sémantique du modèle.

6 km/h, correspondant à $\mathcal{D}_{v_{max,normal}}$, et ayant tous pour espace personnel la valeur normal. Enfin, comme $\tau_{N_{\text{normal}}} = 0$, il ne peut y avoir de violation.

L'algorithme 5 décide donc de l'application de g_p ou de $g_{p_{\text{ref}}}$. Implémenté au sein d'une procédure (Alg. 6), il permet d'obtenir les agents désirés à partir des normes spécifiées.

Cette procédure permet par ailleurs d'ajouter un contrôle supplémentaire, basé sur l'utilisation de l'écart maximal à la norme $\delta_{\max_N}$. En pratique, après initialisation de l'agent s, l'algorithme 5 est appliqué et les valeurs obtenues par l'intermédiaire des fonctions g_p et $g_{p_{\text{ref}}}$ sont stockées dans l'agent. On teste alors le respect de la norme de référence, à l'aide de la procédure *quantify_violation(s)* (cf. Alg. 2). Si la violation excède l'écart maximal à la norme $\delta_{\max_N}$, on rejette l'agent généré et on réitère la procédure. L'utilisation de ce critère supplémentaire permet d'obtenir une conformité à la norme correspondant aux valeurs spécifiées. Couplée avec l'utilisation des facteurs τ_N, cela procure un contrôle complet sur le processus de génération.

Afin de garantir l'arrêt de la procédure, un agent respectant la norme est généré après un certain nombre d'essais ratés (fixé empiriquement à 10). En effet, certains jeux de paramètres peuvent rendre difficile l'obtention de jeux de paramètres en adéquation avec $\delta_{\max_N}$. Par exemple, si le paramètre en violation est un singleton alors que le domaine de son paramètre de référence est un large intervalle, la violation sera souvent importante, et probablement rejetée par l'algorithme. Notons que ce contournement biaise le processus de génération car τ_N ne représente alors plus la proportion d'agents qui tentent de violer la norme, mais qu'il s'agit d'une méthode efficace pour assurer son bon fonctionnement.

2.3.2.3 Génération de la déviance

Le couplage du modèle de données à l'algorithme présenté permet d'obtenir différents types d'agents modèle. Trois d'entre eux sont à distinguer (Fig. 2.7).

Le premier type est un agent « conforme » à la norme. Il est obtenu en instanciant uniquement des paramètres statiques pour les agents, et en leur affectant des valeurs comprise dans les domaines de définition de la norme de référence. N'ayant pas de possibilité d'évoluer en dehors de ce domaine car statiques, ils respecteront toujours la norme. Ce type d'agent s_{comp} vérifie donc $\forall c \in \mathcal{C}_{s_{\text{comp}}},\ v_{s_{\text{comp}_c}} \in \mathcal{D}_p$ où $p \in N_{G_{s_{\text{comp}}}}$ avec $l_{s_{\text{comp}_c}} = l_p$.

Exemple 12 *Un agent instancié à partir de la norme N_{normal} est « conforme ». En effet, $\tau_{N_{\text{normal}}} = 0$ donc aucune violation n'est autorisée, et $v_{\text{max,normal}}$ ainsi que s_{normal} sont des paramètres statiques.*

Algorithme 6 Procédure d'instanciation de la norme : **instantiate_norm**($N,\ \sigma$).

Entrées : une norme $N = \{N,\ N_{\text{ref}},\ \mathcal{P}_N,\ \mathcal{Q}_N,\ \tau_N,\ \delta_{\max_N}\}$;
σ critère global de déterminisme de l'institution.
Sorties : s un agent modèle

assigner à s sa norme de référence : $N_s \leftarrow N$
répéter
    appliquer l'Alg. 5
    tirer une valeur α depuis une distribution aléatoire sur [0,1]
    si $\alpha < \sigma$ **alors**
        pour tout $p \in P_N$ **faire**
            tirer une valeur β depuis une distribution aléatoire sur [0,1]
            si $\beta < \tau_N$ **alors**
                violation potentielle, $v_p \in \mathcal{D}_{p_{\text{ref}}} : v_p \leftarrow g_{p_{\text{ref}}}$
            sinon
                pas de violation, $v_p \in \mathcal{D}_p : v_p \leftarrow g_p$
            fin si
            sauvegarder la valeur obtenue dans $s : v_p(s) \leftarrow v_p$
        fin pour
    sinon
        pour tout $p \in \mathcal{P}_N$ **faire**
            pas de violation, $v_p \in \mathcal{D}_p : v_p \leftarrow g_p$
            sauvegarder la valeur obtenue dans $s : v_p(s) \leftarrow v_p$
        fin pour
    fin si
jusqu'à quantify_violation$(s) < \delta_{\max_N}$ {cf. Alg. 2}
retourner s

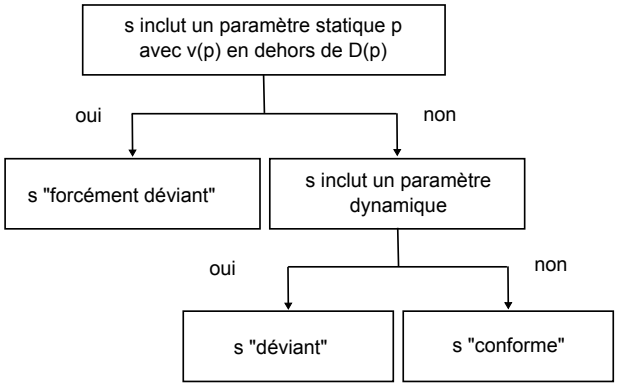


FIGURE 2.7 – Classification de la déviance en fonction des paramètres de la norme.

Le second type d'agent est un agent « forcément déviant ». Il est basé sur des paramètres statiques du modèle, mais est en violation dès sa création : au moins un des paramètre le constituant prend sa valeur en dehors du domaine de définition spécifié dans la norme de référence (cf. Définition 8). Les paramètres étant utilisés de manière statique dans le modèle, l'agent n'aura pas possibilité de revenir au sein de la norme et restera en violation pendant toute la durée de la simulation. Ce type d'agent s_{fdev} vérifie à tout instant la définition 8, $\exists c \in \mathcal{C}_{s_{\text{fdev}}},\ v_{s_{\text{fdev}_c}} \notin \mathcal{D}_p$ où $p \in N_{G_{s_{\text{fdev}}}}$ avec $l_{s_{\text{fdev}_c}} = l_p$.

Exemple 13 *Définissons la norme $N_{\text{normal}'}$, identique à N_{normal} sauf pour le taux d'écart à la norme : $\tau_{N_{\text{normal}'}} = 1$. Supposons qu'un agent a soit instancié à partir de cette norme. D'après les définitions, le paramètre de référence de $v_{\text{max,normal}}$ est v_{max}, de domaine de définition $[0, 20]$ km/h qui est donc le domaine depuis lequel la valeur peut être choisie. On suppose que la procédure 6 affecte à a la valeur $v_{\text{max}}(a) = 12$ km/h. Alors a est « forcément déviant », car v_{max} est un paramètre statique et à tout instant on aura $v_{\text{max}}(a) \notin \mathcal{D}_{v_{\text{max,normal}}} = [4, 6]$ km/h.*

Enfin le troisième type d'agent est « déviant ». À la différence des deux autres, il est basé sur au moins un paramètre dynamique du modèle. Il n'y a pas de contrainte sur son état initial : l'agent pourra être créé déjà en violation, ou non. Notons que s'il inclut également des paramètres statiques et qu'au moins un de ceux-ci est instancié en dehors du domaine de définition de la norme, alors il fera partie de la catégorie « forcément déviant ». Si au contraire un paramètre dynamique est instancié en dehors du domaine de définition, il a la possibilité d'évoluer et donc de retourner au sein du domaine.

Exemple 14 *Supposons qu'il existe une norme N_{desire} dont un des paramètres soit la vitesse désirée des agents. Alors un agent instancié à partir de cette norme sera « déviant » : la vitesse désirée est un paramètre dynamique de la simulation.*

Notons qu'un agent déviant pourra ne jamais être en violation, malgré l'évolution des paramètres dynamiques : si les violations sont interdites par l'utilisateur, le contrôle par les normes imposera à la simulation de ne pas dépasser les bornes spécifiées par l'utilisateur.

2.4 Conclusion

Dans ce second chapitre, nous nous sommes intéressés aux approches normatives et avons introduit les deux premiers axes de notre modèle. Formalisées historiquement dans les domaines des sciences juridiques et des sciences sociales, les normes ont éveillé l'intérêt de la communauté des systèmes multi-agents. En effet, elles offrent la possibilité de spécifier des contraintes sur les actions des agents, et fournissent une structure de référence permettant à ceux-ci d'interagir dans un cadre spécifié. Elles apportent ainsi des solutions à des problèmes de coopération, comme par exemple la navigation de robots, et fournissent des éléments pour la construction de systèmes multi-agents ouverts, dans lesquels à la fois les agents, mais aussi l'organisation, peuvent s'adapter pour remplir un objectif global.

Notre approche se place dans une perspective différente. Nous proposons d'utiliser les normes pour construire des profils comportementaux pour les agents : les normes contiennent la spécification de leur paramétrage, et fournissent une structure de référence qui peut être utilisée à l'exécution pour vérifier leur conformité. Notre définition des normes s'appuie sur la construction de paramètres reflétant ceux de la simulation, spécifiant leurs domaines de définition et les possibilités de déviation autorisées par le concepteur. En utilisant ces éléments, nous proposons des algorithmes permettant de construire un paramétrage pour les agents de la simulation. Les normes spécifiées peuvent être violées : nous présentons les mécanismes permettant de détecter, quantifier et réagir à ces violations.

Enfin, nous proposons dans la dernière partie un algorithme générique permettant de contrôler l'introduction d'éléments aléatoires dans un processus d'instanciation. Cet algorithme peut être considéré comme un « générateur aléatoire de comportements aléatoires », et permet de générer automatiquement des combinaisons de paramètres. En le couplant à notre structure de données, nous utilisons cet algorithme pour générer les valeurs du paramétrage des agents. De cette façon, la différenciation comportementale au sein des normes est introduite naturellement et est maîtrisée de manière fine grâce aux paramètres disponibles.

Chapitre 3

Variété et conformité

La première partie de ce chapitre présente le dernier axe de notre modèle, qui concerne l'observation et l'analyse du système. Nous précisons tout d'abord les motivations qui ont conduit à l'intégration de ces éléments, puis décrivons la méthode utilisée. Nous nous intéressons ensuite à ses applications : inférence de normes, classification du comportement des agents et automatisation de la configuration. Dans un second temps, nous montrons comment le modèle permet d'introduire de la variété dans la simulation et de contrôler la conformité des comportements. La troisième partie décrit les principaux modes d'utilisation de l'outil. En particulier, son caractère non intrusif le rend utilisable pour de nombreuses applications. Enfin, nous illustrons sa généricité en présentant des exemples dans différents domaines : la simulation de foules, et la création automatisée de créatures d'espèces variées.

3.1 Gestion du système

Dans cette première section, nous présentons tout d'abord les motivations qui ont conduit à développer cette partie du modèle. Nos besoins nous ont dirigé vers l'utilisation de réseaux de Kohonen, dont nous décrivons la mise en œuvre. Nous détaillons enfin leur utilisation pour résoudre nos différentes problématiques.

3.1.1 Introduction

Nous introduisons tout d'abord les motivations qui ont conduit au développement d'un axe concernant l'observation et l'analyse du système. Nous présentons ensuite différentes méthodes de classification non supervisée, et les raisons qui nous ont fait choisir les réseaux de Kohonen pour répondre à nos besoins.

3.1.1.1 Motivations

Les deux premiers axes du modèle proposé, présentés dans le Chapitre 2, permettent de décrire et de générer les comportements en maîtrisant leur conformité avec les spécifications. Plusieurs fonctionnalités apportent un complément pertinent à ces deux axes. Tout d'abord, une fois un comportement généré et affecté à un agent, la quantification de l'écart à la norme permet de contrôler son évolution. Cependant, étant pris en charge par la simulation, le comportement

de l'agent peut évoluer vers une configuration ne correspondant pas à sa norme initiale : s'il en est informé, l'utilisateur peut tenir compte de cet élément pour améliorer le paramétrage. Par ailleurs, l'observation de la population des agents peut permettre de détecter l'émergence de normes : la configuration initiale ne reflète en effet pas nécessairement les normes correspondant en pratique au comportement des agents. Enfin, le modèle nécessite que des ensembles de définition soient fournis pour chacun des paramètres dans chacune des normes. Il serait particulièrement intéressant que cette calibration initiale soit apprise à partir de simulations antérieures, voire de données réelles si on en dispose. Cela faciliterait la configuration du modèle.

Différentes problématiques doivent par ailleurs être prises en compte. Tout d'abord, le mécanisme offrant ces fonctionnalités ne doit pas nécessiter de configuration spécifique au domaine, afin de préserver la généricité du modèle. Ensuite, il doit être automatisé, pour limiter le besoin de supervision et permettre son exploitation par un maximum d'utilisateurs. Une fois les données construites suivant le format spécifié, l'utilisateur ne doit pas avoir à intervenir. Enfin, les données proviennent soit d'observations réelles, soit d'enregistrements de la simulation. Elles pourront donc être de qualité très variable : le mécanisme doit être robuste aux erreurs.

Pour répondre à ces problématiques, nous avons choisi d'utiliser des techniques de classification non supervisée. En effet, ce type d'algorithme ne nécessite pas de retour de l'utilisateur lors de la classification : il regroupe simplement les données similaires. L'intervention d'un expert n'est pas nécessaire pour valider les catégories inférées, mais permettra de libeller les catégories si on le souhaite. Par ailleurs, ces algorithmes fournissent une solution générique : en construisant les entrées à partir des paramètres utilisés par les agents, et la sortie en correspondance avec la structure des normes, on obtient une solution utilisable par la plupart des applications. Enfin, ils sont robustes, car les données bruitées seront classées dans les catégories existantes, sans créer d'erreurs.

3.1.1.2 Méthodes de classification non supervisée

De nombreuses méthodes de classification non supervisée ont été développées [Gallant, 1993]. Nous en présentons brièvement trois exemples.

Dans l'algorithme des k-means, le nombre k de catégories souhaitées est choisi à l'avance [15]. Chaque catégorie est associée à un centroïde c décrivant l'ensemble des N points $\{E^j\}$ de la catégorie. c est défini comme le vecteur dont chaque composante est la moyenne des valeurs de cette composante parmi les exemples de la catégorie : $c = \frac{1}{N} \cdot \sum E^j$. Lors de la classification, chaque exemple est déplacé dans la catégorie dont le centroïde est le plus proche, puis les centroïdes de la catégorie d'arrivée et de celle de départ sont mis à jour avec ce nouvel exemple. L'algorithme se termine lorsqu'une passe sur l'ensemble des exemples ne produit plus de nouvelles affectations. Cet algorithme est rapide, et est intéressant si le critère ciblé est de minimiser la somme des carrés des distances entre les exemples et leurs centroïdes.

Les algorithmes ART (pour Adaptive Resonance Theory), développés initialement par Gail Carpenter et Steve Grossberg [Carpenter & Grossberg, 1987], sont des algorithmes non supervisés basés sur des réseaux de neurones. Une de leur particularité est que le nombre de catégories n'est pas fixé initialement. Le système limite ainsi l'influence d'entrées qui pourraient perturber l'entraînement passé : l'algorithme ne permet à une entrée de modifier une catégorie que si elle en est suffisamment proche, sinon il en crée une nouvelle. Le nombre de catégories n'étant pas spécifié, un paramètre de vigilance contrôle leur granularité. L'algorithme repose sur le principe

15. Nous présentons ici le *convergent k-means clustering* [Anderberg, 1973].

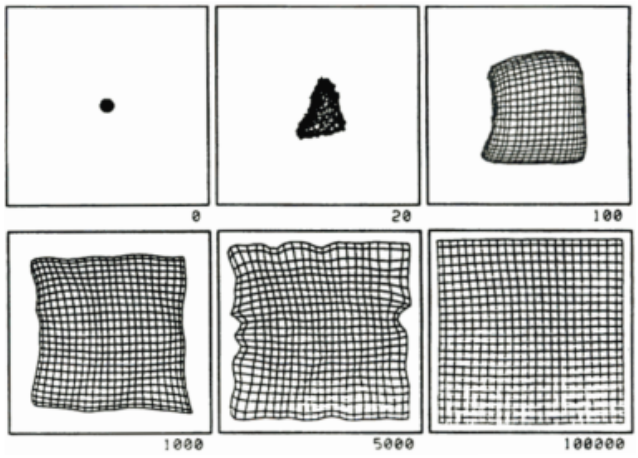


FIGURE 3.1 – Une grille de neurones s'auto-organisant suivant un carré (d'après [Kohonen, 1995]).

suivant : si une entrée est suffisamment proche du prototype d'une catégorie, et que le paramètre de vigilance est respecté, elle est placée dans la catégorie et le prototype est ajusté ; sinon, un autre prototype est essayé ; s'ils ont tous été testés, une nouvelle catégorie est créée.

Enfin, les cartes auto-adaptatives, proposées initialement par Teuvo Kohonen [Kohonen, 1995], sont un autre type de réseau de neurones destiné à la classification non supervisée. L'idée fondamentale est de considérer les neurones comme possédant une position dans l'espace, et donc un voisinage au sens spatial. L'objectif de l'algorithme est que le comportement des neurones soit cohérent avec leur structure spatiale, c'est-à-dire que des entrées proches déclenchent des neurones proches. Afin d'obtenir ce résultat, les neurones du voisinage du gagnant sont tous activés lors de l'entraînement, et ajustés avec un poids fonction de la distance dans le voisinage. Cela permet d'obtenir une représentation topologique des données : on peut par exemple représenter ainsi des figures géométriques (Fig. 3.1). Un usage classique de ce type de réseau est l'apprentissage du langage : sa structure est bien adaptée pour assimiler des différences de prononciation. Le principal avantage des cartes auto-adaptatives est qu'elles permettent la visualisation de données complexes dans une représentation à deux dimensions.

Les réseaux de type ART présentent l'avantage de ne pas avoir à fixer a priori le nombre de catégories. Cependant, le paramètre de vigilance dépend fortement du type de données, et limite l'automatisation de l'approche. Pour les réseaux de Kohonen, la représentation spatiale des données permet aux utilisateurs de visualiser plus facilement les résultats, et de les regrouper intuitivement. De plus, ils permettent de définir une distance qui peut être fixée par l'utilisateur. Ces points nous ont conduit à choisir ce type de réseau, qui permettent en outre de minimiser la configuration pour les usages souhaités, tout en offrant des possibilités d'extension pour les utilisateurs souhaitant les exploiter sur d'autres applications.

3.1.2 Les cartes auto-adaptatives ou réseaux de Kohonen

Les réseaux de Kohonen constituent la base du mécanisme d'observation et d'analyse de notre modèle. Dans cette partie, nous présentons leur fonctionnement, l'algorithme d'apprentissage et un exemple l'illustrant.

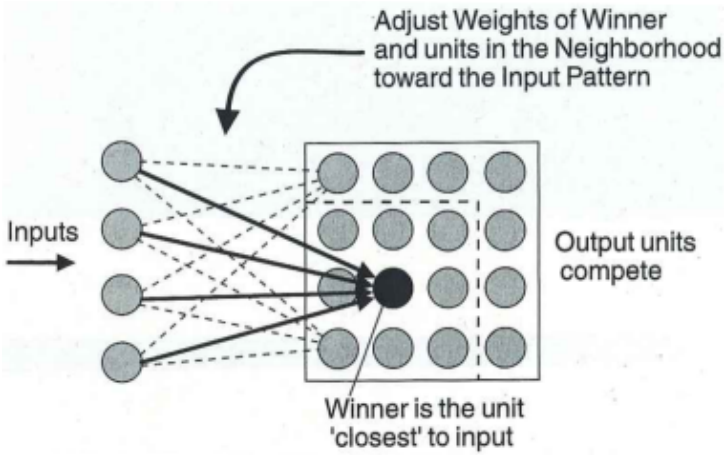


FIGURE 3.2 – Exemple de structure d'un réseau de Kohonen.

3.1.2.1 Fonctionnement

Les réseaux de Kohonen sont des réseaux de neurones simple couche, comportant une couche d'entrée et une couche de sortie (Fig. 3.2). A chaque fois qu'un vecteur d'entrée est présenté au réseau, sa distance à chaque neurone dans la couche de sortie est calculée. Celui dont la distance au vecteur d'entrée est la plus faible est déclaré gagnant. Ce neurone et ceux de son voisinage sont alors ajustés en déplaçant leur poids vers le vecteur d'entrée. Le réseau s'auto-organise, et construit progressivement une carte topologique des entrées. Une des propriétés de cette carte est que les vecteurs qui sont proches dans l'espace d'entrée (au sens de la distance utilisée) correspondent à des neurones qui sont proches dans la couche de sortie du réseau (au sens du voisinage spatial).

Une des caractéristiques du réseau est le type de topologie choisie pour la représentation spatiale des neurones. Typiquement, elle sera soit rectangulaire (matrice à deux dimensions), soit hexagonale. Il est cependant possible d'utiliser tout type de structure : un graphe, une matrice multidimensionnelle... Par exemple, pour la reconnaissances d'images en trois dimensions, on utilise une topologie qui est également en trois dimensions.

Une autre caractéristique importante est la définition du voisinage des neurones. Pour permettre leur spécialisation, il faut utiliser une fonction décroissante du temps : initialement, le voisinage est important et de nombreux neurones sont ajustés à chaque itération. Il est ensuite réduit au cours de l'entraînement. Deux possibilités sont classiquement utilisées : spécifier un ensemble de voisins pour chaque point, et le restreindre avec le nombre d'itérations ; ou utiliser une fonction Gaussienne qui se contracte progressivement. Notons que la distance utilisée peut être choisie par l'utilisateur, par exemple la distance Euclidienne ou la distance de Levenstein.

En ce qui concerne l'initialisation du réseau, celle-ci peut se faire avec des valeurs aléatoires. En effet, des vecteurs initialement désordonnés sont ordonnés par l'algorithme, généralement après quelques centaines d'itérations (cf. [Kohonen, 1995], p. 115). Cependant, une telle politique n'est pas nécessairement la meilleure ou la plus rapide. Une autre possibilité est d'initialiser les vecteurs en utilisant des exemples choisis aléatoirement, ou encore d'initialiser les neurones de manière déjà ordonnée, avec des valeurs provenant de l'espace des vecteurs propres des exemples. L'algorithme débute ainsi avec des paramètres plus précis, ce qui permet une convergence plus rapide.

Algorithme 7 Algorithme d'entraînement d'une carte auto-adaptative (réseau de Kohonen).

1: **initialisation :** initialiser le poids de tous les neurones de sortie
2: **répéter**
3: sélectionner un exemple E^k
4: déterminer le neurone de sortie u_j dont les poids W_j sont les plus proches de E^k
5: ajuster le poids des neurones :
$W_i = W_i + \alpha(t) \cdot C_{ij}(t) \cdot d_j$
avec $\alpha(t)$ le taux d'apprentissage, $C_{ij}(t)$ la fonction de voisinage et d_j la distance entre W_j et E^k
6: $t = t + 1$
7: **jusqu'à** $t = T$

Enfin, une étape optionnelle de calibration de la carte peut être réalisée. Il s'agit d'étiqueter les neurones par des labels, en mode dirigé. Pour la réaliser, on utilise des exemples provenant de classes thématiques connues. En les entrant dans le réseau, ils permettent de détecter les neurones gagnants et de leur affecter le label du thème de l'exemple correspondant. Les classes thématiques seront alors décrites par un ou plusieurs neurones appartenant à la même région de la carte, et qui correspondent à un ou plusieurs vecteurs prototypes.

3.1.2.2 Algorithme

L'algorithme fonctionne de la manière suivante (Alg. 7). Pour illustrer le mécanisme, considérons l'exemple d'un réseau de topologie rectangulaire. Chaque neurone u_j de la matrice possède un vecteur de poids W_j. Tout d'abord, l'entrée E^k est présentée à la première couche de neurones, qui est simplement destinée à stocker les composantes du vecteur. Ensuite, la distance entre E^k et chaque neurone de sortie est calculée, en utilisant la distance choisie (ici la distance Euclidienne) :

$$d_j = \|E^k - W_j\|$$

Le neurone j dont la distance d_j est la plus faible est déclaré gagnant. Le poids du neurone gagnant et des neurones de son voisinage sont alors ajustés :

$$W_i(t+1) = W_i(t) + \alpha(t) \cdot C_{ij}(t) \cdot d_j$$

où $\alpha(t)$ est le taux d'apprentissage à l'itération t et $C_{ij}(t)$ est la valeur de la fonction de voisinage entre les neurones i et j. La clef de l'algorithme est que le poids des neurones voisins est également modifié, et pas seulement celui du neurone gagnant : sans cela il n'y aurait pas de relation topologique entre les neurones.

A chaque itération, les ajustements sont pondérés par le taux d'apprentissage $\alpha(t)$. Il est choisi par l'utilisateur, et doit être une fonction décroissante du nombre d'itérations afin de permettre une stabilisation progressive des neurones. On peut par exemple utiliser :

$$\alpha(t) = \alpha_{\text{initial}} \cdot \left(\frac{\alpha_{\text{final}}}{\alpha_{\text{initial}}} \right)^{\frac{t}{T}}$$

où T est le nombre maximal d'itérations. Ici $\alpha(t)$ décroît avec l'augmentation du nombre d'itérations. Des valeurs classiques pour α_{initial} et α_{final} sont respectivement 1.0 et 0.05.

h \ i	1	2	3
1	(−3, +2)	(+5, +4)	(+7, −6)*
2	(−5, −5)	(+1, 0)	(−1, +3)
3	(+4, −2)	(−3, −3)	(+5, −2)

TABLE 3.1 – Poids des vecteurs des neurones à la 500e itération.

Enfin, la fonction $C_{ij}(t)$ décrit le voisinage des neurones, et peut également être choisie par l'utilisateur. Il s'agira par exemple d'une fonction Gaussienne :

$$C_{ij}(t) = \exp\left(-\frac{\|i-j\|^2}{2\cdot\sigma(t)^2}\right)$$

où i et j sont les coordonnées des neurones dans la carte à deux dimensions et $\sigma(t)^2$ est la largeur de la fonction de voisinage. $C_{ij}(t)$ est construite pour être large lorsque t est petit, puis décroît jusqu'à une valeur ne comprenant plus qu'un neurone lorsque t atteint sa valeur maximale.

3.1.2.3 Exemple

Afin d'illustrer le fonctionnement, voici la présentation d'une itération d'un problème hypothétique [16]. Considérons une matrice 3x3 de neurones, de topologie rectangulaire. On utilise la distance euclidienne, et on prend pour taux d'apprentissage la fonction $\alpha(t)$ ayant pour définition :

$$\alpha(t) = \left(1 - \frac{t-1}{T}\right)$$

avec T le nombre total d'itérations.

On considère l'itération 501 sur un total de 1000. Les poids sont représentés dans le Tableau 3.1. À cette itération, le voisinage d'un neurone est constitué des 8 neurones adjacents (gauche, droite, haut, bas et les 4 diagonales).

Lors de cette itération, l'exemple sélectionné est le vecteur (6, −5). Le neurone $u_{1,3}$, marqué par une * dans le tableau, est le plus proche de cette valeur. Ses voisins sont les cellules à gauche, en dessous et en diagonale en bas à gauche ($u_{1,2}$, $u_{2,3}$ et $u_{2,2}$), et $\alpha(501) = \frac{1}{2}$. L'ajustement des poids des neurones se calcule de la manière suivante :

$$\begin{aligned}
W_{1,2} &= \begin{pmatrix}+5\\+4\end{pmatrix} + \frac{1}{2}\cdot\left(\begin{pmatrix}+6\\-5\end{pmatrix} - \begin{pmatrix}+5\\+4\end{pmatrix}\right) = \begin{pmatrix}+5.5\\-0.5\end{pmatrix}\\
W_{1,3} &= \begin{pmatrix}+7\\-6\end{pmatrix} + \frac{1}{2}\cdot\left(\begin{pmatrix}+6\\-5\end{pmatrix} - \begin{pmatrix}+7\\-6\end{pmatrix}\right) = \begin{pmatrix}+6.5\\-5.5\end{pmatrix}\\
W_{2,2} &= \begin{pmatrix}+1\\0\end{pmatrix} + \frac{1}{2}\cdot\left(\begin{pmatrix}+6\\-5\end{pmatrix} - \begin{pmatrix}+1\\0\end{pmatrix}\right) = \begin{pmatrix}+3.5\\-2.5\end{pmatrix}\\
W_{2,3} &= \begin{pmatrix}-1\\+3\end{pmatrix} + \frac{1}{2}\cdot\left(\begin{pmatrix}+6\\-5\end{pmatrix} - \begin{pmatrix}-1\\+3\end{pmatrix}\right) = \begin{pmatrix}+2.5\\-1\end{pmatrix}
\end{aligned}$$

Les autres poids sont inchangés.

16. Exemple adapté d'après [Gallant, 1993].

3.1.3 Applications

Nous avons exploité cette méthode de classification pour trois usages différents : l'inférence de normes dans la simulation, la classification du comportement des agents, et l'automatisation de la configuration des normes [Lacroix et al., 2009a].

3.1.3.1 Inférence de normes

Les réseaux de Kohonen nous servent tout d'abord à inférer les normes de la simulation. En effet, comme nous l'avons vu, le paramétrage des agents est susceptible d'évoluer vers des valeurs qui ne correspondent pas à la norme initialement attribuée. Cependant, ces nouveaux paramétrages peuvent être considérés comme un ensemble de normes « de fait ». Déterminer leurs caractéristiques fournit un outil d'analyse de la simulation par l'utilisateur : celui-ci peut ainsi voir si sa configuration donne les résultats attendus et si les catégories qui apparaissent lui permettent d'expliquer les résultats obtenus.

L'objectif est donc d'utiliser la classification pour transformer des données en un ensemble restreint de normes associées à leur paramétrage, ces normes pouvant alors être utilisées pour analyser la simulation. Les données peuvent provenir d'enregistrements de la simulation ou de données réelles correspondant à la simulation. Ce dernier cas sera traité plus en détails dans la dernière partie de cette section (Sect. 3.1.3.3).

Ensemble d'exemples Les exemples fournis à l'algorithme sont des vecteurs constitués de valeurs des paramètres de la simulation. On se restreint ici uniquement aux paramètres externes $\mathcal{P}_e = \{p_i, \forall i \in [1, n]\}$. Les exemples peuvent ne contenir qu'une partie des paramètres de l'application : on a donc $\forall E$, $E = (v_k)$, $k \leq n$. Les composantes v_k du vecteur d'entrée sont libellées par l'utilisateur lors de la préparation des données et correspondent à un paramètre de référence de la simulation $p_k \in \mathcal{P}_e$.

Exemple 15 *Supposons que dans la simulation de piétons présentée plus haut, on enregistre à l'instant* t *toutes les vitesses instantanées des agents ainsi que leur espace personnel. Ce couple de donnée (v, s), disponible pour chaque agent, peut alors servir d'exemple pour l'entraînement du réseau : on a* $\mathcal{E} = \{E_i\}$ *avec* $\forall i$, $E_i = (v_i, s_i)$.

La première composante des vecteurs d'exemple est décrite par le label v_{max} *et la seconde par* s*, correspondant aux paramètres de référence définis dans la simulation.*

Paramétrage du réseau de Kohonen Nous avons choisi d'utiliser les valeurs suivantes :

- topologie : rectangulaire,
- nombre de neurones : afin de construire dynamiquement le réseau au démarrage de l'algorithme, sa taille est adaptée suivant celle des exemples fournis. On choisit un réseau de taille $(k+1) \cdot (k+1)$, avec k le nombre de composantes des vecteurs exemple. Le réseau comporte donc au total $(k+1)^2$ neurones,
- un taux d'apprentissage : on choisit celui décrit dans la Section 3.1.2.2, avec $\alpha_{\text{initial}} = 1.0$ et $\alpha_{\text{final}} = 0.05$,
- fonction de voisinage : une fonction Gaussienne, telle que présentée dans la Section 3.1.2.2,
- nombre d'itérations : si $k^2 < 100$, $T = 100$, sinon $T = k^2$.

Différents essais ont été menés afin d'évaluer le nombre de neurones nécessaires dans notre cas. Pour ce faire, nous avons entrainé un réseau sur des données enregistrées à partir de normes existantes en faisant varier le nombre de neurones, et nous avons analysé dans quelle mesure les normes initiales pouvaient être déduites des neurones entraînés. Nous avons constaté qu'un nombre trop faible de neurones ne permettait pas de représenter les données de manière précise, ce qui nous a conduit à choisir la valeur présentée ci-dessus. Les autres paramètres ont été déterminés à partir des valeurs recommandées dans la littérature.

Entraînement L'objectif est de fournir un maximum d'éléments à l'utilisateur. Après l'entraînement du réseau, deux types d'informations seront mises à sa disposition :

1. des éléments bruts, i.e. la représentation des données ainsi que les poids des neurones. En fonction de son expertise, l'utilisateur peut les exploiter pour analyser les résultats et améliorer son paramétrage.
2. un ensemble de normes créées automatiquement à partir des catégories inférées. Elles lui fourniront un point de comparaison auquel confronter son propre paramétrage.

Les différentes étapes de l'algorithme utilisé sont présentées dans l'Alg. 8. Lors de l'initialisation, un réseau de Kohonen de taille adaptée aux exemples est construit, tel que présenté plus haut. Pour chaque neurone du réseau, une norme est créée. Notons que ces normes ne contiennent que les paramètres décrits dans les exemples. La première étape consiste en l'entraînement du réseau par l'ensemble d'exemples. On dispose alors des éléments bruts correspondant à l'élément 1 cité ci-dessus.

La seconde étape est la création automatique des normes. Pour chaque neurone $u_{i,j}$, on dispose des poids $W_{i,j}$ obtenus lors du processus d'apprentissage, et d'une norme $N_{i,j}$. Par construction, chaque composante $w_{i,j,h}$ de $W_{i,j}$ correspond à un paramètre de $N_{i,j}$ dont le paramètre de référence est celui décrivant la composante utilisée de l'exemple. On assigne alors au paramètre p_h de la norme $N_{i,j}$ la valeur par défaut $w_{i,j,h}$. Il utilise la distribution de probabilité par défaut et la fonction distance par défaut. Cette étape définit toutes les valeurs par défaut des paramètres des normes créées à l'étape initiale.

La troisième étape permet de construire le domaine de définition des paramètres. Pour ce faire, nous réalisons une passe sur l'ensemble des exemples en mode classification (et non apprentissage comme précédemment). Chaque exemple déclenche l'activation d'un neurone de la couche de sortie. On sauvegarde la valeur maximale et minimale de chacun des paramètres pour chacun des neurones de sortie. Ces valeurs extrêmes fournissent les bornes des domaines de définition des paramètres [17]. On stocke par ailleurs également la proportion d'agents correspondant à chacune des catégories, i.e. déclenchant le même neurone. De cette façon, la proportion d'agents correspondant à chacune des normes est connue. A la fin de cette étape, les paramètres et les normes sont entièrement définis, à l'exception des éléments de référence.

Par ailleurs, afin de permettre un fonctionnement cohérent, les domaines des paramètres de référence correspondent aux bornes extrêmes des normes ; ils sont construits au cours du processus. La norme racine N_{init} inclut l'ensemble de ces paramètres et est affectée comme norme de référence à toutes les normes $N_{i,j}$. L'ensemble de ces éléments est regroupé dans une institution spécifique, afin de faciliter son utilisation et son stockage.

L'ensemble des normes ainsi créé fournit à l'utilisateur un outil d'analyse de son paramétrage. De plus, un second usage est possible. En effet, ces normes représentent l'ensemble de

17. Méthode applicable seulement pour des paramètres réels ou pouvant être représentés par des nombres réels.

Algorithme 8 Inférence de normes.

Entrées : une simulation de paramètres externes $\mathcal{P}_e = \{p_i, i \in [1,\ n]\}$; un ensemble d'exemples $\mathcal{E} = \{E_i\}$ avec $\forall i,\ E_i = (v_k)$ et $k \leq n$, les v_k correspondant à des paramètres de la simulation

1: **initialisation du réseau :** création d'un réseau de Kohonen $\mathcal{K}$ rectangulaire de taille $(k+1) \cdot (k+1)$, de neurones $u_{i,j}$ de poids $W_{i,j} = (w_{i,j,h})$, $h \leq k$
2: **initialisation des normes :** création de $(k+1)^2$ normes $N_{i,j}$, contenant les k paramètres p_k dont les libellés correspondent aux exemples, avec $Q_n = \emptyset$, $\tau_n = 0$ et $\delta_{\max_N} = 1$.
Toutes les $N_{i,j}$ ont pour norme de référence N_{init} initialisée de la même façon, et dont les p_{ref} sont construits au cours du processus
3: **entraînement :** entraîner $\mathcal{K}$ avec l'ensemble d'exemples $\mathcal{E}$
4: **pour tout** $i, j \leq k+1$ **faire**
5: **pour tout** $h \leq k$ **faire**
6: enregistrer la valeur $w_{i,j,h}$ du prototype du neurone comme valeur par défaut du paramètre correspondant p_h dans la norme $N_{i,j}$: $v_{d_{N_{i,j}}}(p_h) \leftarrow w_{i,j,h}$
7: si $w_{i,j,h}$ est supérieur au maximum ou inférieur au minimum de $\mathcal{D}_{p_{\text{ref}}(p_h)}$, mettre à jour la borne correspondante du domaine
8: **fin pour**
9: **fin pour**
10: créer un vecteur de proportions $\vec{prop} = (prop_{N_{1,1}}, \cdots, prop_{N_{k,k}}) = 0$
11: **pour tout** $E_i \in \mathcal{E}$ **faire**
12: classer l'exemple E_i avec le réseau $\mathcal{K}$: $u_{i,j}$ le neurone gagnant
13: incrémenter $prop_{N_{i,j}}$
14: **pour tout** $h \leq k$ **faire**
15: si $w_{i,j,h}$ est supérieur au maximum ou inférieur au minimum de $\mathcal{D}_{p_{i,j,h}}$, mettre à jour la borne correspondante du domaine
16: **fin pour**
17: **fin pour**
18: normaliser les proportions : $\vec{prop} = \vec{prop}/\text{Card}(\mathcal{E})$
19: stocker les $N_{i,j}$ et N_{init} dans une institution

Algorithme 9 Classification d'un agent.

Entrées : un agent modèle a_m de norme de référence N_{a_m} et de paramètres $\mathcal{C}_{a_m} = \{(l_p, v_{p_{a_m}}), p \in \mathcal{P}_{N_{a_m}}\}$

pour tout norme N **faire**
  pour tout paramètre $p \in \mathcal{P}_{N_{a_m}}$ **faire**
    si $p \in \mathcal{P}_N$ **alors**
      $q \leftarrow q + f_p(v_{p_{a_m}})$
    sinon
      $q \leftarrow q + f_p(v_{d_{p_{\text{ref},p}}})$
    fin si
  fin pour
  $q_{a_m}(N) \leftarrow \frac{q}{\text{Card}(N_{a_m})}$
fin pour

la population des agents présents dans la simulation au moment de l'enregistrement. À partir de ces éléments, il est donc possible de recréer une situation similaire à l'aide de générateurs automatiques.

3.1.3.2 Classification du comportement d'un agent

La seconde application est de catégoriser le comportement des agents. En effet, différentes questions peuvent se poser en cours de simulation ou a posteriori : le paramétrage est-il resté proche de la norme utilisée pour le créer ? A-t-il évolué vers un paramétrage correspondant à une autre norme ? Ou bien s'est-il totalement éloigné de ce que nous avions prévu à l'origine ?

Une première réponse est fournie par la comparaison systématique du paramétrage actuel de l'agent avec les normes initiales. Pour ce faire, on utilise la quantification de la norme pour chacune des normes. Toutes les normes ne définissent pas tous les paramètres, mais tous les paramètres sont disponibles dans l'agent modèle. Afin de ne pas défavoriser la norme initiale en comparant des paramètres qui n'avaient pas été spécifiés, on procède la manière suivante (Alg. 9) :

1. on se restreint aux paramètres définis dans la norme initiale N_{a_m} de l'agent,
2. pour chacune des normes $N \neq N_{a_m}$, si un paramètre n'est pas défini dans N mais qu'il l'est dans N_{a_m}, on réalise la comparaison en utilisant la valeur par défaut du paramètre de référence correspondant. Cela permet de ne pas léser les normes définissant beaucoup de paramètre vis à vis des autres, et réciproquement,
3. on quantifie le paramétrage par rapport aux valeurs ainsi obtenues.

Exemple 16 *Dans l'exemple des piétons, nous avions défini une norme* N_{normal}. *Supposons qu'il existe une autre norme* N_{presse}, *qui décrive le comportement de piétons pressés : vitesse préférée plus élevée et espace personnel plus réduit.* N_{presse} *est définie par* $l = N_{\text{presse}}$, $N_{\text{ref}} = N_{\text{pietons}}$, $\mathcal{P} = (v_{\text{presse}}, s_{\text{presse}})$, *les autres éléments étant ceux par défaut.*

On définit les paramètres v_{presse} *et* s_{presse} *respectivement par :*
- $l = v_{\text{presse}}$, $p_{\text{ref}} = v_{max}$, $\mathcal{D}_p = [5, 7]$ *km/h*, $v_{d_p} = 6$ *km/h, et* g_p *et* f_p *par défaut,*
- $l = s_{\text{presse}}$, $p_{\text{ref}} = s$, $\mathcal{D}_p = petit$, $v_{d_p} = petit$, *et* g_p *et* f_p *par défaut.*

A l'instant t, l'agent a_m de norme initiale N_{normal} a pour paramètres $v = 5.8$ km/h et $s =$ petit. En le classifiant à l'aide de l'algorithme présenté, on obtient les valeurs suivantes :

$$\begin{aligned} q_{a_m}(N_{\text{normal}}) &= \frac{1}{2} \cdot \left(\frac{\mid 5.8 - 5 \mid}{20 - 0} + 1 \right) = 0.52 \\ q_{a_m}(N_{\text{presse}}) &= \frac{1}{2} \cdot \left(\frac{\mid 5.8 - 6 \mid}{20 - 0} + 0 \right) = 0.005 \end{aligned}$$

A cet instant, a_m est donc plus proche de la norme N_{presse} que de la norme N_{normal}.

Cette méthode fournit un point de comparaison avec les normes initiales, et permet à l'utilisateur de déterminer de quelle norme un agent est le plus proche à un instant t. Utilisée itérativement sur tous les agents, elle donne une image de la répartition de l'ensemble des agents de la simulation par rapport aux normes définies initialement.

Une autre manière d'aborder cette problématique est de comparer l'état de l'agent avec les normes inférées de la simulation à cet instant. Cela sera particulièrement intéressant si tous les agents ont évolué vers un paramétrage très différent de leur initialisation, ou quand la quantification montre que le paramétrage est éloigné de toutes les normes initiales. On réalise alors la classification de l'agent par rapport aux normes inférées selon la méthode présentée dans la section précédente. Si on souhaite réaliser le processus en cours de simulation, il peut être déclenché manuellement.

La classification du comportement des agents fournit ainsi un second outil d'analyse de la simulation et de ses résultats, outil qui pourra être mis à profit par les utilisateurs pour configurer le modèle ou interpréter les résultats.

3.1.3.3 Calibration automatisée à partir de données réelles

Enfin, le dernier axe d'utilisation de l'inférence est le calibrage automatique de la configuration à partir de données réelles. Dans les simulations représentant des situations observables, concernant par exemple des piétons, des véhicules ou des animaux, certaines données sont mesurables et peuvent être enregistrées. La méthode présentée plus haut permet ainsi de préparer de manière automatisée une configuration de normes correspondant à ces observations.

Pour ce faire, il est nécessaire de mettre en correspondance les données enregistrées et les paramètres de la simulation. Cela implique une étape de préparation et de pré-traitement des données. On procède ensuite en utilisant l'algorithme 8, en utilisant cette fois les données réelles au lieu des données enregistrées.

Cette méthode fournit un moyen de paramétrage simple du modèle, et permet théoriquement de reproduire facilement des situations observées. Notons qu'en pratique, on se heurte ici à une des limites de l'approche : l'identification d'un paramétrage et du comportement d'un agent. Les données étant enregistrées à un instant t donné, et un comportement se caractérisant dans le temps, la simple restitution des paramètres enregistrés ne produira pas le comportement réel des agents. Il s'agit cependant d'un premier élément intéressant pour aller vers l'automatisation de l'approche. Nous verrons dans les perspectives quelles méthodes peuvent être envisagées pour améliorer ce point.

3.2 Variété et conformité

Dans cette seconde partie, nous nous intéressons aux possibilités offertes par le modèle pour créer de la variété dans le comportement des agents, et pour contrôler la conformité de leur paramétrage avec les spécifications.

3.2.1 Variété

Les normes offrent un outil flexible pour créer des comportements variés. Deux leviers peuvent être utilisés pour influer sur ce critère : la construction des normes et la violation.

3.2.1.1 Variété par la construction des normes

La capacité à créer autant de paramètres et de normes que souhaité ouvre de grandes possibilités de variété par simple construction des normes.

Tout d'abord, les domaines de définition des paramètres peuvent être construits de manière à présenter des caractéristiques spécifiques. Par exemple, si le comportement que l'on souhaite reproduire est destiné à être attribué à un grand nombre d'agents, le domaine pourra être défini par une large plage de valeurs. Au contraire, en utilisant un domaine très réduit, comme un singleton, le paramétrage résultant est connu dès la spécification : tous les agents instanciant ce paramètre possèderont la même valeur, en supposant les agents non violateurs.

Par ailleurs, les normes définissent un nombre quelconque de paramètres, ce qui leur donne une grande flexibilité. Par exemple, si une norme ne spécifie qu'un seul paramètre dont le domaine de définition est réduit à un singleton, elle produira toujours le même paramétrage : tous les paramètres prendront la valeur par défaut de celui de référence, et le paramètre spécifié prendra celle définie dans la norme. Au contraire, si la norme définit tous les paramètres de la simulation, chacun d'entre eux possédant un large domaine de définition, alors une grande variété de paramétrages d'agent pourra être produite.

Par ailleurs, les normes présentent l'avantage de préserver un contrôle total sur la configuration de la simulation. En effet, si une norme est supprimée, le paramétrage correspondant ne sera plus jamais créé. De plus, contraindre le domaine de définition des paramètres est peu limitant pour les utilisateurs, qui peuvent obtenir le comportement recherché simplement en définissant une nouvelle norme.

Exemple 17 *Supposons que nous souhaitons étudier l'influence de la présence de joggeurs sur le flux de piétons dans la simulation. En plus de la norme* N_{normal}*, on introduit la norme* N_{joggeur} *définissant le paramètre* $v_{max,joggeur}$ *tel que* $v_{d_{max,joggeur}} = 11$ *km/h et* $\mathcal{D}_{v_{max,joggeur}} = \{11\}$ *km/h.*

En générant une population composée de 99 % d'agents instanciant N_{normal} *et de 1 % d'agents instanciant* N_{joggeur}*, les joggeurs sont introduits naturellement dans la simulation, sans devoir être configurés manuellement. De plus, la désactivation de cette norme suffit à empêcher l'apparition de nouveaux joggeurs.*

La richesse de définition des normes s'appuie par ailleurs sur l'introduction de variété au sein de chacune des normes. Lors de l'instanciation du paramétrage, la distribution de probabilité des paramètres permet en effet de créer de la variété parmi les comportements appartenant à

une même norme : ceux-ci prennent des valeurs différentes au sein du domaine défini. Le moteur de génération introduit ainsi naturellement de la variété au sein de chacune des normes au cours du processus d'instanciation.

Notons que cette distribution peut être adaptée suivant les besoins de l'utilisateur. Par défaut, la fonction utilisée est une distribution uniforme sur le domaine de définition, mais il est possible d'utiliser tout type de fonction, comme par exemple une fonction gaussienne, ce qui augmente la flexibilité de l'approche.

Exemple 18 *Deux agents instanciés à partir de la norme* N_{normal} *se verront attribuer des jeux de paramètres différents, sans intervention de l'utilisateur. Ces jeux de paramètres resteront dans les plages de valeur spécifiées dans la norme. En créant deux agents* a_1 *et* a_2 *instanciant* N_{normal}*, on pourra par exemple obtenir pour* a_1*,* $v_{max}(a_1) = 5.2$ *km/h et* $s(a_1) = normal$*, et pour* a_2*,* $v_{max}(a_2) = 4.7$ *km/h et* $s(a_2) = normal$*.*

La définition d'une seule norme suffit à obtenir un paramétrage différent pour chacun des agent de la simulation pris en charge par le modèle. L'introduction de normes différentes permet de spécifier des familles ou des types d'agents dont les paramétrages auront des caractéristiques communes. Elles offrent la possibilité d'enrichir les spécifications de la simulation, sans complexifier le processus outre mesure.

3.2.1.2 Variété par la violation des normes

Le second levier pour introduire de la variété dans le comportement des agents est l'utilisation de violations de la norme.

Les violations permettent d'introduire des comportements non spécifiés dans la simulation. Différentes utilisations peuvent en être faite. Tout d'abord, leur introduction permet de tester la robustesse de l'organisation des agents à des perturbations extérieures. L'ajout de tels agents peut aussi déstabiliser le système, dans le but de le faire évoluer vers un état plus stable. Cet usage est comparable à l'utilisation du recuit simulé pour sortir des optimums locaux en optimisation. Enfin, pour citer un exemple pratique, les simulateurs de conduite permettent de tester des systèmes d'aide au conducteur. Introduire des comportements atypiques, qui déstabilisent le conducteur et le conduisent à réagir de manière imprévue, élargit le domaine des essais réalisés et donc la robustesse du système testé.

Dans le modèle, la spécification des violations se fait par l'intermédiaire de trois paramètres : le taux de violation de la norme τ_N, l'écart maximal à la norme $\delta_{\max_N}$, et le critère global de déterminisme de l'institution σ. Si τ_N est fixé à une valeur non nulle, il permet de créer des paramétrages qui, bien qu'ayant pour référence la norme choisie, pourront s'en distinguer dans une proportion déterminée par $\delta_{\max_N}$. Lors de l'instanciation du comportement d'un agent (Alg. 6, p. 67), τ_N est testé pour sélectionner le domaine de définition depuis lequel un paramètre p sera instancié. Il s'agira soit du domaine $\mathcal{D}_p(N)$ défini dans la norme N (pas de violation), soit du domaine $\mathcal{D}_p(N_{ref})$ défini dans la norme de référence $N_{ref}(N)$. Dans ce dernier cas, il y violation si la valeur utilisée provient de $\mathcal{D}_p(N_{ref}) - \mathcal{D}_p(N)$.

Le taux de violation donne donc la possibilité d'obtenir des agents violateurs. L'écart maximal à la norme permet d'éviter qu'un paramétrage ne s'écarte trop de la norme. Enfin, le paramètre σ, défini au niveau global dans l'institution, permet d'activer ou de désactiver très simplement l'utilisation de la violation (si $\sigma = 0$, aucune violation n'est possible, quelque soit la valeur des τ_N).

Exemple 19 *Une autre méthode pour obtenir les joggeurs introduits dans l'exemple 17 est d'utiliser la violation de la norme. Pour cela, on définit la norme* $N_{\text{normal}''}$ *identique à* N_{normal} *mais avec un taux de violation* $\tau_{N_{\text{normal}''}} = 0.01$*. On fixe de plus* $\sigma = 1$ *pour autoriser les violations. En instanciant un paramétrage depuis* $N_{\text{normal}''}$*, des agents possèdant une vitesse rapide, assimilables à des joggeurs, seront créés.*

Notons que dans ce cas, nous obtiendrons non seulement des joggeurs, mais aussi des personnes se déplaçant lentement, d'autres acceptant de faibles distances interpersonnelles et toute une palette de comportements. Si l'objectif est de tester spécifiquement l'influence de l'introduction d'agents plus rapides, l'utilisation d'une norme N_{joggeur} *sera plus adaptée. S'il s'agit d'observer l'influence de perturbations variées et pas nécessairement prévues, les violations apportent une meilleure solution.*

3.2.2 Conformité

En plus de l'introduction aisée de variété dans les comportements, le modèle proposé permet de contrôler la conformité du paramétrage des agents. Dans cette partie, nous nous intéressons tout d'abord à la distinction entre cohérence et conformité, avant de présenter les mécanismes de contrôle mis en place avec le modèle.

3.2.2.1 Cohérence et conformité

Tout au long de ce travail, nous avons parfois confondu la notion de cohérence et celle de conformité. Comme nous l'avons souligné dans la Section 2.3.2.1 (p. 63), les comportements instanciés par le modèle de différenciation ne seront cohérents que si la spécification fournie par la norme, le « profil comportemental », l'est aussi. Le modèle proposé n'est pas en mesure de garantir la cohérence des comportements pour tout paramétrage. En effet, rien ne garantit que les comportements définis grâce aux normes le soient. Au contraire, un utilisateur pourrait même vouloir expérimenter l'influence de comportements incohérents, et les introduire en les spécifiant à l'aide de normes.

Cependant, lors de l'exécution, le modèle permet de contrôler la conformité du paramétrage des agents avec la spécification fournie par la norme. Il s'agit donc d'évaluer dans quelle mesure le paramétrage effectif des agents correspond à celui spécifié dans leur norme de référence, et de réagir en fonction des critères fixés par l'utilisateur.

3.2.2.2 Contrôle de la conformité

Deux éléments permettent de contrôler la conformité : la norme racine d'une part, et l'écart maximal à la norme d'autre part.

La norme racine La norme racine est une norme régimentée : elle ne peut être violée (cf. Sect. 2.2.2.3 p. 52). Cela signifie que si un agent de la simulation prend une valeur en dehors du domaine de définition du paramètre racine correspondant, cette valeur est interdite et forcée à la borne correspondante du domaine.

L'algorithme 10 présente ce fonctionnement. Pour chacun des agents modèle, le paramétrage mis à jour par la simulation est vérifié. Si un paramètre sort du domaine de définition du

Algorithme 10 Régimentation des paramètres racine.

Entrées : $\mathcal{A}_m$ l'ensemble des agents modèles ; $\mathcal{A}$ l'ensemble des agents de la simulation

pour tout $a_m \in \mathcal{A}_m$ **faire**
    créer une copie temporaire de a_m pour le calcul : $a_{temp} \leftarrow a_m$
    recopier les valeurs des paramètres de l'agent de la simulation a correspondant à a_m dans a_{temp} : $\forall p$, $v_p(a_{temp}) \leftarrow v_p(a)$
    pour tout paramètre $p \in N_{\text{init}}$ **faire**
        si $v_p(a_{temp}) \notin \mathcal{D}_p(N_{\text{init}})$ **alors**
            forcer la valeur de a à son ancienne valeur, provenant de a_m : $v_p(a) \leftarrow v_p(a_m)$
        sinon
            mettre à jour la valeur dans a_m : $v_p(a_m) \leftarrow v_p(a)$
        fin si
    fin pour
fin pour

paramètre racine correspondant, alors sa valeur est fixée en forçant l'utilisation de celle du pas de temps précédent. Sinon, aucune réaction n'a lieu et le nouveau paramétrage est sauvegardé dans l'agent modèle.

Exemple 20 *La norme racine N_{pietons} contient le paramètre v_{max} dont le domaine de définition $\mathcal{D}_{v_{max}}$ est l'intervalle [0,20] km/h. Tout agent géré par le modèle ne pourra donc avoir de vitesse supérieure à 20 km/h. Notons que cela interdit d'avoir des agents effectuant un sprint rapide, par exemple pour un comportement de fuite*[18].

La régimentation de la norme racine fournit un premier outil de contrôle de la conformité aux spécifications : quelle que soit l'évolution de la simulation, des bornes limites sont spécifiées pour les agents instanciés par le modèle.

L'écart maximal à la norme L'écart maximal à la norme permet par ailleurs de contraindre les comportements lors de l'exécution. Ce mécanisme correspond à la réaction aux violations présentée dans la Section 2.2.2 (p. 58). En pratique, l'algorithme proposé permet de préserver la conformité du paramétrage en suivant les spécifications de l'utilisateur.

La conformité est spécifiée par le paramètre $\delta_{\max_N}$. Si $\delta_{\max_N} = 0$, aucune violation ne sera autorisée. Si $\delta_{\max_N} = 1$, les paramètres peuvent évoluer librement et toutes les violations sont possibles. Entre les deux, ce paramètre défini le degré de conformité aux spécifications, en autorisant l'apparition de violations plus ou moins grandes.

Notons que ce n'est pas parce que les violations sont autorisées qu'elles se produiront. Tout dépend des paramètres spécifiés par les normes : si la norme spécifie des paramètres statiques du modèle, il n'y aura jamais de violation déclenchée par le modèle de décision. Les seules violations qui pourraient se produire seraient liées à l'intervention d'un module externe ou de l'utilisateur, modifiant directement la valeur d'un paramètre. Par ailleurs, si les paramètres spécifiés dans la norme évoluent, mais restent en permanence dans les limites autorisées par la norme, il n'y aura jamais de violation. Enfin si les normes autorisent des plages de valeurs très larges pour les paramètres qu'elles spécifient, les violations seront rares, voire inexistantes.

18. Pour référence, un champion de 100 m court à une vitesse de l'ordre de 36 km/h, un marathonien de niveau mondial à une vitesse moyenne de 20 km/h.

Algorithme 11 Contrôle de la conformité à l'exécution.

Entrées : $\mathcal{A}_m$ l'ensemble des agents modèles ; $\mathcal{A}$ l'ensemble des agents de la simulation

pour tout $a_m \in \mathcal{A}_m$ **faire**
  créer une copie temporaire de a_m pour le calcul : $a_{temp} \leftarrow a_m$
  recopier toutes les valeurs des paramètres de l'agent de la simulation a correspondant à a_m dans a_{temp} : $\forall p,\ v_p(a_{temp}) \leftarrow v_p(a)$
  régimentation (cf. Alg. 10)
  pour tout paramètre $p \in N_{\text{init}}$ **faire**
    si $v_p(a_{temp}) \notin \mathcal{D}_p(N_{\text{init}})$ **alors**
      forcer la valeur de a à son ancienne valeur, provenant de a_m : $v_p(a) \leftarrow v_p(a_m)$
    sinon
      mettre à jour la valeur dans a_m : $v_p(a_m) \leftarrow v_p(a)$
    fin si
  fin pour
  violations (cf. Alg. 3)
  quantifier les violations de a_{temp} (correspondant au paramétrage de a) : $q_{a_{temp}} \leftarrow$ **quantify_violation**(a_{temp})
  si $q_{a_{temp}} \leq \delta_{\max_{N_{a_m}}}$ **alors**
    l'écart maximal à la norme n'est pas dépassé, la simulation se poursuit : $\forall p \in \mathcal{P}_a,\ v_p(a_m) \leftarrow v_p(a)$
  sinon
    la violation est interdite, on force les paramètres de a à leur ancienne valeur : $\forall p \in \mathcal{P}_{a_m},\ v_p(a) \leftarrow v_p(a_m)$
  fin si
fin pour

Mécanisme de contrôle Finalement, le mécanisme de contrôle à l'exécution présenté dans l'algorithme 11 est constitué d'une fusion des algorithmes 3 (p. 59) et 10 (p. 85) : il vérifie tout d'abord qu'aucun paramètre ne se situe en dehors du domaine de définition du paramètre racine correspondant, puis qu'aucun paramétrage n'excède l'écart maximal à la norme autorisée.

La limite de ce fonctionnement, déjà évoquée dans la Section 2.2.3.3, est qu'imposer aux paramètres du modèle de rester dans les bornes spécifiées peut potentiellement conduire à des comportements inattendus. Le modèle de simulation n'intègre en effet pas nécessairement les mécanismes permettant de gérer des interventions « manuelles », telles que celles que nous réalisons. Le mécanisme de contrôle doit être utilisé prudemment, en possédant souvent des connaissances approfondies sur le modèle de décision des agents.

3.3 Utilisation de l'outil

Cette section présente l'utilisation du modèle de différenciation comportementale comme un outil permettant d'introduire de la diversité dans une simulation tout en maîtrisant la conformité des éléments créés. Nous présentons tout d'abord les différents modes d'utilisation envisageables, puis leur mise en application.

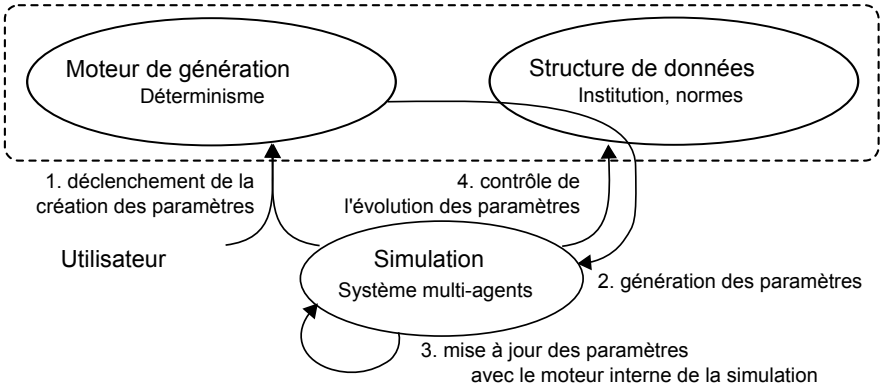


FIGURE 3.3 – Utilisation du modèle proposé en tant que module externe à la simulation.

3.3.1 Différents modes d'utilisation

Le modèle peut être utilisé à différents niveaux de la simulation. Tout d'abord, il peut être utilisé comme un outil externe fournissant des paramètres d'entrée à la simulation, tel que présenté dans la Figure 3.3. La simulation se déroule en utilisant son modèle de décision interne, et ne sollicite le modèle de différenciation comportementale que dans des situations particulières, comme la création des paramètres.

Une autre façon d'utiliser le modèle est de l'appliquer non seulement sur les paramètres externes de la simulation, mais aussi sur ses paramètres internes. Dans ce cas, le modèle doit être intégré en tant que module interne de la simulation. Il est cependant nécessaire de procéder prudemment, les paramètres internes étant souvent dynamiques : le mécanisme de contrôle peut alors avoir des effets non désirés, comme nous le présentions dans la Section 2.2.3.3 (p. 58). Dans ce cas, l'outil peut également être utilisé en tant qu'outil de reporting, en mettant à profit les capacités de contrôle présentées dans la Section 3.1. Étant en mesure d'observer les paramètres internes de la simulation, il peut alors fournir des informations plus complètes sur son fonctionnement.

L'utilisation de l'outil présente deux avantages principaux : son caractère non-intrusif et la possibilité de réaliser la conception en dehors de l'agent. En effet, à partir du moment où une simulation expose ses paramètres de configuration, l'intégration de l'outil ne nécessite aucune modification de celle-ci. La définition des normes peut se baser sur ces paramètres et interagir avec la simulation par leur biais. L'utilisation de l'outil est alors totalement non-intrusive. Dans ce cas, le modèle peut être paramétré pour générer des valeurs conformes à celles attendues en entrée de la simulation et introduire facilement les apports du modèle.

Le second avantage est de permettre une conception en dehors des agents. Par l'intermédiaire de l'outil, leur paramétrage peut être spécifié à l'extérieur de la simulation, ce qui offre une plus grande flexibilité lors de la conception et de l'utilisation. Cela fournit par exemple la possibilité à des utilisateurs non programmeurs de l'exploiter pour leurs expérimentations.

3.3.2 Utilisation de l'outil

L'utilisation de l'outil repose sur la définition du paramétrage des normes d'une part, et le couplage à la simulation d'autre part.

Paramétrage de l'outil Pour permettre une utilisation intuitive, nous limitons le nombre d'éléments de configuration nécessaires pour une utilisation immédiate. Pour utiliser l'outil, un utilisateur n'a besoin de spécifier que les éléments suivants :

- des paramètres, leurs valeurs par défaut et leurs domaines de définition. Par exemple : la vitesse maximale v_{max} d'un piéton vaut par défaut 5 km/h et peut prendre sa valeur entre 4 et 6 km/h.
- une norme, à laquelle est associée un ou plusieurs paramètres. Par exemple : la norme N_{normal} à laquelle est associée le paramètre v_{max}.

Avec ces simples éléments, l'utilisateur peut utiliser le modèle pour générer des paramétrages pour les agents. Notons que la définition d'une seul paramètre et d'une seule norme suffisent.

Paramétrage par défaut Les éléments qui n'ont pas été demandés à l'utilisateur sont complétés automatiquement, en utilisant un paramétrage par défaut.

Pour les paramètres, les éléments suivants sont ajoutés :

- p_{ref} : pour chaque paramètre p défini par l'utilisateur, un paramètre « racine » p_{rac} est automatiquement créé, avec des valeurs identiques à celles de p. Ce paramètre est associé à p en tant que paramètre de référence,
- g_p : prend sa valeur par défaut (Sect. 2.2.2 p. 49),
- f_p : prend sa valeur par défaut (Sect. 2.2.2 p. 49).

Une norme racine N_{init} contenant tous les paramètres « racine » p_{rac} est automatiquement créée.

Pour les normes, des valeurs par défaut sont attribuées aux paramètres suivants :

- $N_{\text{ref}} = N_{\text{init}}$: la norme de référence de toutes les normes est N_{init},
- $\mathcal{Q}_N = \emptyset$: les normes ne possèdent pas de propriétés particulières,
- $\tau_N = 0$: par défaut, aucun paramétrage violateur ne sera généré. En effet, l'utilisation des violations peut être contre-intuitive, et nécessite une compréhension avancée du modèle,
- $\delta_{\max_N} = 1$: par défaut, les violations ne sont pas contraintes par le modèle. Comme nous l'avons vu dans la Section 2.2.3.3, ces contraintes peuvent conduire à des comportements inattendus de la simulation. S'il souhaite les mettre à profit, l'utilisateur doit faire ce choix en connaissance de cause.

Enfin, une institution est automatiquement créée. Elle contient l'ensemble des normes créées manuellement, ainsi que la norme racine. La valeur du facteur global de déterminisme σ est fixée à 0, afin d'interdire par défaut toute violation.

Utilisation avancée : définir les paramètres racine Pour exploiter au mieux le modèle, les paramètres « racine » du modèle peuvent être définis par l'utilisateur. Cette étape nécessite une certaine expertise, mais pourra être réalisée par tout utilisateur possédant une bonne connaissance du fonctionnement de la simulation. Typiquement, elle ne sera réalisée qu'une seule fois : une fois les éléments racine du modèle construits, tout utilisateur est en mesure de concevoir son propre jeu de normes à partir de ceux-ci. Il est également possible de ne permettre à l'utilisateur final que la modification de certains des paramètres disponibles, afin de contrôler l'utilisation du modèle.

Les paramètres « racine » sont définis en leur associant les domaines de définition correspondant aux valeurs extrêmes qui pourront être prises. Éventuellement, un jeu de normes utilisable directement par les utilisateurs finaux peut également être construit. La norme racine et l'institution sont construites automatiquement, mais le concepteur a la possibilité de modifier leur contenu.

Couplage avec la simulation Adossé au moteur de génération, le paramétrage défini ci-dessus permet de créer la variété souhaitée. Il reste à faire parvenir ces valeurs au moteur de la simulation. Deux méthodes peuvent être utilisée, suivant son architecture.

Tout d'abord, si la simulation expose une fonction permettant de créer des agents par une requête externe, l'outil peut l'utiliser. Par exemple, dans certaines simulations un message réseau réalise cette fonction. Certains paramètres d'entrée devront peut-être lui être ajoutés, mais les impacts sont mineurs : il s'agira alors le plus souvent d'une simple surcharge de constructeur. Nous utilisons cette méthode pour l'application du modèle à la simulation de trafic qui est présentée dans le Chapitre 4.

Si une telle fonction n'est pas disponible, il est alors nécessaire de construire le point d'entrée. Soit il est possible de modifier le code source de l'application pour l'introduire, si les impacts ne sont pas trop importants, soit le modèle peut être ajouté en tant que module interne de la simulation pour pouvoir créer lui-même les agents. Notons que dans ce cas les impacts sont plus importants et que le caractère non-intrusif de l'approche est en partie perdu, suivant l'importance des modifications nécessaires. L'introduction du modèle reste cependant intéressante de par son aspect modulaire et la conception en dehors de l'agent qu'elle introduit dans la simulation.

3.4 Généricité

Dans cette partie, nous présentons deux exemples d'application illustrant la généricité de l'approche : l'introduction de variété dans la simulation de foules, puis la création de créature d'espèces différentes dans une simulation de type jeu vidéo.

3.4.1 Variété dans la simulation de foules

Dans le Chapitre 1, un des exemples que nous avons présenté concernait l'introduction de variété dans les simulations de foules (Sect. 1.3.3, p. 32). Les méthodes utilisées était basées sur la variation de couleur des textures des modèles graphiques, sur l'ajout d'accessoires et sur la modification de la corpulence des agents. Ces trois méthodes reposent sur la sélection aléatoire des composantes qui seront instanciées pour chacun des agents.

Par ailleurs, les différents paramètres utilisés nécessitaient l'utilisation de contraintes sur les domaines de définition. Pour la variation des couleurs, la palette graphique est restreinte dans l'espace TSV : la teinte de 20 à 250, la saturation de 30 à 80 % et la luminosité de 40 à 100 %. Ces valeurs sont codées en dur dans l'application. Pour l'ajout d'accessoires sur les agents, le nombre et le type d'accessoire pour chaque agent est également restreint. Chaque modèle graphique se voit associer pour chaque type d'accessoire une liste stockée directement dans le modèle. Enfin, pour la corpulence, un paramètre décrit le pourcentage de la taille initiale à appliquer. Dans tous les cas, la variété introduite repose donc sur des paramètres stockés en dur dans différentes parties de l'application.

Le modèle de différenciation comportementale que nous proposons permet de gérer ces différents critères. Pour ce faire, il suffit de définir les paramètres correspondant aux éléments présentés et de leur associer des normes. Une manière de procéder est de créer une norme pour chacun des types d'agents que l'on souhaite produire. Par exemple ici, une norme N_{homme} et une norme N_{femme}, qu'il faut distinguer car ils n'utilisent pas les mêmes ensembles d'accessoires.

Nous créons les paramètres suivants :

1. trois paramètres pour les éléments de la palette graphique :
   - un paramètre teinte t, avec $\mathcal{D}_t = [20,\ 250],\ v_{d_t} = 135$,
   - un paramètre saturation sat, avec $\mathcal{D}_{sat} = [30,\ 80],\ v_{d_{sat}} = 55$,
   - un paramètre luminosité lum, avec $\mathcal{D}_{lum} = [40,\ 100],\ v_{d_{lum}} = 70$,
2. un paramètre pour chaque catégorie d'accessoire (dans le cadre de cet exemple, nous ne définissons que deux catégories sur les sept existantes dans l'application réelle) :
   - un paramètre s_{femme} pour les sacs à main pour femme avec :

$$\mathcal{D}_{s_{\text{femme}}} = \{aucun, sac_fantaisie,\ pochette,\ sac_cuir\},\ v_{d_{s_{\text{femme}}}} = aucun$$

   - un paramètre s_{homme} pour les sacs pour homme avec :

$$\mathcal{D}_{s_{\text{homme}}} = \{aucun,\ besace_cuir,\ mallette_noire\},\ v_{d_{s_{\text{homme}}}} = aucun$$

   - un paramètre c_{femme} pour les chapeaux pour femme avec :

$$\mathcal{D}_{c_{\text{femme}}} = \{aucun,\ capeline,\ bandeau\},\ v_{d_{c_{\text{femme}}}} = aucun$$

   - un paramètre c_{homme} pour les chapeaux pour homme avec :

$$\mathcal{D}_{c_{\text{homme}}} = \{aucun,\ casquette,\ borsalino\},\ v_{d_{c_{\text{homme}}}} = aucun$$

3. un paramètre pour la corpulence c, avec $\mathcal{D}_c = [30,\ 130], v_{d_c} = 50$.

De la même façon, des paramètres concernant les vêtements, les coupes de cheveux, ou tout autre élément variable sur les agents peuvent être créés.

Notons qu'une des problématiques rencontrée dans ce travail était de ne pas créer d'agents surchargés d'accessoires, qui manqueraient tout autant de réalisme que des agents n'en possédant aucun. Nous prenons en compte ce point en modifiant la distribution de probabilité pour le tirage des paramètres les concernant. Supposons que l'objectif soit que les agents portent en moyenne deux accessoires. Au lieu de la distribution uniforme par défaut, on utilise une fonction qui a 5 chances sur 7 de tirer la valeur par défaut *aucun*, et un tirage aléatoire sur les autres éléments dans les autres cas. Pour la corpulence, on utilise une distribution gaussienne autour de la valeur par défaut.

Les normes sont alors créées de la manière suivante :
- pour N_{femme}, $\mathcal{P}_{N_{\text{femme}}} = \{t,\ sat,\ lum,\ s_{\text{femme}},\ c_{\text{femme}}\}$
- pour N_{homme}, $\mathcal{P}_{N_{\text{homme}}} = \{t,\ sat,\ lum,\ s_{\text{homme}},\ c_{\text{homme}}\}$

Conformément au fonctionnement présenté dans la Section 3.3.2, les éléments du modèle non spécifiés sont créés automatiquement : paramètres racines, norme racine, institution, éléments complémentaires des paramètres et des normes présentés. Le couplage avec la simulation est réalisé en fonction des possibilités offertes par l'application.

En permettant une conception en dehors de l'agent, la flexibilité apportée par le modèle rend la configuration accessible à des utilisateurs ne possédant pas une expertise graphique. Ainsi, si un utilisateur ne souhaite pas voir apparaître certains accessoires, il peut simplement les supprimer de la norme. S'il ne veut pas voir apparaître d'accessoire du tout, il lui suffit de supprimer le paramètre de la norme. Tous ces éléments sont disponibles dans l'outil : il n'est pas nécessaire de modifier les modèles graphiques des agents, ou le modèle de simulation dans le cas des couleurs ou de la corpulence. Des utilisateurs non experts peuvent faire varier les paramètres sans avoir besoin d'une connaissance approfondie du fonctionnement interne de la simulation. De plus, la gestion des paramètres est réalisée de manière centralisée, ce qui facilite la maintenance, et ainsi le déterminisme du processus est mieux maîtrisé.

paramètre	N_{init}	N_{gentil}	N_{mechant}
aggressivité	[0, 100]	[10, 50]	[50, 90]
nombre d'enfants	[0, 5]	[1, 4]	[0, 2]
vitesse	[0, 25]	[5, 10]	[15, 20]
taille	[10, 20]	[10, 16]	[14, 20]

TABLE 3.2 – Paramètres définissant les normes qui caractérisent les espèces dans la simulation.

3.4.2 Un créateur de créature

Nous avons appliqué l'outil à un second exemple : la génération de diversité parmi des espèces de « créatures ». Les caractéristiques définissant un individu peuvent être variées, mais doivent rester dans certaines limites pour qu'il appartienne toujours à son espèce initiale.

Afin d'illustrer ce fonctionnement, on peut réaliser une analogie entre les couples norme / agent modèle et génotype / phénotype. En génétique, un organisme est caractérisé par son génotype et son phénotype. Le phénotype représente ses propriétés observables : morphologie, physiologie, comportement, etc. Il est le résultat de l'expression de ses gènes, ajouté à l'influence de l'environnement lors de son développement. Le génotype est la constitution génétique de l'organisme, sa « recette ».

Dans la perspective de notre modèle, l'ensemble des paramètres des normes peut être considéré comme une « soupe primordiale » de gènes, de laquelle sont issus tous les organismes présents dans l'environnement. Ces caractéristiques sont associées à leur expression potentielle en tant que phénotype, sous forme de domaines de définition. Les normes contiennent alors des ensembles de paramètres communs à une espèce particulière : le « génome » de cette espèce. L'instanciation d'une norme, expression du génome en tant que phénotype, est le paramétrage d'un agent. En poussant l'analogie, les violations rendues possibles par le modèle peuvent enfin être vues comme des mutations. Le génotype est ainsi contenu dans les normes, le phénotype est l'instanciation de ces normes par le modèle.

Le modèle est utilisé pour générer des créatures de plusieurs espèces différentes, qui peuvent se déplacer dans la simulation et interagir en fonction de leurs caractéristiques et de leur espèce.

Deux interactions sont possibles dans la simulation. Lors de la rencontre entre deux agents, ceux-ci peuvent se combattre s'ils sont d'espèces différentes, se reproduire s'ils sont de la même espèce, ou s'ignorer. Quatre paramètres sont utilisés :

- l'agressivité quantifie la chance qu'a un agent d'interagir quand il rencontre un autre agent. Plus l'agressivité est élevée, plus il a de chance de combattre. Le nombre $(1 - aggressivite)$ est utilisé pour déterminer les chances de reproduction. S'il n'y a pas interaction, les agents s'ignorent et poursuivent leur chemin,
- le nombre d'enfants détermine la taille des portées des agents s'il y a reproduction,
- la vitesse caractérise sa vitesse de déplacement,
- enfin la taille détermine sa force physique : plus un agent est grand, plus il a de chance de triompher lors d'un combat.

À partir de ces paramètres, différentes espèces peuvent être définies, espèces qui sont décrites sous la forme de normes. Par exemple, on peut définir une espèce d'agents « gentils » et une espèce d'agents « méchants ». Les gentils sont peu agressifs, se déplacent lentement, sont petits mais ont beaucoup d'enfants. Au contraire les méchants sont agressifs, se déplacent vite, sont grands mais ont peu d'enfants. Les normes correspondant à ces deux espèces sont présentées dans le Tableau 3.2.

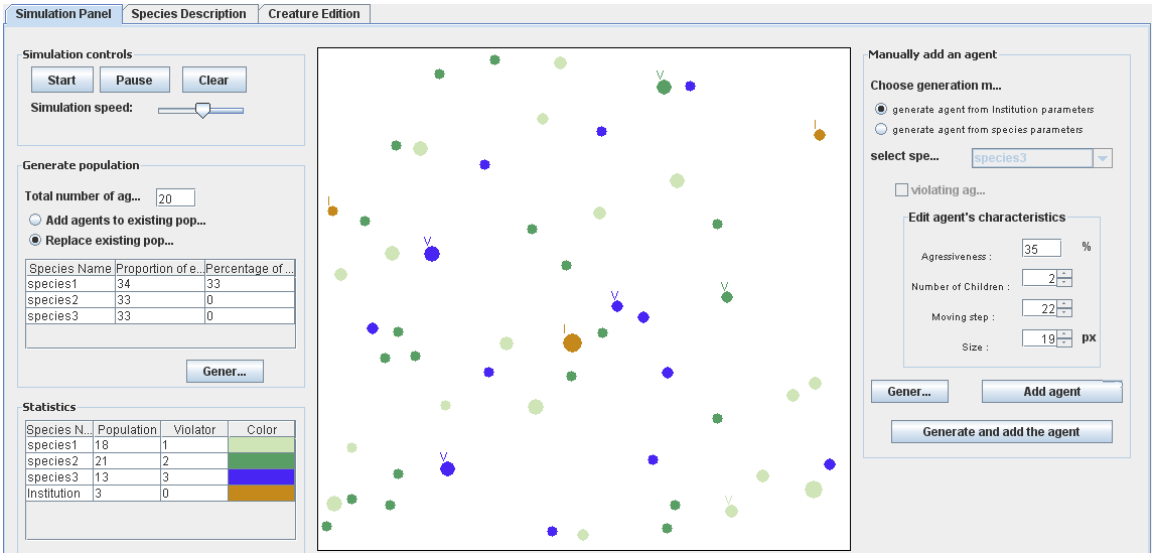


FIGURE 3.4 – Applet Java de démonstration de la création d'agents d'espèces différentes, basée sur l'utilisation du modèle de différenciation comportementale proposé.

Une implémentation de cet exemple a été réalisée dans le cadre d'un projet de fin d'études [Macret, 2009]. Cette simulation permet par exemple de visualiser l'influence des différents paramètres sur le déroulement de la simulation : quels sont les critères qui conduisent une espèce à survivre, plutôt qu'une autre ? Quelles sont les conséquences de l'introduction d'agents violateurs, qui dans ce contexte peuvent être considérés comme « mutants » ? Réalisée sous la forme d'une applet Java, elle a permis la validation de l'utilisation du modèle, et fournit un outil de démonstration de ses possibilités (Fig. 3.4).

3.5 Conclusion

Dans ce troisième chapitre, nous nous sommes intéressés au dernier axe de notre modèle, concernant l'observation et l'analyse des normes et de la simulation. Nous avons tout d'abord introduit nos motivations pour ces aspects, avant de présenter différentes techniques de classification non supervisée. Nous avons choisi d'utiliser les réseaux de Kohonen, ou cartes auto-adaptatives. Basés sur des réseaux de neurones, ils construisent une représentation topologique des données d'entrée, ce qui fournit une représentation intuitive pour les utilisateurs. Ces algorithmes nous fournissent la base pour trois usages. Tout d'abord, ils nous permettent d'inférer les normes de la simulation, et ainsi fournir à l'utilisateur des éléments représentant les normes utilisées par les agents au cours de la simulation. Notre deuxième usage est la classification du comportement des agents. Celui-ci peut être comparé à l'ensemble initial des normes, mais également aux normes inférées, afin d'étudier l'évolution de son paramétrage. Enfin, le dernier usage est la création automatisée de configurations de normes à partir de données enregistrées de la simulation.

Dans un second temps, nous avons montré comment le modèle proposé permet de créer de la variété dans le comportement des agents, tout en permettant de contrôler leur conformité aux spécifications. La variété peut être atteinte par deux moyens. Le premier est de se baser sur la configuration des normes. Suivant les besoins, les paramètres utilisées peuvent définir

des ensembles larges ou précis, et les normes utiliser n'importe quel nombre de paramètres. Le mécanisme de génération du paramétrage des agents permet ensuite de générer de la variété au sein des normes. Le second moyen se base sur la violations des normes. En les autorisant, l'utilisateur peut obtenir des comportements non spécifiés, tout en définissant à la fois leur proportion et l'écart à la norme qu'ils présenteront. Ces deux moyens permettent d'introduire aisément de la variété dans la simulation. Le contrôle de la conformité est basé d'une part sur la norme racine, et d'autre part sur le paramètre d'écart à la norme. La norme racine étant régimentée, son paramétrage définit un intervalle maximal dans lequel les paramètres peuvent évoluer. L'écart à la norme définit quant à lui les variations autorisées par rapport à la norme. En utilisant ces deux points, le mécanisme de contrôle peut garantir la conformité aux spécifications lors de l'exécution de la simulation.

Nous avons ensuite présenté l'utilisation du modèle comme un outil dans les simulations. Il peut être utilisé sur le paramétrage externe des simulations, en présentant deux avantages intéressants. Tout d'abord, il est non-intrusif, pouvant être utilisé uniquement avec les paramètres existants de la simulation sans modification du fonctionnement de celle-ci. Il permet ensuite de réaliser la conception à l'extérieur de l'agent, rendant celle-ci plus flexible, abordable par un plus grand nombre d'utilisateurs, et plus maintenable. Afin d'utiliser l'outil, les paramètres et les normes doivent être définis. Pour faciliter l'utilisation de l'outil et le processus de configuration, les éléments non intuitifs du modèle sont construits par défaut. Cela en permet une utilisation rapide et aisée.

Enfin, dans un dernier temps nous avons présenté l'utilisation de l'outil sur deux exemples, afin de démontrer la généricité de l'approche. Le premier exemple concerne la simulation de foule, où les critères de variété sont tirés aléatoirement dans des intervalles spécifiques. En construisant des normes basées sur ces paramètres, le comportement des agents peut être généré facilement en utilisant l'outil. Le second exemple concerne la génération d'agents d'espèces différentes. Dans ce cadre, nous nous basons sur une analogie entre les critères définissant une espèce et ceux définissant une norme. Le paramétrage généré depuis une norme définit alors un agent de l'espèce correspondante. Les facilités offertes par l'outil permettent d'expérimenter facilement différents paramétrages afin d'équilibrer la simulation, et de créer une grande variété de comportements d'agents au sein des espèces.

Chapitre 4

Application à la simulation de trafic

Ce chapitre présente l'application du modèle proposé à la simulation de trafic dans les simulateurs de conduite. Cette application se base sur la suite logicielle SCANeR™, développée et utilisée au Centre Technique de Simulation de Renault. Dédiée aux simulateurs de conduite, elle intègre des capacités de simulation de trafic. Nous présentons tout d'abord le fonctionnement de SCANeR™, ainsi que celui de son modèle de trafic. Dans un second temps, nous nous intéresserons à la façon dont le modèle de différenciation comportementale a été intégré à l'application et aux nouvelles fonctionnalités rendues disponibles.

4.1 La suite logicielle SCANeR™

Dans cette première section, nous présentons tout d'abord la suite logicielle SCANeR™ à travers ses applications et son architecture. Nous nous intéressons ensuite aux éléments relatifs à la simulation de trafic, des aspects scénarisation à la description de l'environnement. Enfin, nous détaillons le fonctionnement du modèle de trafic pour les véhicules autonomes et la manière dont les piétons sont pris en compte par l'application.

4.1.1 Présentation

Cette première partie présente l'historique de SCANeR™ et illustre certaines de ses applications. Nous décrivons ensuite l'architecture du logiciel et son fonctionnement général.

4.1.1.1 Historique et applications

La suite logicielle SCANeR™ (pour Simulation de Conduite Automobile Normalisée en Réseau) a été développée initialement par Renault dans le cadre du projet européen Prometheus, en 1990. Le développement se poursuivant, elle s'est enrichie et a évolué sous le nom de SCANeR™ II. En 1997, la commercialisation du logiciel débute, la société Oktal se chargeant de la distribution [19]. Depuis 2005, SCANeR™ est développé à la fois par Renault et Oktal, qui en sont désormais copropriétaires.

19. Plus d'un trentaine d'utilisateurs sont aujourd'hui recensés à travers le monde : constructeurs automobiles (Renault, PSA, Audi), universités (Universita di Napoli, University of Minessota), instituts de recherche (SINTEF en Norvège, CTAG en Espagne), entreprises pour des besoins de formation (Macif) [Oktal, 2009].



FIGURE 4.1 – Le simulateur d'éclairage du Centre Technique de Simulation, vu de l'extérieur (à gauche), et avec un point de vue depuis le poste de conduite (à droite).

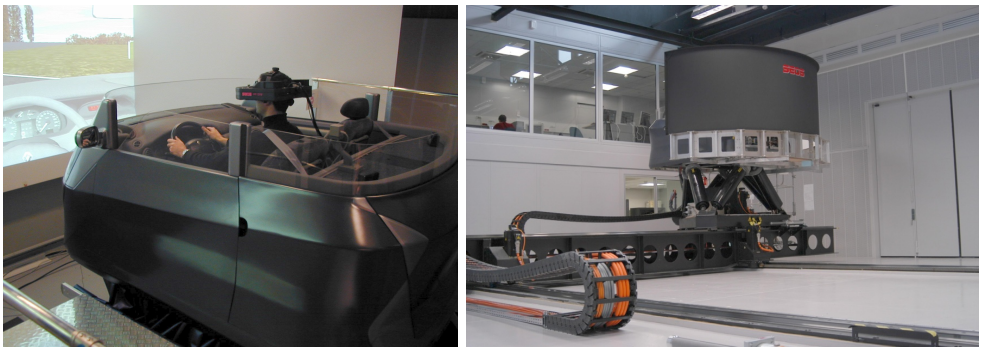


FIGURE 4.2 – A gauche le simulateur CARDS© 2 couplé à un casque de réalité virtuelle, utilisé pour des tests d'ergonomie en situation de conduite. A droite le simulateur Ultimate, qui utilise une plateforme mobile à vérins 6 axes montée sur deux rails $X \cdot Y$.

Centré sur les simulateurs conduite, le champs d'application de SCANeR™ vise principalement l'ingénierie et la recherche. Par exemple, il permet de mettre en œuvre des simulateurs d'éclairage destinés à tester de nouvelles versions de phares sans réaliser de prototype physique. Les cartographies d'éclairage sont réalisées par les concepteurs des phares (typiquement des fournisseurs comme Valéo), puis intégrées directement dans le simulateur (Fig. 4.1). Les experts métier peuvent alors évaluer le niveau de prestation sur simulateur. Cela permet d'accélérer le processus de conception en transformant une répétition de cycles « prototype – essai » longs et coûteux, en un processus purement numérique plus économe en délais et ressources.

Les simulateurs permettent également d'évaluer l'introduction de nouveautés dans les véhicules, par exemple au niveau du design ou des systèmes d'aide à la conduite. Des techniques de réalité augmentée sont utilisées par les ergonomes afin d'effectuer des tests en situation de conduite, comme l'évaluation des angles morts ou de la visibilité (Fig. 4.2). Souvent, ce type d'expérimentation serait impossible sans recours à la simulation, pour des questions de risque et de coût.

Par ailleurs, certains simulateurs intègrent des plateformes mobiles, afin d'améliorer le ressenti physique lors de la simulation. Ces systèmes visent à limiter les incohérences entre les stimuli visuels et kinesthésiques du conducteur, qui peuvent conduire au « mal du simulateur ». Ils permettent d'étendre le champ d'applications aux études de système de contrôle dynamique du véhicule, au confort du conducteur ou aux prestations liées au freinage et à la direction. Chez

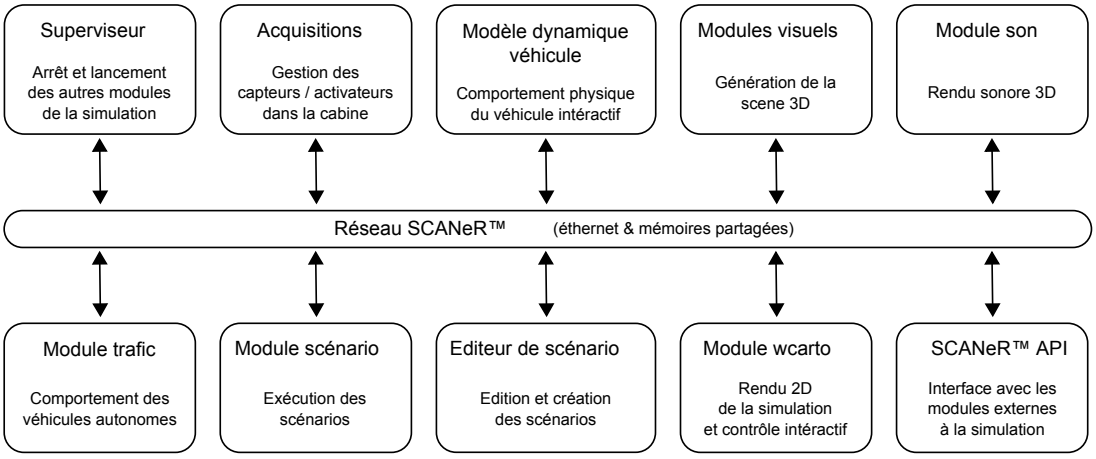


FIGURE 4.3 – Architecture de SCANeR™ : les différents modules peuvent être exécutés sur des machines différentes et communiquent à travers le réseau ou des mémoires partagées.

Renault, deux simulateurs intègrent de tels éléments : le simulateur CARDS© 2, qui utilise une plateforme mobile à vérins 6 axes (Fig. 4.2), et le simulateur haute performance Ultimate, qui utilise une plateforme mobile à vérins 6 axes montée sur deux rails $X \cdot Y$ permettant un déplacement latéral et longitudinal (Fig. 4.2). Ces simulateurs sont par ailleurs les supports de travaux de recherche, par exemple sur l'influence de la restitution du mouvement [Reymond et al., 2001] ou des systèmes à retour d'effort [Toffin et al., 2007], ou encore sur les perceptions des conducteurs [Kemeny & Panerai, 2003].

D'autres usages sont aujourd'hui en forte croissance, comme la formation sur des problématiques environnement et sécurité. Pour ne citer qu'un exemple, les simulateurs peuvent être utilisés afin de former les conducteurs à la conduite économique, en leur montrant l'impact de leur comportement sur leur consommation de carburant.

4.1.1.2 Architecture logicielle

Observons tout d'abord quelle est l'architecture matérielle opérant derrière un simulateur de conduite tel qu'Ultimate. Pour mettre en oeuvre un tel système, il est nécessaire de gérer le rendu visuel, réalisé généralement par au moins trois vidéo projecteurs, le contrôle de la plateforme et sa stratégie de commande, le modèle dynamique véhicule, le trafic autonome, la scénarisation, ou encore le rendu sonore...

La complexité et la multiplicité des configurations possibles a conduit à différents choix logiciel. Tout d'abord, SCANeR™ est conçu suivant une architecture distribuée. Les différents modules de l'application peuvent être exécutés sur des machines différentes, et communiquent à travers le réseau. Les communications se font soit par éthernet, soit par l'intermédiaire de mémoires partagées.

Les principaux modules de l'application réalisent les actions suivantes (Fig. 4.3) :
- le superviseur permet de démarrer et de stopper les différents modules. Il spécifie l'ensemble de la configuration, et en particulier les fichiers de données et les exécutables qui seront utilisés,
- les acquisitions récupèrent les données mesurées par les capteurs disponibles, comme le volant ou les pédales,



FIGURE 4.4 – Une capture d'écran d'une partie des modules disponibles dans SCANeR™. De gauche à droite et de haut en bas : la console d'erreur, le rendu 2D (wcarto), le superviseur, deux visuels et le module de gestion des points de vue caméra.

- le modèle dynamique véhicule et le contrôle plateforme calculent les réactions physiques du véhicule et les restituent au conducteur,
- le module visuel calcule la scène 3D qui est présentée au conducteur,
- le module son effectue le rendu sonore de l'environnement, incluant la circulation et le véhicule (bruits moteur, bruits aérodynamiques...),
- le module trafic simule des véhicules autonomes évoluant autour du véhicule conduit,
- l'éditeur de scénario permet la création et la modification du trafic, ainsi que la scénarisation de l'expérimentation,
- le module scénario exécute ces scénarios,
- et enfin le module wcarto permet de visualiser la simulation en deux dimensions et de contrôler le trafic en cours d'exécution.

La Figure 4.4 présente la mise en œuvre de certains de ces modules.

Comme nous l'avons vu dans la section 1.1.2 (p. 10), le contexte des simulateurs implique la prise en compte de différentes contraintes sur les temps de calcul. Par exemple, les performances sont importantes au niveau du visuel pour fournir un rendu fluide et de qualité, mais le modèle dynamique véhicule et le contrôle plateforme nécessitent du temps réel dur. Il s'agit d'une fonctionnalité importante pour les utilisateurs qui souhaitent évaluer des systèmes réels sur simulateur. De plus, le temps réel permet de garantir la sécurité du système au niveau logiciel. Différentes techniques sont donc mises en œuvre pour répondre à ce besoin, comme l'utilisation de noyaux spécifiques comme RTX [20] ou de mémoires SCRAMNeT [21].

Par ailleurs, un point important dans SCANeR™ est l'ouverture de son architecture. Les communications sont basées sur un protocole commun, et une API permet d'interagir avec l'ensemble des modules. Cette API est fournie aux utilisateurs et rend possible l'interfaçage d'applications

20. RTX est une extension de Microsoft Windows dédiée aux applications temps réel. Il fonctionne en mode noyau en parallèle du noyau NT.

21. Les SCRAMNeT, ou Shared Commom Random Access Memory Network, sont des mémoires volatiles de type RAM partagées par réseau de fibre optique.

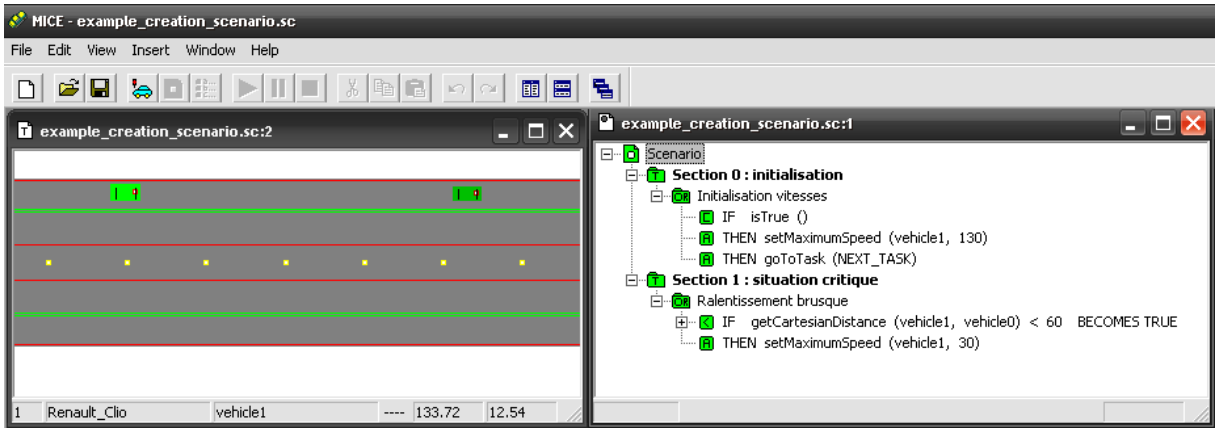


FIGURE 4.5 – Le module de création et d'édition de scénario `scenario`, présentant à droite une règle de scénarisation dans le langage fonctionnel utilisé par SCANeR™.

existantes ou le développement de modules spécifiques. Par exemple, des modèles dynamique véhicule du commerce peuvent ainsi facilement être couplés à SCANeR™. L'API permet d'intervenir sur l'ensemble des éléments de la simulation, du contrôle de la plateforme à celui des véhicules autonomes, permettant l'intégration de fonctionnalités avancées sans modification du noyau de l'application. Nous utilisons d'ailleurs ses possibilités pour les développements réalisés au cours de nos travaux.

4.1.2 La simulation de trafic dans SCANeR™

La simulation de trafic dans SCANeR™ est basée sur deux fonctionnalités principales : les capacités de scénarisation d'une part, et le modèle de trafic d'autre part. Nous présentons ici les modules assurant ces fonctions, ainsi qu'un élément fondamental sur lequel repose le comportement des véhicules autonomes : la description de l'environnement.

4.1.2.1 Édition et exécution des scénarios

Deux modules sont dédiés à la gestion des capacités de scénarisation de l'application. Le premier est chargé de l'exécution des scénarios (module `scenario`). Son rôle est d'interpréter les données qui ont été préparées lors de la conception du scénario, et d'envoyer les informations adéquates aux autres modules. L'exécution se fait de manière séquentielle : un scénario est constitué d'un ensemble de règles, regroupées en tâches. Une tâche est exécutée entièrement (i.e. toutes ses règles sont testées) à chaque pas de temps, et ce tant qu'elle est active. Les tâches sont activées et désactivées en utilisant des mécanismes du type `goto` (Fig. 4.5).

Le second module dédié à la scénarisation, `mice`, est destiné à la création et à l'édition des scénarios. Il propose une interface graphique permettant de spécifier les conditions initiales de la simulation. Il est possible de choisir la base de donnée qui sera utilisée et de spécifier la position, le type, l'itinéraire et le paramétrage des véhicules présents initialement dans le scénario. Notons que ces véhicules doivent être créés manuellement un à un, et leurs paramètres modifiés individuellement.

L'interface de `mice` permet par ailleurs de concevoir les règles de scénarisation, en utilisant soit le langage fonctionnel de l'application, soit le langage Python. Des règles d'action peuvent

être spécifiées : la règle présentée dans la Figure 4.5 permet d'obliger le véhicule 1 à ralentir à 30 km/h lorsque le véhicule 0 s'approche à moins de 60 mètres de lui. La situation créée est celle d'un véhicule freinant brusquement devant le véhicule 0. Cela permet de créer des situations accidentogènes et d'étudier la réaction des conducteurs.

Les règles accessibles par scénario sont variées, et permettent d'influer sur de nombreux éléments de la simulation. On pourra par exemple :

- modifier les conditions météorologiques : il se met à pleuvoir lorsque le véhicule passe à un point particulier,
- adapter l'interface présentée au conducteur, en insérant des images à la volée dans le visuel. Cela permet de personnaliser l'affichage et de simuler par exemple des systèmes d'aide à la conduite,
- exporter des données afin de les enregistrer.

Actions et évènements peuvent être couplés et enchaînés, ce qui permet de concevoir des scénarios complexes.

Notons que la scénarisation du comportement des véhicules du trafic permet de s'affranchir des règles du modèle de décision des agents. En effet, il est possible d'imposer à un véhicule de ralentir comme dans la règle présentée Figure 4.5, mais ce ralentissement ne se fera que dans les limites permises par le modèle de décision. Dans certains cas, cela ne permet pas d'obtenir l'effet désiré par le scénariste. Des règles supplémentaires permettent donc de forcer les véhicules à agir, même si cela doit les conduire à violer leur modèle de décision. Ce point est important pour le modèle de trafic : on cherche à avoir un trafic autonome, mais contrôlable au maximum et capable de réagir de manière adéquate face à des véhicules pilotés par scénario et forcés à violer les règles de comportement. C'est un problème complexe, auquel le modèle de trafic cherche à apporter une solution.

4.1.2.2 Fonctionnement du module trafic

Le module trafic gère le comportement des véhicules autonomes dans l'application. Comme nous l'avons vu dans le Chapitre 1 (Sect. 1.1.2 p. 9), les contraintes de la simulation de conduite imposent l'utilisation d'une simulation microscopique. Le module trafic de SCANeR™ utilise ainsi une architecture multi-agents [Champion et al., 1999].

Chaque véhicule est un agent, qui décide de ses actions de manière indépendante [22]. Ces agents peuvent être considérés comme hybrides : sans être cognitifs, car ne possédant pas de gestion évoluée de leurs buts, ils utilisent des formes de mémoire et de raisonnement qui vont au-delà du purement réactif. Ils évoluent dans un environnement spatialement situé, fortement contraint par l'infrastructure routière.

Les conducteurs réels sont associés à des véhicules particuliers dans la simulation, appelés véhicules interactifs. Ces véhicules sont pris en compte par le trafic, mais leur données sont mises à jour par les acquisitions et non par le modèle de trafic. Plusieurs véhicules interactifs peuvent être présents simultanément dans la simulation, et il est donc possible de faire conduire plusieurs conducteur humains dans le même scénario.

Le trafic peut être distribué entre différentes machines afin de répartir la charge de calcul. Pour ce faire, un ou plusieurs modules trafic doivent être lancés sur chacune des machines utilisées. La répartition des véhicules se fait lors de leur création, à l'aide de tables statiques les

22. Le modèle de décision est détaillé dans la Section 4.1.3

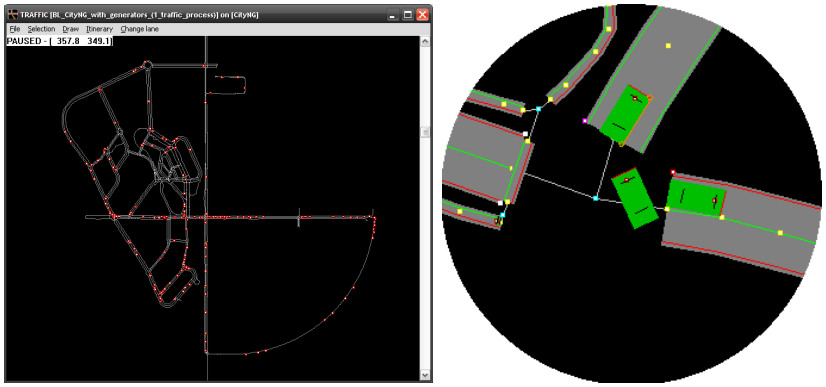


FIGURE 4.6 – Rendu visuel 2D du trafic routier avec le module `wcarto` : à gauche une vue globale d'une simulation sur réseau urbain (les points rouges sont les véhicules), à droite la vue détaillée d'une intersection.

affectant à un module spécifique. Il n'y a pas de répartition dynamique de la charge. Les différents modules mettent à jour les véhicules qui leur ont été confiés, puis envoient les informations sur le réseau.

La synchronisation entre les modules n'est pas gérée de manière explicite. Des désynchronisations peuvent donc survenir en cas de surcharge du réseau, ou si une des machines à laquelle est affecté un module est en surcharge. Ces différents évènements doivent être gérés manuellement, mais en pratique ce point ne pose pas problème car les capacités matérielles sont volontairement surdimensionnées. Notons que pour pallier ce type d'incident, le module visuel est capable d'extrapoler la prochaine position des véhicules autonomes afin d'éviter des discontinuités dans l'affichage.

Enfin le module de rendu 2D `wcarto` permet le chargement des scénarios et la visualisation du trafic en temps réel (Fig. 4.6). Il permet également d'interagir avec la simulation en éditant les propriétés des véhicules, ou en les déplaçant manuellement.

4.1.2.3 Description de l'environnement

Il existe plusieurs façon de représenter l'environnement dans les simulateurs de conduite (cf. Sect 1.1.3 p. 12). Le choix fait dans SCANeR™ est de baser le modèle de décision des véhicules autonomes sur une description logique de l'environnement. Actuellement, deux formats de description de données cohabitent : l'association RNS (Road Network and Signs) et RS (Road Surface) d'une part [Oktal, 2007a], et le format RND (Road Network Description) d'autre part [Oktal, 2007b]. Le RNS contient une description discontinue de l'environnement, qui est utilisée par les véhicules autonomes pour récupérer les données logiques concernant l'infrastructure routière. Le RS est une description continue dédiée au modèle dynamique du véhicule interactif (Fig. 4.7). Enfin, le RND fusionne de ces deux niveaux de description au sein d'un format unique.

Le RNS définit la représentation de l'ensemble des routes de la base de données. Les routes sont représentées par un ensemble de points discrétisés depuis leur axe central, et sont connectées entre-elles par des nœuds. Le RNS contient en outre tous les éléments logiques les décrivant : nombre et largeur des voies, direction de conduite, voies réservées à un type de véhicule... Le type et la position des éléments de signalisation sont également intégrés (panneaux, feux et marquage au sol).

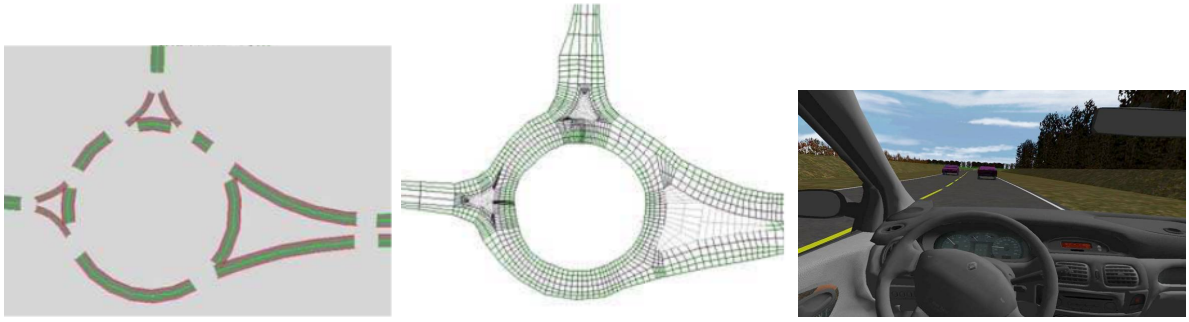


FIGURE 4.7 – Les bases de données dans SCANeR™ : de gauche à droite la description logique pour les véhicules autonomes (RNS), la description continue pour les véhicules interactifs (RS), et la vue 3D pour le conducteur.

Le RS utilise des surfaces de Béziers afin de décrire la surface de la route. Elles sont nécessaires afin d'avoir une dérivée continue entre les différentes surfaces, et permettre au simulateur de calculer sa position sans discontinuité. Rouler sur une description graphique (i.e. polygonale) de la base n'est en effet pas envisageable avec des simulateurs dynamiques : des variations brusques risquent d'être ressenties par le conducteur au passage sur un nouveau polygone, alors qu'aucune différence visuelle n'apparaît. Le RS inclut par ailleurs les caractéristiques propres à la surface de la route : adhérence, nature (asphalte, graviers...), type (normal, trottoir...) et régularité de la surface (lisse, rugueux...). Ces éléments sont exploités par le modèle dynamique véhicule.

Cette séparation en deux couches distinctes avait été choisie pour des problèmes de performance : l'utilisation d'une description continue pour calculer le déplacement des véhicules autonomes était trop coûteuse en temps de calcul. Cependant, la description discontinue de la route entraîne certaines limitations sur le rendu graphique et le comportement des véhicules. Avec l'augmentation de la puissance de calcul disponible, les compromis rendus nécessaires par la séparation RS / RNS ne sont plus nécessaires.

Une nouvelle description du réseau routier, le RND, a donc été introduite afin de résoudre les difficultés rencontrées avec le couple RNS / RS et d'offrir plus de flexibilité et d'extensibilité [Lacroix et al., 2007]. Le RND fusionne la couche logique du RNS avec la description continue propre au RS. Cette unification signifie que le trafic et le modèle dynamique véhicule fonctionnent désormais en utilisant des données identiques. Par ailleurs, le format est conforme à la spécification OPENDRIVE® [Dupuis & Grezlikowski, 2006], qui est une proposition de standardisation destinée à faciliter l'échange de données entre différentes plateformes de simulation de conduite.

En ce qui concerne la base de données graphique, elle est découplée de la description logique. Le processus de création des bases de données est généralement réalisé à l'aide d'outils dédiés : dans le cas de SCANeR™, il s'agit de l'application Evariste©. Classiquement le processus de création d'une base débute par la préparation de la logique de l'infrastructure routière. Une fois cette structure logique fixée, différents éléments sont générés : la description logique (RNS + RS ou RND), et une version simple de la 3D. Des infographistes interviennent alors afin d'améliorer la qualité du rendu graphique de la base, par exemple en ajoutant les textures correspondant à un environnement réel. Notons que comme il n'existe pas de lien physique entre les descriptions logique et graphique, il faut veiller à ne pas décorréler les différentes couches.

Les propriétés offertes par cette méthode de description de l'environnement offrent ainsi une solution sur laquelle peuvent s'appuyer les véhicules autonomes, les véhicules interactifs et les

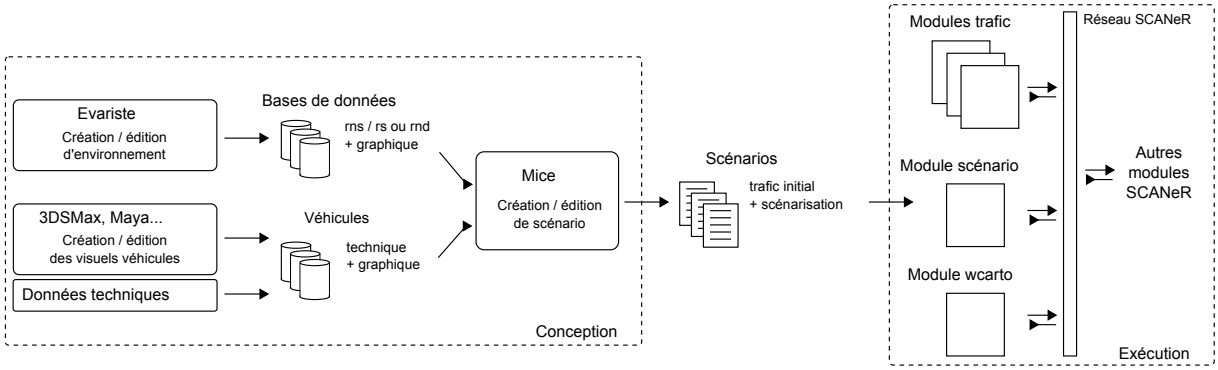


FIGURE 4.8 – Chaîne de conception et d'exécution des scénarios.

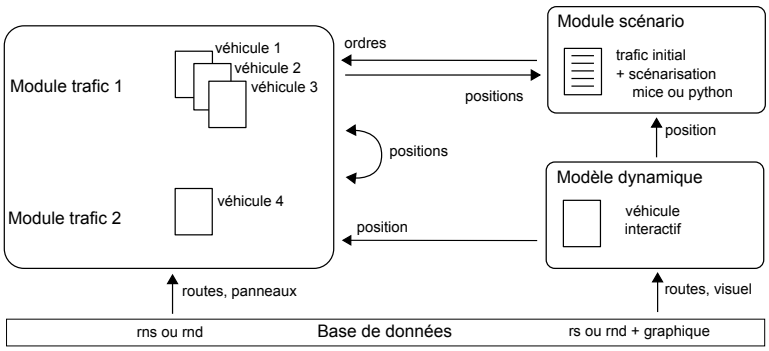


FIGURE 4.9 – Interactions entre les entités simulées (l'ensemble des messages transitent sur le réseau SCANeR™).

conducteurs réels.

4.1.2.4 Interactions entre les modules

Le fonctionnement du trafic dans SCANeR™ peut être résumé par les Figures 4.8 et 4.9. La Figure 4.8 représente la chaîne de conception et d'exécution des scénarios. Lors de la phase de conception, le designer construit un scénario utilisant une base de données et des véhicules pré-existants ou conçus spécifiquement. Le travail dans l'éditeur permet de construire l'état initial du trafic, ainsi que les règles de scénarisation qui s'appliqueront pendant la simulation. Ces données sont sauvegardées sous forme d'un scénario, avant d'être utilisées par les modules trafic et scénario à l'exécution.

La Figure 4.9 schématise les interactions entre les entités simulées. Différents modules trafic peuvent être présents dans la simulation, chacun mettant à jour un ensemble dédié de véhicules. Tous les modules reçoivent les données mises à jour, par l'intermédiaire du réseau. Les véhicules autonomes peuvent être contraints par des règles de scénario, et des véhicules interactifs représentant les conducteurs réels sont intégrés dans la boucle. Leur mise à jour est faite directement par les acquisitions provenant des simulateurs.

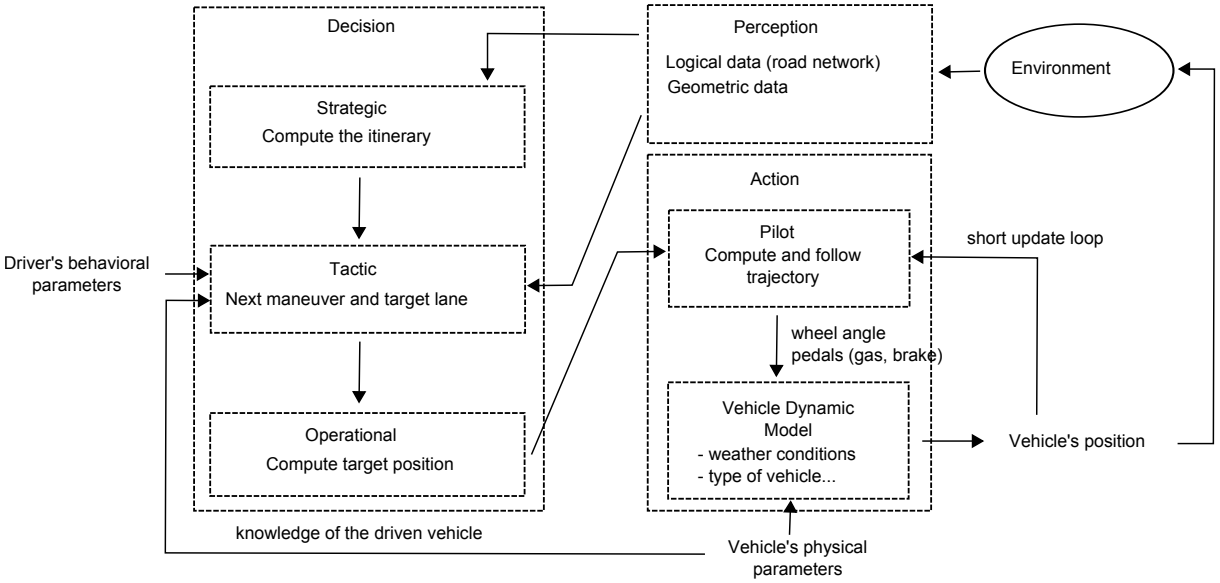


FIGURE 4.10 – Architecture des agents.

4.1.3 Le modèle de trafic

Dans cette partie, nous détaillons le modèle de décision utilisé par les véhicules autonomes dans SCANeR™. Il est articulé en trois phases « perception – décision – action » (Fig. 4.10). Durant la phase de perception, les véhicules mettent à jour leur connaissance de l'environnement. La phase de décision les conduit à déterminer leur prochain but, et la phase d'action à calculer leur prochaine position.

4.1.3.1 Perceptions

Les véhicules utilisent différentes méthodes pour mettre à jour leur représentation de l'environnement. Dans tous les cas, cette représentation reste partielle, les agents n'ayant qu'une vue locale de l'environnement.

Tout d'abord, ils se basent sur la description logique du réseau afin de récupérer des informations sur l'infrastructure routière dans leur voisinage. Lors de cette phase, les agents observent le tronçon de route sur lequel ils se trouvent, ainsi que le prochain tronçon appartenant à leur itinéraire. Cela leur permet de déterminer la courbure de la route, la courbure entre le tronçon courant et le suivant (cas d'une intersection par exemple), les limitations de vitesse, les panneaux et la proximité d'une intersection.

La perception des autres agents s'appuie également sur la structure hiérarchique du réseau routier. Un agent détecte ainsi les véhicules évoluant sur un chemin commun au sien : même tronçon, tronçon précédent ou tronçon suivant. Si l'agent approche d'une intersection, les véhicules se dirigeant vers la même intersection que lui sont également détectés. Une fenêtre suivant la courbure de la route et proportionnelle à la vitesse est utilisée dans ce but (Fig. 4.11). Un conducteur conduisant à une vitesse élevée observe ainsi à une plus grande distance devant lui, ce qui lui permet d'anticiper. Les piétons sont perçus en utilisant un calcul de distance cartésienne. En effet, ils n'évoluent pas sur les mêmes voies : la détection logique n'est pas en mesure de les répérer.

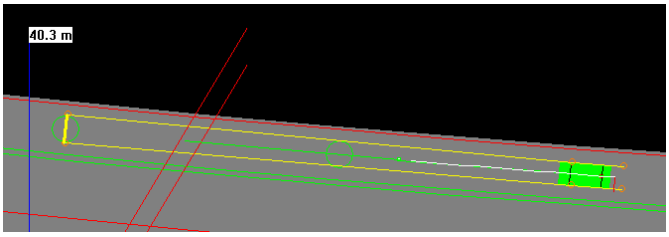


FIGURE 4.11 – Perception des agents.

Enfin un radar à courte portée permet de détecter les véhicules n'ayant pas été perçus par les autres modes et qui sont susceptibles de créer une situation critique. En effet, la perception logique utilisée ne permet pas de détecter les véhicules se trouvant à des positions inhabituelles, par exemple s'ils ne sont pas sur le réseau routier. Ces situations sont détectées en calculant le temps avant collision, utilisant la distance et les directions respectives des véhicules. Si une collision est imminente, un indicateur d'urgence déclenche une réaction spécifique lors de la phase de décision.

Notons finalement que les occlusions ne sont pas prises en compte dans les perceptions. En effet, la description logique de la route ne contient pas les éléments permettant de réaliser ces calculs. De même, les éléments de décors masqués par d'autres véhicules sont perçus par les agents : par exemple, un panneau caché par un camion sera pris en compte.

4.1.3.2 Décision

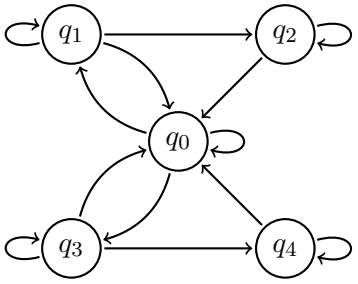
Une fois leur représentation de l'environnement mise à jour, les agents déterminent la prochaine action qu'ils vont exécuter. Cette phase de décision s'articule en trois niveaux : stratégique, tactique et opérationnelle. Ils correspondent au modèle de conducteur proposé par Michon [Michon, 1985].

Le niveau stratégique correspond au choix du but global dans le réseau routier. Dans SCANeR™, l'agent utilise ce niveau pour déterminer la prochaine route qu'il va emprunter. En pratique, soit un itinéraire a été défini à la création de l'agent, soit celui-ci choisit son chemin de manière aléatoire à chaque intersection. Il n'y a pas de planification dynamique en fonction d'une destination à atteindre.

Dans le niveau tactique, l'agent sélectionne son but à court terme en combinant le but provenant du niveau stratégique et les contraintes environnementales. L'objectif est d'adapter son déplacement dans le réseau, pour utiliser de manière efficiente changements de ligne, dépassements... Le calcul est basé sur un automate à états finis, permettant de choisir entre différentes possibilités (Fig. 4.12) :

- « aller tout droit » : l'agent continue tout droit en conduisant à sa vitesse désirée, ou en suivant le véhicule précédent s'il y en a un,
- « se rabattre » : l'agent change de ligne, soit pour se placer le plus à droite possible sur la chaussée, soit pour se préparer à sortir d'une autoroute ou effectuer un placement correct à l'entrée d'une intersection,
- « déboîter » : l'agent change de ligne afin de se placer correctement ou pour dépasser un véhicule unique ou une file de véhicules.

Jouant le rôle de mémoire à court terme, des états de transition sont utilisés pour permettre aux agents de poursuivre une action enclenchée (« préparer rabattement » et « préparer déboi-



q_0 : Aller tout droit
q_1 : Préparer rabattement
q_2 : Se rabattre
q_3 : Préparer déboîtement
q_4 : Déboîter

FIGURE 4.12 – L'automate à état fini utilisé dans le modèle de décision des agents.

tement »). Notons que tous les états peuvent être forcés par scénario, en violation ou non du modèle de décision.

Les transitions entre les différents états sont basées sur une fonction de « scoring » dépendant d'éléments variés, qui vont de l'espace disponible devant le véhicule à la courbure de la route. Par exemple, la transition de l'état « aller tout droit » à l'état « préparer déboîtement » utilise les critères suivants :

```
scoring(déboitement) gagne &
non(peut déclencher rabattement) &
( déboitement forcé || temps dernier rabattement > 5 s )
```

Cette règle force les véhicules à rouler le plus à droite possible. Notons que ces fonctions sont basées sur une calibration empirique.

Enfin, au niveau opérationnel, l'agent effectue la manœuvre choisie. Il calcule l'accélération et l'angle volant qui lui permettront d'atteindre le but court terme. Par exemple, si le véhicule est dans l'état « aller tout droit », l'accélération choisie est le minimum des accélérations provenant de l'accélération maximum du véhicule, d'un freinage d'urgence, d'un freinage lié à la courbure de la route ou de la signalisation, et de l'accélération calculée par le modèle de suivi de véhicule. Cette dernière utilise le modèle suivant :

$$a = \alpha \cdot (\Delta d - t \cdot v) + \beta \cdot \Delta v$$

où a est l'accélération résultante du suivi de véhicule, Δd et Δv sont les différence de distance et de vitesse avec le véhicule suivi, $t \cdot v$ est la distance de sécurité (t est un paramètre du modèle) et α, β sont des variables d'ajustement. Le calcul de l'accélération dans les états de déboîtement et de rabattement est similaire. La principale différence est le choix du véhicule suivi : il est dans la voie cible plutôt que dans la même voie.

Finalement l'angle volant est calculé à partir de la courbe d'Hermite [23] joignant le milieu de l'essieu arrière et le but. On vérifie par ailleurs que l'angle volant maximal n'est pas dépassé, et qu'il n'y a pas de variation trop rapide (moins de 20 % de l'angle maximal par pas de temps). Notons que le mode de calcul permet aux véhicules de couper les virages, et qu'un facteur de déviation au centre de la voie est pris en compte afin de simuler le fait que les conducteurs ne restent pas en permanence au milieu de leur voie.

23. Les courbes d'Hermite (ou « splines » de degré 3) permettent d'approcher des contours complexes en utilisant une interpolation basée sur une description polynomiale de degré 3, et d'obtenir une continuité $\mathcal{C}^2$.

4.1.3.3 Action

L'accélération et l'angle volant issus de la phase de décision permettent à l'agent de calculer sa nouvelle position. La phase d'action est décomposée en deux tâches, effectuées par le pilote et le modèle dynamique. Le pilote joue le rôle d'interface entre les sorties de la phase de décision et le modèle dynamique. Il recalcule la trajectoire rejoignant le but en fonction des paramètres physiques du véhicule. Il se base ensuite sur un algorithme géométrique afin de rester à proximité de cette trajectoire, en utilisant une régulation par boucle courte sur la position et la vitesse. Sa sortie est un point cible.

Un fois le point cible déterminé, le calcul final est effectué en utilisant un modèle dynamique du véhicule. Ce type de modèle est utilisé lors de la conception des véhicules réels [24], et prend en compte les liaisons au sol, le comportement du moteur et des roues motrices, le comportement des roues non motrices ou encore les efforts au niveau de la direction et du volant.

Ces modèles utilisent de nombreux paramètres en entrée, comme par exemple :
- les actions conducteur (angle volant, position des pédales, rapports de vitesse...)
- les caractéristiques de l'environnement routier : adhérence, conditions climatiques, profil de la route...
- les paramètres physique du véhicule : masse des roues, des ressorts, couple moteur...

La validité de ces modèles est mesurée sur des manœuvres élémentaires de conduite comme le changement de file ou la prise de virage. Une fois validés, ils permettent de tester différents paramétrages en utilisant des données simulées ou en rejouant des données enregistrées sur piste. Le déplacement du véhicule interactif dans SCANeR™ est géré par MADA, l'outil officiel de Renault.

Dans le cas des véhicules autonomes, le modèle dynamique utilise des équations dynamiques simplifiées afin de limiter la charge de calcul et le nombre de paramètres nécessaires. En effet, intégrer un modèle dynamique véhicule complet serait sur-dimensionné par rapport au besoin. Cependant le modèle utilisé modélise chaque élément du véhicule, ce qui permet de calculer une dynamique réaliste. Par exemple, les phases d'accélération ou de décélération prennent en compte la raideur des amortisseurs, le franchissement de dos ânes et autres obstacles est simulé de manière réaliste et les véhicules multi-essieux sont pris en compte. A la fin de cette phase d'action, l'agent a déterminé sa nouvelle position.

4.1.3.4 Paramétrage du modèle

Ces phases de perception, décision et action sont communes à l'ensemble des agents, qui les effectuent à chaque itération du module trafic. La diversité des comportements est obtenue à travers le paramétrage du modèle, qui est accessible lors de l'initialisation, puis par scénario lors de l'exécution. Le modèle des véhicules repose sur deux types de paramètres : des paramètres physiques pour les véhicules, et des paramètres pseudo-psychologiques pour les conducteurs.

Les paramètres physiques sont utilisés par le modèle dynamique véhicule. Ils concernent les limites physiques qui peuvent être atteintes par un véhicule donné. On y trouvera ainsi la dimension du véhicule, sa vitesse maximale, ses accélérations et décélérations maximales, ou encore l'angle volant maximal.

24. De nombreux modèles sont disponibles : on peut citer MADA utilisé chez Renault, Callas édité par Oktal, IPG-CARMAKER utilisé par exemple chez Audi, ou encore AMESIM.

paramètre	valeurs possibles	valeur par défaut	agit sur
maximal speed	$[0, \infty]$ km/h	130 km/h	vitesse maximale véhicule
safety time	$[0, \infty]$ s	2 s	distance de sécurité
overtakink risk	$[-1, 2]$	0	prise de risque dépassement
speed limit risk	$[0, \infty]$	1	respect limites de vitesse
observe priority	$\{true, false\}$	$true$	respect des priorités
observe signalization	$\{true, false\}$	$true$	respect de la signalisation

TABLE 4.1 – Les paramètres pseudo-psychologiques du modèle de décision des agents.

Les paramètres pseudo-psychologiques interviennent au niveau de la phase de décision, et influencent le niveau tactique (Tab. 4.1) :

- la « vitesse maximale » décrit la limite maximale de vitesse pouvant être utilisée par le conducteur. Il s'agit d'un nombre réel pouvant prendre sa valeur entre 0 et $+\infty$ km/h. Par défaut, la vitesse maximale vaut 130 km/h,
- le « temps de sécurité » reflète la marge de sécurité adoptée a minima par le conducteur. Multipliée par la vitesse, elle permet d'obtenir la distance de sécurité utilisée à un instant donné. Ce paramètre peur prendre sa valeur entre 0 et $+\infty$ secondes. Sa valeur par défaut est 2 secondes,
- l'« overtaking risk » décrit le risque qu'un conducteur sera prêt à prendre lors d'un dépassement. Il est utilisé comme facteur correctif sur la distance minimale acceptée pour le véhicule arrivant en face. Il peut prendre une valeur réelle entre -1 et 2, -1 décrivant un conducteur prudent, 0 un conducteur normal et 2 un conducteur agressif. Sa valeur par défaut est 0,
- le « speed limit risk » autorise les véhicule à dépasser les limites de vitesse. Il s'agit d'un nombre entre 0 et $+\infty$, qui est utilisé comme coefficient de proportionnalité sur la limite de vitesse de la route. Un facteur supérieur à 1 autorise donc à dépasser les limites. Sa valeur par défaut est 1,
- le « respect des priorités » permet au véhicule de ne pas tenir compte des priorités à droite. Il s'agit d'un booléen : s'il vaut faux, le véhicule ne respecte aucune priorité, s'il vaut vrai il les respecte toutes. Par défaut, sa valeur est vrai,
- enfin le « respect de la signalisation » autorise le véhicule à ne pas tenir compte des panneaux et feux. Le fonctionnement est le même que pour le respect des priorités, mais la valeur fausse conduit au non respect des stops, cédez le passage et feux.

Les paramètres disponibles permettent donc d'agir sur de nombreux aspects du comportement des conducteurs, en particulier relatifs à la prise de risque et au respect du code de la route.

4.1.4 Le cas des piétons

Outre les véhicules, un autre type d'agents est présent dans les simulations : les piétons. Dans la simulation de conduite, ceux-ci peuvent jouer deux rôles particuliers. Tout d'abord, ils peuvent être utilisés afin de créer des situations accidentogènes : un scénario classique est un piéton surgissant sur la chaussée, alors qu'il était masqué par un véhicule stationné. Prendre en compte cette possibilité implique qu'il est nécessaire de pouvoir scénariser les actions des piétons tout comme celles des véhicules. Cependant la plupart des piétons présents dans la simulation ne seront jamais amenés à interagir de manière directe avec le trafic. Il s'agit alors de piétons

d'« ambiance », présents afin de « peupler » les bases de données et augmenter le réalisme.

Dans SCANeR™, ces deux types de piétons sont considérés de manière distincte. Tout d'abord, les piétons destinés à interagir avec le trafic routier utilisent le même modèle de décision que les véhicules. Ils possèdent cependant un paramétrage particulier, et des modèles graphiques adaptés. Des animations spécifiques sont ajoutées afin que les déplacements soient réalistes, à l'aide technologies comme Cal3D [Cal3D, 2009]. Cela permet d'ajouter de manière transparente le mouvement des membres, ou d'adapter la démarche des piétons en fonction de leur vitesse. L'implémentation similaire des piétons et des véhicules présente l'avantage de pouvoir les scénariser en utilisant une syntaxe et un éditeur communs. Notons que cet amalgame, qui peut paraître contre intuitif, est lié à l'implémentation originale des piétons dans l'application. La structure de plus haut niveau étant le véhicule, elle a été utilisée comme une classe abstraite « agent » aurait pu l'être, conduisant aujourd'hui à une confusion sémantique.

En ce qui concerne les piétons d'ambiance, la problématique principale est de ne pas impacter les performances. Le choix réalisé a donc été de les simuler en utilisant un modèle purement réactif basé sur les *steering behaviors* proposés par Craig Reynolds [Reynolds, 1999]. L'implémentation utilise les bibliothèques du projet OpenSteer [Reynolds, 2004] mettant en application les travaux de Reynolds. Ce modèle est tout à fait adapté au besoin, permettant d'obtenir des comportements réalistes tout en étant peu coûteux en temps de calcul : plusieurs centaines de piétons peuvent être présents simultanément sans impact notable sur les performances. La principale limitation se situe au niveau de la charge de calcul pour le rendu graphique. En effet, l'affichage des piétons est coûteux, et il est donc nécessaire de prendre en compte cet élément lors de la conception des scénarios.

Le comportement des piétons n'était pas l'objectif applicatif principal de nos travaux, mais nous reviendrons dans les perspectives sur les possibilités d'utilisation du modèle dans leur cas (Sect. 5.5.2 p. 140).

4.2 Application du modèle à SCANeR™

Après cette description de SCANeR™, intéressons nous à la façon dont le modèle de différenciation comportementale est utilisé dans l'application, ainsi qu'aux nouvelles fonctionnalités qu'il apporte. Dans cette section, nous présentons tout d'abord les objectifs poursuivis par l'intégration du modèle dans SCANeR™. Nous décrivons ensuite comment l'implémentation a été réalisée, avant de présenter les modules développés.

4.2.1 Objectif

Cette première partie introduit la problématique ciblée par l'utilisation du modèle, ainsi que la solution qu'il permet de lui apporter.

4.2.1.1 Problématique

Dans SCANeR™, le comportement des véhicules autonomes repose sur un paramétrage qui est défini lors de leur création dans l'éditeur de scénario (cf. Sect. 4.1.2 p. 99). Dans cet éditeur, la création des véhicules est manuelle : il faut choisir une position cible, puis ajouter un véhicule. De plus, tous les véhicules créés se voient attribuer par défaut un modèle physique identique.

Pour ne pas avoir un unique type de véhicule dans la simulation, il est donc nécessaire d'éditer chacun d'eux manuellement. Un double clic sur le véhicule donne accès à ses propriétés, parmi lesquelles se trouvent le modèle physique et les paramètres pseudo-psychologiques. Il est alors possible de les modifier.

Par ailleurs, l'effet de certains paramètres pseudo-psychologiques est cumulatif. Le choix de valeurs inadéquates peut conduire à l'apparition de comportements inattendus, sans que le concepteur n'en ait nécessairement conscience lors de la spécification. En effet, les relations d'interdépendance et les risques associés sont mal connus des utilisateurs. Par sécurité, ils se contentent donc souvent d'utiliser les paramètres par défaut, ne faisant éventuellement varier que la vitesse maximale.

La conséquence de ces deux points est que bien souvent les scénarios ne contiennent qu'un nombre restreint de véhicules, et que la grande majorité des véhicules conservent le paramétrage par défaut. Ajouter un grand nombre de véhicules, et modifier leurs paramètres, est en effet un travail long et répétitif, que les concepteurs ne sont pas nécessairement prêts à réaliser. Ils se contentent d'ajouter les véhicules nécessaires à l'expérimentation souhaitée, sans inclure de véhicules supplémentaires destinés au trafic ambiant.

Or le faible nombre de véhicule est un problème pour l'immersion, les conducteurs réels ayant l'impression de rouler dans des environnements vides. De plus, un paramétrage identique pour la plupart des véhicules fait qu'ils ont tous le même comportement, ce qui conduit à des situations peu réalistes : par exemple tous les véhicules se suivent à égale distance sur la route sans chercher à doubler, car ils utilisent tous la même vitesse maximale et la même distance de sécurité. L'utilisation de l'outil, qui fournit pourtant la possibilité de créer des environnements riches et variés, conduit à la production de simulations au réalisme limité car les fonctionnalités par défaut ne sont pas adaptées au besoin.

4.2.1.2 Une solution : les styles de conduite

L'introduction de styles de conduite au sein du trafic routier est un facteur d'immersion en simulation de conduite [Wright et al., 2002]. Les utilisateurs sont en effet capables de percevoir et d'identifier différents types de personnalités simulées par les véhicules du trafic, et cela augmente leur impression de réalisme. Ces styles de conduite reflètent le fait qu'un conducteur puisse être agressif, prudent, etc. Ils correspondent à un ensemble de traits de personnalité qui, de par leur cohérence, font apparaître une personnalité identifiable.

Introduire des styles de conduite dans SCANeR™ permet de résoudre la problématique liée à l'interdépendance des paramètres : au lieu d'avoir à définir les paramètres un à un, au risque de faire des erreurs liées à leur corrélation, les utilisateurs choisissent le style de conduite souhaité pour le conducteur virtuel. La spécification par style est suffisante dans la majorité des cas, et en particulier pour le trafic ambiant, bien que moins précise que la spécification par paramètre : l'utilisateur se repose en effet sur le système pour le choix des valeurs finales. Il est toutefois nécessaire de conserver la possibilité de pouvoir modifier manuellement chacun des paramètres des véhicules, au cas où le concepteur souhaite les définir finement.

De plus, automatiser l'utilisation des styles de conduite lors de la création des véhicules résout la problématique des paramétrages véhicules identiques. Si un style de conducteur normal est attribué à chaque véhicule lors de sa création, le concepteur ne sera pas surpris par l'apparition de comportements inattendus. Ce style se décline pourtant par un comportement légèrement

différent pour chacun des agents, ce qui permet de ne pas voir apparaître des situations comme celle citée plus haut, où tous les véhicules se suivent sans interagir.

Ces fonctionnalités simplifient suffisamment la création des véhicules pour ne plus restreindre le concepteur lors de la création des véhicules ambiants. Les opérations, plus légères et plus rapides, permettent de créer facilement un grand nombre de véhicules. Le problème des environnements qui paraissent vides trouve donc sa solution ici. De plus, des outils automatisant la création des véhicules elle-même peuvent également être fournis.

4.2.1.3 Retour au modèle

Le modèle de différenciation comportementale permet de répondre à ces différentes problématiques, aussi bien du point de vue conception que du point de vue utilisation.

Les styles de conduite sont des « classes » de conducteurs qui possèdent des traits de personnalité similaires. Ces traits de personnalité sont décrits par les paramètres du modèle de trafic. Les styles de conduite peuvent donc être représentés par des normes, les paramètres des styles se déclinant naturellement sur les paramètres des normes. Le moteur de génération permet alors de créer automatiquement et facilement des agents appartenant à la norme souhaitée, c'est-à-dire des conducteurs utilisant le style de conduite spécifié. Les possibilités d'analyse offertes par le modèle peuvent être exploitées en cas d'utilisation de paramètres dynamiques, ainsi que pour automatiser la configuration des styles depuis des données réelles.

4.2.2 Implémentation

L'utilisation du modèle de différenciation comportementale vise donc à introduire des styles de conduite dans la simulation. Nous décrivons ici les choix réalisés lors de l'implémentation du modèle au niveau de l'architecture d'une part, et au niveau du paramétrage des normes d'autre part.

4.2.2.1 Choix d'architecture

L'implémentation du modèle au sein de SCANeR™ a impliqué différents choix au niveau architecture. Pour la plupart, ces choix sont liés à des contraintes opérationnelles et au contexte de développement. Comme nous l'avons vu, SCANeR™ est une copropriété de Renault et Oktal (Sect. 4.1.1 p. 95). Les deux sociétés réalisent donc en parallèle des développements sur l'application, et des synchronisations régulières du code source ont lieu afin de mettre à jour leurs versions respectives.

Un critère de réussite important pour les différents acteurs de la thèse, et en particulier pour Renault et Oktal, était que les développements réalisés soient intégrés de manière pérenne dans SCANeR™. Pour répondre avec succès à ce besoin, il a été nécessaire de prendre en compte une difficulté particulière : Oktal débutait début 2008 un projet de refonte du module trafic, qui visait à moderniser l'architecture afin de gagner en maintenance et en évolutivité, sans apporter de changements fonctionnels majeur [25]. Réaliser les développements directement au sein du module trafic existant impliquait donc qu'il faudrait mener à la fin du projet un processus de fusion délicat, voire problématique, pour synchroniser les différentes versions du module trafic.

25. Ces travaux sont actuellement en cours et seront disponibles fin 2009 dans la version commerciale.

l_p		p_{ref}	$\mathcal{D}_p$	v_{d_p}	g_p	f_p
v_{max}	(maximal speed)	0	[0, 300] km/h	130 km/h	$\mu = 130,\ \sigma = 10$	default
t_s	(safety time)	0	[0, 10] s	2 s	$\mu = 2,\ \sigma = 1$	default
r_o	(overtakink risk)	0	$[-1, 2]$	0	$\mu = 0,\ \sigma = 1$	default
r_s	(speed limit risk)	0	[0, 2]	1	$\mu = 1,\ \sigma = 1$	default
o_p	(observe priority)	0	$\{true, false\}$	$true$	default	default
o_s	(observe signalization)	0	$\{true, false\}$	$true$	default	default

TABLE 4.2 – Définition des paramètres de la norme racine N_{init} : il s'agit donc des paramètres initiaux qui seront utilisés par les normes décrivant les styles de conduite.

Il existait également un problème de délais, la version refondue devant être opérationnelle avant la fin des travaux de thèse pour pouvoir réaliser la synchronisation.

Ces risques nous ont conduit à choisir une autre approche : intégrer le modèle dans des modules séparés s'interfaçant avec le trafic à travers la SCANeR™ API. En effet, la compatibilité ascendante est alors garantie, ce qui permettait de pérenniser les développements. La méthode rendait de plus possible un fonctionnement avec toutes les versions de l'application compatibles avec l'API, et une réintégration peu coûteuse dans la nouvelle version du trafic.

Cependant, ce choix impliquait également certaines limitations. Toute modification de fond sur le trafic devait être évitée, et il était donc nécessaire de n'utiliser que les fonctionnalités disponibles par l'intermédiaire de l'API. En particulier, seuls les paramètres existants du modèle de trafic, définis dans la Section 4.1.3 (p. 107), peuvent être utilisé . Ces paramètres permettent d'obtenir une grande variété de comportements, mais leur aspect statique, leur interdépendance, et le peu de maîtrise que nous avons de leur influence réelle sur le modèle sont des limitations dont il faut tenir compte.

4.2.2.2 Choix du paramétrage

SCANeR™ est commercialisé dans de nombreux pays, entre lesquels les styles de conduite peuvent varier de manière importante. Un comportement agressif en France ne sera en effet pas forcément perçu comme tel en Italie. Il est donc nécessaire de permettre aux utilisateurs de modifier les styles utilisés. Par ailleurs, certains scénarios nécessitent un contrôle fin du paramétrage des véhicules. Il faut donc permettre aux utilisateurs d'ajouter et de supprimer des styles, et donc des normes, en fonction de leurs besoins spécifiques.

Outre ce besoin d'ouverture, le paramétrage fourni par défaut doit satisfaire la majorité des besoins. En effet, les utilisateurs se reposent souvent sur lui, et il doit donc produire un rendu réaliste. D'après les résultats décrits dans [Wright et al., 2002], seul un nombre limité de types de conduite est distingué par les utilisateurs de simulateurs de conduite : agressif, normal et prudent. Un paramétrage basé sur des éléments de granularité plus fine comprenant le genre, l'âge, l'agressivité, l'intoxication ou la fatigue n'apporte en effet pas de changement à la manière dont les comportements observés sont perçus (cf. Sect. 1.3.4 p. 34). Le paramétrage par défaut fourni aux utilisateurs sera donc basé sur ces trois styles de conduite. Ceux-ci seront éditables, et des styles pourront être ajoutés ou supprimés facilement.

L'architecture choisie impose le choix des paramètres, qui doivent être ceux du modèle de trafic existant. L'influence de chacun d'entre eux n'étant pas précisément mesurable, nous avons procédé empiriquement à la sélection des domaines de définition, en utilisant des intervalles qui

l_p	p_{ref}	$\mathcal{D}_p$	v_{d_p}	g_p	f_p
$v_{max,prudent}$	v_{max}	[90, 110] km/h	100 km/h	$\mu = 100,\ \sigma = 10$	default
$t_{s,prudent}$	t_s	[1.5, 2.5] s	2.0 s	$\mu = 2.0,\ \sigma = 0.25$	default
$r_{o,prudent}$	r_o	$[-1.0,\ 0.0]$	-0.5	$\mu = -0.5,\ \sigma = 0.25$	default
$r_{s,prudent}$	r_s	[0.8, 1.0]	0.9	$\mu = 0.9,\ \sigma = 0.025$	default
$o_{p,prudent}$	o_p	$\{true\}$	$true$	default	default
$o_{s,prudent}$	o_s	$\{true\}$	$true$	default	default

TABLE 4.3 – Définition des paramètres de la norme N_{prudent}.

l_p	p_{ref}	$\mathcal{D}_p$	v_{d_p}	g_p	f_p
$v_{max,normal}$	v_{max}	[110, 130] km/h	120 km/h	$\mu = 120,\ \sigma = 10$	default
$t_{s,normal}$	t_s	[1.0, 2.0] s	1.5 s	$\mu = 1.5,\ \sigma = 0.25$	default
$r_{o,normal}$	r_o	[0.0, 1.0]	0.5	$\mu = 0.5,\ \sigma = 0.25$	default
$r_{s,normal}$	r_s	[0.9, 1.1]	1.0	$\mu = 1.0,\ \sigma = 0.025$	default
$o_{p,normal}$	o_p	$\{true\}$	$true$	default	default
$o_{s,normal}$	o_s	$\{true\}$	$true$	default	default

TABLE 4.4 – Définition des paramètres de la norme N_{normal}.

p_{ref}	p_{ref}	$\mathcal{D}_p$	v_{d_p}	g_p	f_p
$v_{max,agressif}$	v_{max}	[130, 150] km/h	140 km/h	$\mu = 140,\ \sigma = 10$	default
$t_{s,agressif}$	t_s	[0.5, 1.5] s	1.0 s	$\mu = 1.0,\ \sigma = 0.25$	default
$r_{o,agressif}$	r_o	[1.0, 2.0]	1.5	$\mu = 1.5,\ \sigma = 0.25$	default
$r_{s,agressif}$	r_s	[1.0, 1.2]	1.1	$\mu = 1.1,\ \sigma = 0.025$	default
$o_{p,agressif}$	o_p	$\{true,\ false\}$	$true$	default	default
$o_{s,agressif}$	o_s	$\{true,\ false\}$	$true$	default	default

TABLE 4.5 – Définition des paramètres de la norme N_{agressif}.

leur sont intuitivement associés dans le monde réel. Les paramètres racine sont définis en se basant sur les valeurs présentées dans la section 4.1.3 (p. 107), ce qui conduit aux définitions du tableau 4.2. Afin de ne pas autoriser de valeurs trop extrêmes, la vitesse maximale est limitée à 300 km/h, le temps de sécurité à 10 secondes et le « speed limit risk » à 2. Pour toutes les valeurs réelles la distribution des paramètres g_p est définie par des distributions normales de moyenne μ et de variance σ, tronquées aux limites de $\mathcal{D}_p$. Pour les paramètres discrets, on utilise la distribution par défaut, qui est uniforme. Enfin les fonctions f_p, qui décrivent l'écart à la norme, sont les fonctions par défaut. Elles valent 0 sur l'intervalle $\mathcal{D}_p$, puis 1 en dehors si le paramètre est discret, la distance au centre de $\mathcal{D}_p$ si le paramètre est réel.

La norme racine N_{init} est définie de la manière suivante :
- $l_N = N_{\text{init}}$
- $N_{\text{ref}_{N_{\text{init}}}} = 0$
- $\mathcal{P}_{N_{\text{init}}} = \{v_{max},\ t_s,\ r_o,\ r_s,\ o_p,\ o_s\}$
- $\mathcal{Q}_{N_{\text{init}}} = \{France\}$

On définit ensuite trois normes correspondant aux trois styles de conduite que l'on souhaite obtenir : N_{prudent}, N_{normal} et N_{agressif}. Les paramétrages correspondant à ces normes sont présentés dans les tableaux 4.3, 4.4 et 4.5. Pour ces trois normes, le taux de violation est nul et l'écart maximal à la norme est fixé à 1 ($\tau_{N_i} = 0$ et $\delta_{\max_{N_i}} = 1$). Par ailleurs, on considère

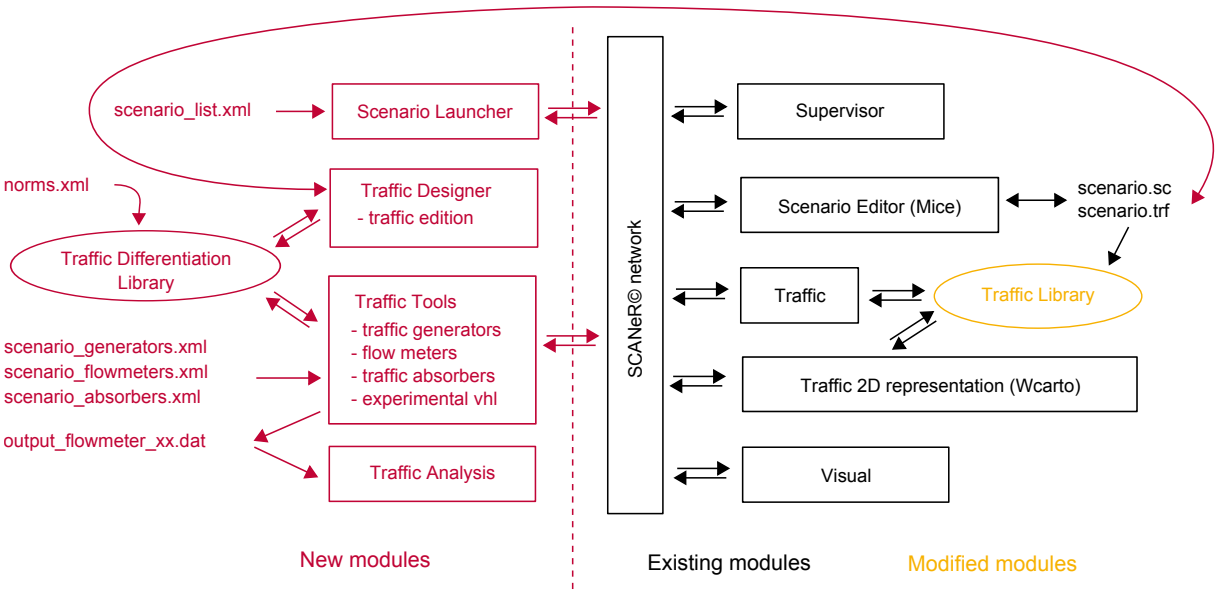


FIGURE 4.13 – Intégration des modules développés dans l'architecture existante.

que les paramètres définis correspondent à une conduite en France sur autoroute, critères qui constituent les propriétés des normes ($\mathcal{Q}_{N_i} = \{France,\ autoroute\}$).

Notons que le Chapitre 5 présente une évaluation des choix réalisés, incluant en particulier une comparaison de différents ensembles de normes et des paramétrages associés. La calibration reste empirique, mais nous verrons le processus qui a permis d'affiner les valeurs utilisées et de fournir un paramétrage répondant aux besoins.

Enfin, l'institution est définie par :
- $l_I = I_{trafic}$
- $\mathcal{N}_I = \{N_{\text{init}},\ N_{\text{prudent}},\ N_{\text{normal}},\ N_{\text{agressif}}\}$
- $\sigma = 0$

4.2.3 Intégration du modèle dans SCANeR™

Les choix réalisés ont conduit à la définition de l'architecture que nous présentons dans cette section. Nous décrivons ensuite les modules logiciels développés au cours de la thèse.

4.2.3.1 Architecture

Les développements réalisés au cours de la thèse ont été intégrés dans des modules distincts, interfacés à SCANeR™ par l'intermédiaire de la SCANeR™ API. Trois modules appliquant le modèle ont été introduits (Fig. 4.13) :

1. Le module « Traffic Designer » est destiné à éditer des scénarios pré-existants. Il permet d'introduire les apports du modèle lors de la phase de préparation des simulations.
2. Le module « Traffic Tools » est destiné à créer dynamiquement des véhicules en cours de simulation. Il permet de simplifier la préparation des scénarios en évitant d'avoir à construire manuellement la partie « trafic ambiant ». Le modèle est appliqué lors de la création des véhicules.

3. Enfin le module « Traffic Analysis » est un outil d'analyse, qui permet d'exploiter à posteriori les résultats des simulations afin de mieux comprendre leur évolution.

Enfin, un module complémentaire, « Scenario Launcher », est destiné à l'automatisation des tests. Il permet de lancer en temps différé des séries de scénarios préprogrammées. Celles-ci sont stockées dans un fichier xml `scenario_list.xml`, qui contient la liste des scénarios à lancer ainsi que la durée des simulations. Ce module a été utilisé pour réaliser les évaluations qui sont présentées dans le Chapitre 5.

Le modèle de différenciation comportementale a été implémenté au sein d'une bibliothèque C++, qui est ensuite utilisée par les modules « Traffic Designer » et « Traffic Tools ». Les normes sont définies dans des fichiers de configuration xml chargés dynamiquement au démarrage de la simulation. Elles peuvent être différentes pour chacun des scénarios, et il est possible d'utiliser différents jeux de normes au sein d'un même scénario. Le fichier utilisé possède la structure suivante (Listing. 4.1) : chaque section est dédiée à une norme particulière, définissant un style de conduite, et lui associe la définition des paramètres.

```
1  <norms>
2      <norm id="AggressiveDriver">
3          <label>Aggressive</label>
4          <parameter id="MaximalSpeed" type="TruncatedNormalDistribution">
5              <mean_value>140</mean_value>
6              <standart_deviation>5</standart_deviation>
7              <minimal_value>130</minimal_value>
8              <maximal_value>150</maximal_value>
9          </parameter>
10         <parameter id="SafetyTime" type="UniformDistribution">
11             <minimal_value>0.4</minimal_value>
12             <maximal_value>1.2</maximal_value>
13         </parameter>
14     </norm>
15 </norms>
```

Listing 4.1 – Exemple de fichier xml de définition de normes. Ici une seule norme est définie, décrivant un style de conduite agressif

Une interface de modification de la configuration des normes est en cours de développement pour la prochaine version du logiciel, mais n'est pas intégrée dans la version prototype développée au cours de la thèse. Le format xml permet cependant déjà aux utilisateurs d'effectuer aisément les modifications qu'ils souhaitent.

Notons que la norme N_{init} est intégrée directement dans l'application. En effet, on ne souhaite pas la rendre disponible en modification aux utilisateurs, l'utilisation de paramètres extrêmes risquant de compromettre le fonctionnement global.

4.2.3.2 Édition de scénario

Le premier module développé, « Traffic Designer », est destiné à l'édition de scénarios existants. Dans SCANeR™, un scénario est défini par un fichier `nom_du_scenario.sc` qui contient les règles de scénarisation, et un fichier `nom_du_scenario.trf` qui contient la définition des conditions initiales de trafic : identifiant, position, nom et paramétrage des véhicules. Le module

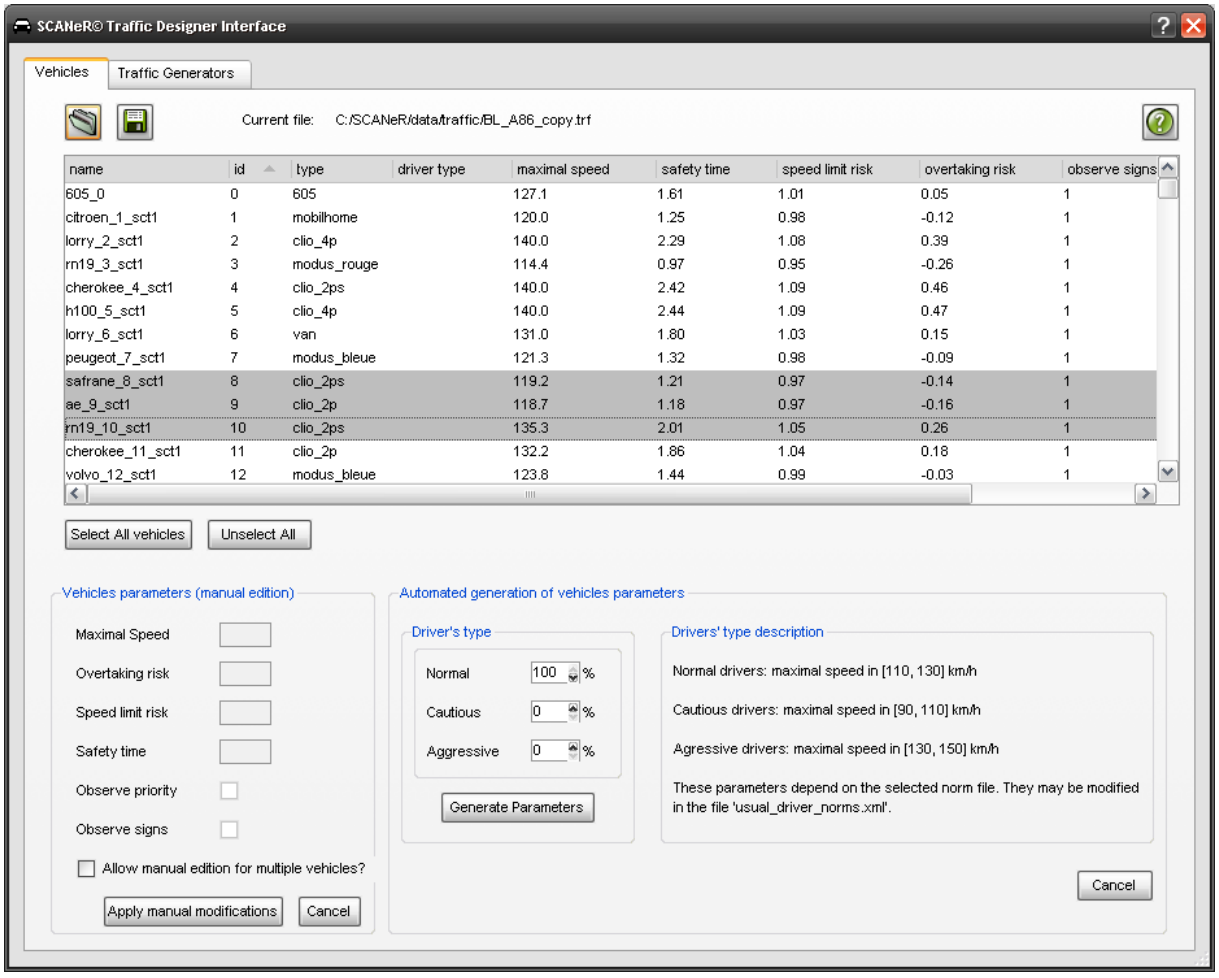


FIGURE 4.14 – Le module « Traffic Designer », permettant d'appliquer le modèle sur des scénarios existants et d'y introduire facilement de la différenciation comportementale.

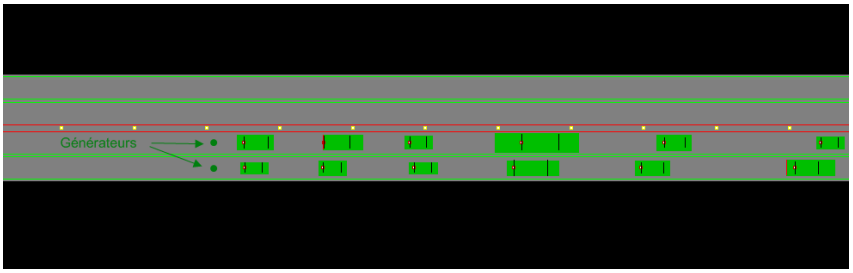


FIGURE 4.15 – Utilisation des générateurs de trafic pour produire un flux de véhicules dans la simulation.

charge ce fichier `.trf` et permet d'éditer les paramètres des véhicules. Son interface est présentée dans la Figure 4.14, et offre différentes fonctionnalités.

Tout d'abord, il est possible d'éditer manuellement les paramètres d'un véhicule ou d'un groupe de véhicules. Cette modification multiple n'était pas disponible dans l'éditeur de scénario. Ensuite, il est possible d'appliquer des normes à un groupe de véhicules suivant une répartition statistique définie par l'utilisateur. Il suffit pour cela de sélectionner les véhicules désirés, de compléter le pourcentage souhaité pour chacune des normes, puis de générer le paramétrage. Les données sont ensuite sauvegardées dans le fichier décrivant le trafic, et le scénario peut être lancé immédiatement.

Une utilisation simple de ce module est par exemple de sélectionner l'ensemble des véhicules, et de leur appliquer la norme « conducteur normal » en saisissant 100 % pour le profil correspondant. De cette façon, on introduit facilement de la différenciation comportementale sans remettre en cause la scénarisation préparée. Il faut cependant prendre soin de ne pas modifier les véhicules dont les paramètres jouent un rôle dans le scénario. Par exemple, dans certains scénarios la présence d'un véhicule lent est nécessaire à la situation de conduite que l'on souhaite produire. Il est alors important de ne pas modifier le paramétrage de ce véhicule.

Ce module permet donc d'appliquer facilement les apports du modèle, sans avoir à modifier le scénario en profondeur. À terme, il sera intégré directement au sein de l'éditeur de scénario, afin de rendre ses fonctionnalités disponibles aisément à tous les utilisateurs.

4.2.3.3 Génération de trafic

Le module « Traffic Tools » est destiné à créer dynamiquement des véhicules durant l'exécution de la simulation. Il fournit également des fonctions complémentaires : suppression automatisée de véhicules, enregistrement de données trafic et visualisation de statistiques en temps réel.

Générateurs de trafic La génération de trafic se fait par l'intermédiaire de « sources » de véhicules, positionnées par l'utilisateur sur le réseau routier (Fig. 4.15). Ces sources créent de nouveaux véhicules suivant la configuration fournie. Cette configuration est définie dans un ensemble de fichiers xml permettant de décrire l'ensemble de leurs caractéristiques. Tout d'abord, l'utilisateur définit l'ensemble des générateurs de trafic, avec leurs positions, identifiants, labels, et l'ensemble de normes qui sera utilisé (Fig. 4.16). Il est possible d'utiliser des normes différentes pour chacun des générateurs.

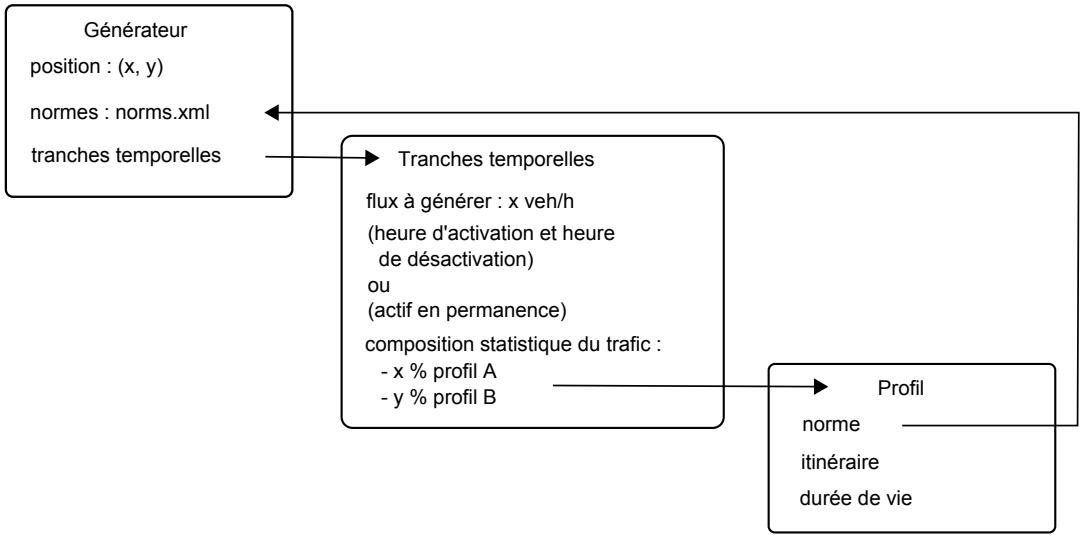


FIGURE 4.16 – Fonctionnement des générateurs de trafic.

Chaque générateur est associé à une ou plusieurs « tranches temporelles » de trafic. Ces tranches temporelles permettent de définir une demande de trafic variable au cours du temps. Pour chacune d'elle, on spécifie le flux de véhicules qui va être créé par le générateur (en nombre de véhicules par heure), les heures de début et de fin d'activité de la tranche (en secondes), ainsi que la composition statistique du trafic créé. La composition du trafic définit les profils de conducteurs qui seront utilisés, ainsi que la proportion de chaque profil. Notons qu'une option permet de définir des tranches actives en permanence pendant la simulation.

Enfin ces profils sont mis en correspondance avec une norme provenant de l'ensemble défini au niveau du générateur. Un itinéraire et une durée de vie peuvent leur être associé, de manière facultative. L'itinéraire permet de créer des demandes origine / destination, tout en offrant une granularité de configuration par type de conducteur. La durée de vie permet de désactiver un véhicule après une durée donnée, afin de limiter les risques de saturation des ressources systèmes.

Les générateurs de trafic permettent ainsi de peupler aisément un environnement avec un trafic ambiant, en limitant les actions du concepteur à la configuration initiale des générateurs. Le modèle de différenciation comportementale est appliqué au moment de la création des véhicules, ce qui permet de produire un trafic varié. Enfin les options avancées offrent la possibilité d'utiliser ces générateurs pour réaliser des expérimentations orientées trafic, en utilisant des matrices origine / destination et des demandes de trafic variables. La Figure 4.18 présente l'onglet du module permettant la visualisation de la configuration des générateurs.

Suppression automatique de véhicules Le module permet également de définir des « puits » de véhicules, destinés à supprimer automatiquement les véhicules autonomes. En pratique ces puits sont des segments définis par l'utilisateur sur le réseau routier. Lorsqu'un véhicule autonome traverse l'un d'eux, il est automatiquement supprimé. Les puits permettent de définir aisément des points de sortie dans le réseau, et de limiter ainsi les surcharges système causées par un trop grand nombre de véhicules présents. Ces puits peuvent être définis par route, par sens de circulation, ou par voie.

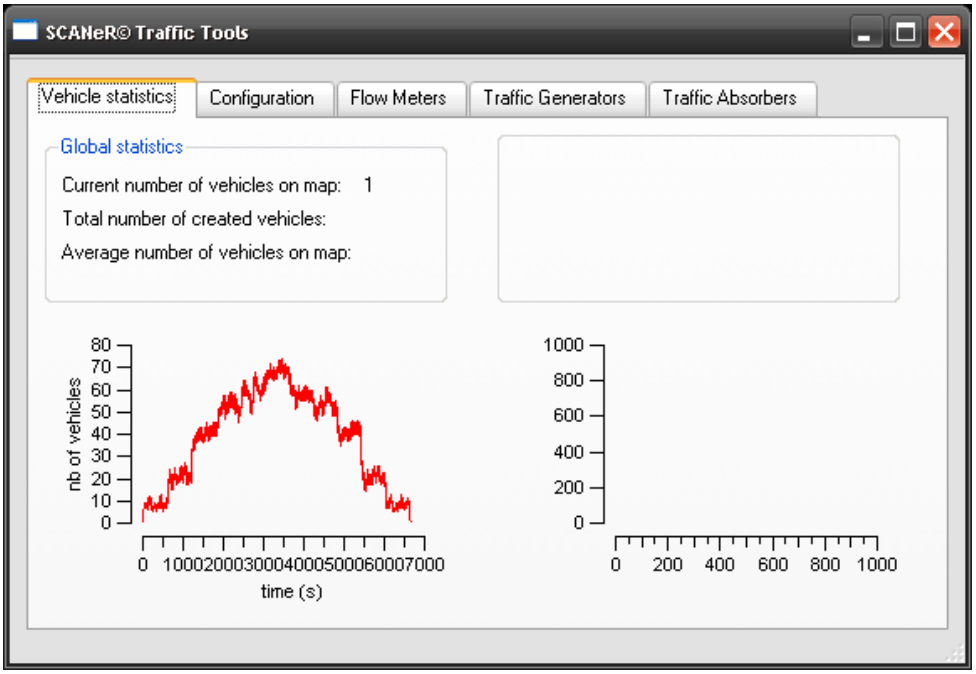


FIGURE 4.17 – Le module « Traffic Tools » permet de visualiser en temps réels différentes données statistiques sur le trafic.

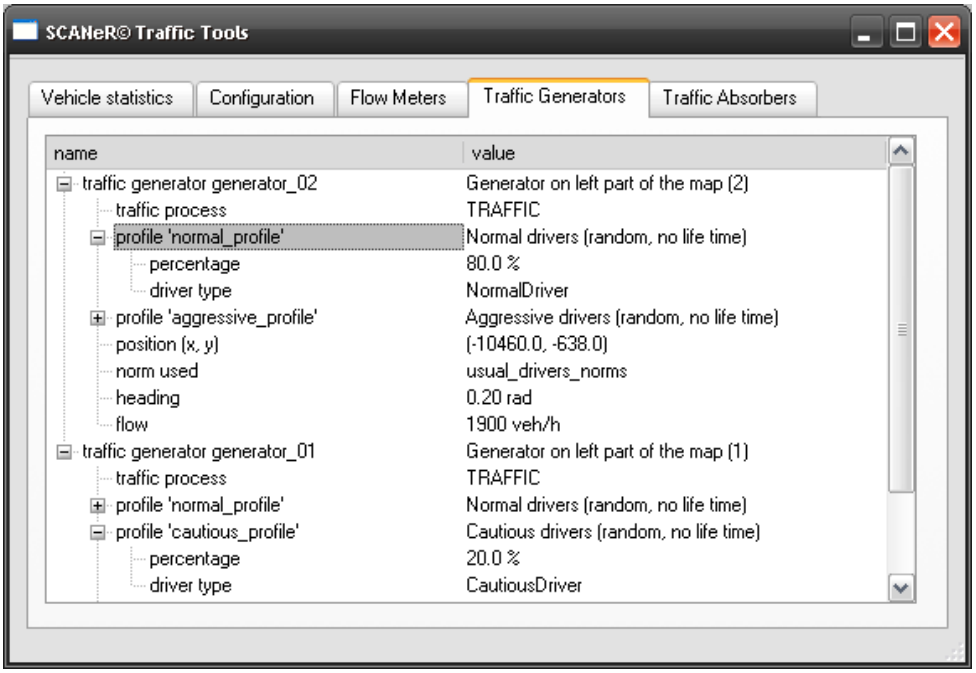


FIGURE 4.18 – Les informations relatives aux générateurs de trafic dans le module « Traffic Tools ».

Mesure de flux Sur le même principe, des flux mètres peuvent être positionnés dans le réseau routier. Définis sous forme de segments de droite, ils enregistrent les données concernant les véhicules qui les traversent. Ces données sont stockées au format texte dans un fichier `nom_du_scenario.timestamp.dat`, qui est enregistré dans le répertoire de sortie de SCANeR™. L'exploitation de ces données permet ensuite de construire des statistiques détaillées sur le trafic.

Statistiques Enfin différentes statistiques sont fournies aux utilisateurs en temps réel. Par exemple, le nombre de véhicules actuellement présents sur le réseau, ou le nombre de véhicules créés durant la simulation sont affichés directement dans l'interface. Ils permettent d'évaluer d'éventuels dysfonctionnements liées à la saturation des ressources systèmes, et fournissent à l'utilisateur un retour sur le bon fonctionnement du module. La Figure 4.17, qui représente l'onglet de visualisation des statistiques, illustre cette fonctionnalité.

Violations Notons que l'utilisation de ce module pour la conception de scénarios nécessitait une maîtrise complète du processus par les utilisateurs. Pour cette raison, il a été décidé de ne pas rendre disponible l'usage des violations, qui peuvent perturber la simulation si elles sont utilisées de manière inadéquate. Par ailleurs, par construction les normes ne définissent que des paramètres statiques de la simulation : les paramétrages générés sont tous « conformes », au sens décrit dans la Section 2.3.2 (p. 66). Le mécanisme de contrôle de la conformité n'est donc pas utilisé dans cette version, bien qu'il reste disponible. Il trouvera son application dès que des violations seront autorisées.

Enfin, l'ensemble des fonctionnalités de ce module sont d'ores et déjà intégrées à la prochaine version commerciale de SCANeR™, prévue pour septembre 2009. En particulier, l'ajout de sources et puits de véhicules sont présents dans l'éditeur de scénario, et le module « Traffic Tools » est utilisé afin de fournir les fonctionnalités associées.

4.2.3.4 Analyse de la simulation

Le module « Traffic Analysis » est destiné à exploiter les données trafic en temps différé (Fig. 4.19). Il permet d'utiliser les données enregistrées par les flux mètres afin d'en inférer les styles de conduite utilisés par les véhicules autonomes et les normes associées. Son fonctionnement est basé sur les méthodes de classification présentées dans le Chapitre 3. L'utilisateur peut spécifier le fichier qu'il souhaite analyser et modifier manuellement les paramètres du réseau de Kohonen.

Ce module est encore à l'état de prototype, et des problèmes de performance limitent son utilisation. Cependant, dès que les capacités d'automatisation de la configuration des normes donneront des résultats exploitables facilement, il sera intégré dans la version d'exploitation de SCANeR™. Nous verrons dans le Chapitre 5 les résultats qu'il permet déjà d'obtenir.

4.2.4 Synthèse

Dans le cadre de l'application du modèle à SCANeR™, les objectifs étaient d'une part de faciliter le travail des concepteurs de scénario, et d'autre part d'améliorer l'immersion des utilisateurs. Le module « Traffic Designer » permet de modifier facilement les scénarios existant pour leur ajouter de la diversité. Le module « Traffic Tools » permet de créer un trafic ambiant

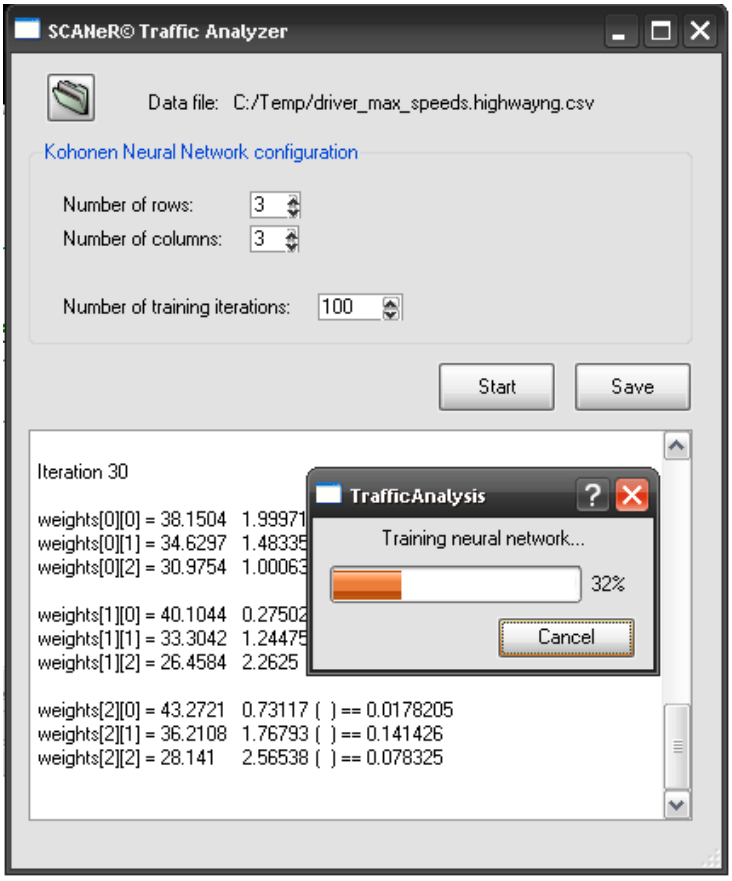


FIGURE 4.19 – Le module « Traffic Monitoring » destiné à la classification des données et à l'inférence de normes.

et de peupler les environnements de conduite. Ils sont configurables facilement, et les configurations sont réutilisables entre les scénarios. Tous deux utilisent le modèle de différentiation comportementale, permettant d'obtenir un paramétrage varié et conforme aux spécifications.

Par ailleurs, le choix de développer des modules distincts basés sur la SCANeR™ API s'est avéré payant. En regard des modifications profondes réalisées lors de la refonte du trafic, le coût de réintégration aurait imposé un recodage complet des développements réalisés pendant la thèse. Adopter une architecture modulaire a permis d'intégrer une partie des développements dans la version officielle de l'application, ce qui n'aurait pu être le cas autrement. Enfin, l'API garantit une compatibilité ascendante assurant la pérennité des réalisations.

L'introduction du modèle permet de confronter les utilisateurs à de nombreux comportements, y compris non spécifiés. Dans le cas de tests de systèmes embarqués, cette variété de situations est intéressante d'un point de vue expérimental, car elle permet d'élargir le domaine de test du système. S'il ne reste bien sûr pas possible de garantir le fonctionnement sur l'ensemble du domaine, la totalité des cas n'ayant pas été testée, cela reste une amélioration importante par rapport à la situation antérieure.

Au vu des besoins utilisateurs, les violations n'ont pour l'heure pas été exploitées dans les modules. Dans le cas d'expérimentations en simulation de conduite, où le contexte nécessite le plus souvent d'être totalement maîtrisé, elles entraînent une dimension non-déterministe qui n'est le plus souvent pas souhaitable.

Pour finir, notons que la généricité du modèle permet de l'appliquer facilement aux autres agents de la simulation, et en particulier aux piétons. Il suffit en effet de définir les normes adéquates, ne correspondant plus à des styles de conduite mais à des styles de « marcheurs », afin d'utiliser directement les modules pour cet usage.

4.3 Conclusion

SCANeR™ est une suite logicielle dédiée à la simulation de conduite. Développée depuis une quinzaine d'année, elle est utilisée dans des domaines variés, de l'ingénierie à l'étude des facteurs humains. Elle repose sur une architecture distribuée, et est destinée à être déployée sur un réseau de machine afin de prendre en compte les architectures matérielles souvent complexes. L'architecture est ouverte, à travers une API permettant d'interfacer facilement de nouveaux modules avec l'application. SCANeR™utilise différents modules pour la partie simulation du trafic routier, qui permettent d'animer le trafic autonome et de scénariser des situations de conduite pour les expérimentations. Le module de trafic repose sur une architecture multi-agents. Chaque véhicule est représenté par un agent, qui utilise un modèle de décision « perception – décision – action ». La phase de perception leur permet de mettre à jour leur représentation de l'environnement, la phase de décision de choisir leur manœuvre, et la phase d'action de calculer leur prochaine position. Ce modèle prend en compte les paramètres physiques du véhicule, ainsi que des paramètres pseudo-psychologiques relatifs au conducteur. Enfin, notons que les piétons peuvent soit intervenir dans la simulation, soit être considérés comme des piétons d'« ambiance » utilisant un modèle spécifique.

Le modèle de différenciation comportementale a été appliqué à SCANeR™ afin de résoudre deux problématiques principales : améliorer le réalisme comportemental et faciliter la conception des scénarios. Cela nécessitait en particulier de fournir un paramétrage par défaut réaliste aux véhicules, et d'automatiser leur création. Afin d'assurer la pérennité des développements,

ceux-ci ont été intégrés en tant que modules distincts du module trafic, en se basant sur la SCANeR™ API. Ce choix imposait cependant de n'utiliser que le paramétrage existant des véhicules. L'application du modèle repose sur l'utilisation des normes pour définir des styles de conduite pour les conducteurs. Trois modules principaux ont été développés afin de les intégrer dans SCANeR™. Le premier permet de modifier les scénarios existants, afin de leur appliquer les styles de conduite sans remise en cause de l'existant. Le second module permet de créer dynamiquement des véhicules pendant l'exécution de la simulation, ce qui permet d'insérer aisément un trafic ambiant. Enfin le troisième module permet d'analyser les simulations. L'ensemble de ces réalisations a apporté à SCANeR™ les avantages du modèle proposé, et permet de grandement faciliter certaines tâches des scénaristes. Elles sont déjà en parties incluses dans la version commerciale de l'application.

Chapitre 5

Évaluation, discussion et perspectives

Dans ce chapitre, nous abordons les différentes expérimentations que nous avons menées pour l'évaluation des apports de nos travaux. Nous présentons tout d'abord les démarches de validation utilisées dans le cadre de la simulation de trafic, afin de motiver les choix réalisés. Dans un second temps, nous nous intéressons à l'évaluation de l'apport des normes dans SCANeR™. Nous présentons les expérimentations menées pour la calibration du paramétrage, ainsi que les résultats de simulations qui illustrent les apports du modèle. Nous abordons ensuite l'évaluation de la simulation de trafic dans SCANeR™, en présentant les cas d'utilisation étudiés. Enfin, nous discutons démarche et résultats du point de vue du modèle d'une part, et du point de vue de l'application d'autre part. Nous terminons en présentant les perspectives de nos travaux.

5.1 Démarche de validation

Dans cette première section, nous introduisons les différentes démarches de validation qui peuvent être envisagées dans le contexte du trafic routier. Tout d'abord, les outils utilisés nécessitent une calibration, dont nous présentons le principe. Nous abordons ensuite les questions relatives à la collecte des données, qui permettent d'obtenir une référence pour la simulation, puis nous nous intéressons enfin aux méthodes de validation du trafic dans les simulateurs de conduite.

5.1.1 Calibration des modèles

Lorsque l'objectif de la simulation est de reproduire des situations réelles, la calibration des modèles de trafic est nécessaire. Il faut en effet être en mesure de reproduire la situation existante de manière précise, afin de pouvoir évaluer l'effet des modifications qui seront apportées sur le réseau routier ou le comportement des véhicules. En ingénierie du trafic, cette problématique est importante et souvent abordée de manière détaillée [Barcelò et al., 1999, Toledo et al., 2003, Ciuffo et al., 2008].

D'une manière générale, il s'agit de déterminer les valeurs des paramètres du modèle, comme ceux du suivi de véhicule ou du changement de ligne, qui permettent de reproduire les données

réelles [Hourdakis et al., 2003]. La calibration peut par exemple se faire en plusieurs étapes « essais – erreurs » :

1. La première étape se base sur une estimation des paramètres du modèle, à partir des données de la littérature. Les résultats simulés sont ensuite comparés aux mesures réelles de volumes de trafic, et le paramétrage est affiné jusqu'à ce que la reproduction soit satisfaisante.
2. La seconde phase de calibration s'intéresse à la reproduction des vitesses. De la même façon, des itérations successives de la simulation permettent de converger vers un paramétrage adéquat.
3. Enfin, la troisième phase s'intéresse à la reproduction d'un critère permettant de qualifier l'objectif de l'étude. Sur des études de stratégies de signalisation ce critère peut être la longueur de la file d'attente sur la voie d'insertion [Hourdakis & Michalopoulos, 2002].

L'écart entre les données réelles et les données simulées est mesuré par différentes techniques. Le RMS (Root Mean Square Error) et le RMSP (Root Mean Square Percentage Error) sont des mesures classiques [Barcelò & Casas, 2002] :

$$RMS_i = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(w_{ij} - v_{ij})^2} \qquad RMSP_i = \sqrt{\frac{1}{m}\sum_{j=1}^{m}\left(\frac{w_{ij} - v_{ij}}{v_{ij}}\right)^2}$$

où les v_{ij} et les w_{ij} sont les séries temporelles à comparer. Ces mesures présentent cependant le défaut d'amplifier fortement les erreurs élevées. La confiance dans la calibration réalisée est finalement établie en simulant une situation qui n'a pas été utilisée pendant la calibration, et en vérifiant que les données obtenues sont proches des données réelles.

Un autre aspect de la calibration consiste à vérifier que les sous-modèles de trafic produisent bien des résultats conformes à la réalité [Fellendorf & Vortisch, 2001, Punzo & Simonelli, 2005]. Enfin, notons que la complexité du processus de calibration a conduit au développement de « frameworks » dédiés, qui ont pour objectif de définir les étapes nécessaires et suffisantes pour la calibration et proposent des méthodologies permettant une automatisation maximale du processus [Ciuffo et al., 2007, Hourdakis et al., 2003].

5.1.2 Collecte de données

La calibration des modèles repose sur l'utilisation de données réelles, qui peuvent être collectées par différentes méthodes [Hidas & Wagner, 2004, Hughes, 1998]. Nous présentons ici quatre d'entre elles : les détecteurs à boucle d'induction, l'analyse de données vidéo, l'utilisation de GPS et l'utilisation de véhicules équipés de systèmes d'enregistrement de la conduite.

Les détecteurs à boucle d'induction sont constitués d'une boucle à induction fixée sur la chaussée, qui repère la déformation du champ magnétique induite par le passage d'un véhicule. Ils permettent de détecter le passage et la présence d'un véhicule au dessus de la boucle, et d'en déduire l'occupation du réseau. Pour mesurer la vitesse, il est nécessaire d'utiliser deux boucles, ou de l'extrapoler à partir des données disponibles et de valeurs moyennes. Ces systèmes sont largement utilisés sur le réseau routier, du fait de leur faible coût et de leur fiabilité élevée. Cependant, ils ne permettent d'obtenir que des données ponctuelles sur les véhicules.

Des données vidéos peuvent être utilisées pour collecter des données temporelles sur les véhicules traversant une portion de route. Cette méthode présente l'avantage de ne pas influencer

le comportement des conducteurs, qui n'ont pas conscience d'être observés. De plus, elle permet de collecter des données sur les comportements de suivi de véhicule et de changement de ligne, difficilement quantifiables avec d'autres approches. Cependant, seule une portion de route de taille limitée peut être observée, d'une longueur de 150 à 200 mètres, et l'analyse des données est complexe et peu automatisée.

L'utilisation de GPS permet de collecter des données sur les caractéristiques individuelles des conducteurs. En équipant deux véhicules de GPS différentiels et en les faisant interagir, il est également possible de collecter des données sur des situations de conduite contrôlées. Les données sont disponibles sur de longues périodes de temps, ce qui permet d'inclure différents types de route et conditions de trafic, et de réaliser une analyse poussée du comportement du conducteur. L'analyse des données est par ailleurs plus aisée que dans le cas des enregistrements vidéos, la vitesse et l'accélération pouvant être directement calculées depuis les données GPS. Cependant, seuls quelques véhicules peuvent être équipés, et le fait que les conducteurs se savent observés introduit des biais dans leur comportement.

Enfin l'utilisation de véhicules équipés de systèmes d'enregistrement dédiés fournit d'importantes quantités de données qui peuvent être analysées de manière fortement automatisée. Ces systèmes sont constitués de radars, de caméras vidéos enregistrant la vue à l'avant et à l'arrière du véhicule, et d'un GPS différentiel. Les radars permettent d'enregistrer la position relative et la vitesse des objets devant et derrière le véhicule, mais les informations obtenues ne sont pertinentes que pour certaines situations de conduite. Par ailleurs, le même biais d'observation que dans le cas du GPS est introduit, car les conducteurs savent qu'ils conduisent un véhicule équipé. Enfin, il est souvent difficile d'observer les véhicules environnants suffisamment longtemps pour être en mesure de qualifier leur comportement.

Chaque méthode possède donc ses avantages et ses inconvénients, et peut fournir certaines données pour la calibration et la validation des modèles. Il n'existe pas aujourd'hui d'ensemble idéal de données, contenant de manière continue à la fois des profils d'accélération et de vitesse de grands ensembles de véhicules évoluant dans divers environnements sur de longues distances et sous différentes conditions de trafic [Hidas & Wagner, 2004]. Un projet initié par la Federal Highway Administration, NGSIM [FHWA, 2009], vise à construire un ensemble de données permettant de fournir une référence répondant à ces critères et disponible pour toutes les applications de simulation de trafic.

5.1.3 Validation du trafic dans les simulateurs de conduite

Dans le contexte des simulateurs de conduite, la validation du trafic doit prendre en compte le niveau macroscopique, mais également s'intéresser de manière détaillée au niveau microscopique.

La validation macroscopique correspond à la calibration réalisée dans le cadre de la simulation de trafic en ingénierie du trafic. Il s'agit de vérifier que le modèle de trafic est capable de reproduire les lois d'écoulement du trafic. Les flux réels et simulés peuvent être comparés, afin de valider le modèle de comportement, par exemple dans le cas des insertions [Hadouaj et al., 2004]. De la même façon, en mesurant les flux sur autoroute puis en les comparant aux données simulées, il est possible de qualifier la qualité de reproduction globale du trafic [Champion et al., 2002].

Cependant, dans le cas de la simulation de conduite, l'objectif principal est le réalisme au niveau microscopique. En effet, il est important que le comportement des véhicules soit perçu comme réaliste. Dans l'idéal, l'objectif serait même d'obtenir un modèle pour lequel le conducteur réel ne serait pas en mesure de conclure si le véhicule à son côté est conduit par un humain ou

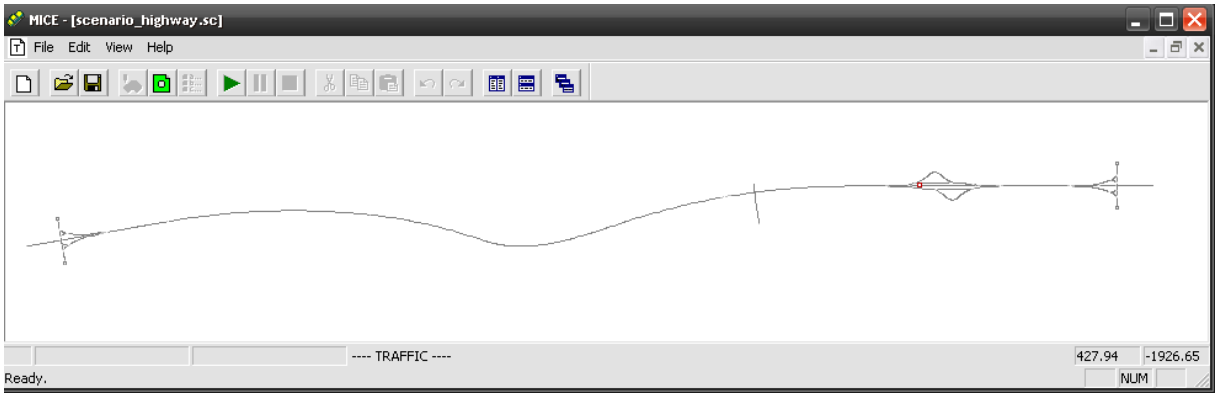


FIGURE 5.1 – La base de données de type autoroute utilisée pour les expérimentations.

par un conducteur virtuel. Les objectifs secondaires sont d'une part que les sorties du modèle microscopique correspondent à celles d'un conducteur réel, et d'autre part que celles du niveau macroscopique correspondent avec les mesures du trafic réel.

La seule méthode utilisée aujourd'hui pour valider le réalisme du comportement des véhicules ambiant est l'utilisation d'observateurs humains, en situation de conduite ou non [Olstam, 2002, Wright et al., 2002]. À l'aide de questionnaires, il est possible d'obtenir une qualification du comportement des véhicules du trafic. Cette méthode est cependant éminemment subjective, le jugement des conducteurs étant très relatif à la fois sur le réalisme des comportements et sur leur classification. Pour pallier en partie ce problème, il est possible de faire appel à des experts plutôt qu'à des conducteurs normaux. Par exemple, des psychologues de la conduite peuvent être sollicités pour observer le trafic simulé, afin de qualifier la qualité des comportements et le réalisme de la simulation [Hadouaj et al., 2000]. Notons finalement qu'il est difficile de définir à quel point le comportement doit être réaliste pour que le modèle soit valide.

5.2 Évaluation de l'apport des normes dans scaner™

Afin de valider les apports du modèle, la confrontation d'utilisateurs avec des scénarios intégrant les éléments introduits et d'autres n'en bénéficiant pas nous a paru trop subjective. Nous avons donc cherché à quantifier les apports du modèle en réalisant des mesures sur le trafic. La présentation des méthodes utilisées et des résultats obtenus fait l'objet de cette partie. Nous présentons tout d'abord les expérimentations ayant permis la calibration du paramétrage des normes, avant d'aborder les résultats issus de simulations qui permettent d'illustrer les apports du modèle.

5.2.1 Description des simulations

Nous présentons ici le contexte des simulations réalisées, ainsi que les normes choisies pour les expérimentations.

Description du contexte Les simulations présentées dans cette section ont été réalisées sur une base de données de type autoroute à 2 fois 2 voies, d'une longueur totale de 16 km (Fig. 5.1).

p_{ref}	$\mathcal{D}_p$	v_{d_p}	g_p
v_{max}	{130} km/h	130 km/h	default
t_s	{2} s	2 s	default
r_o	{0}	0	default
r_s	{1}	1	default
o_p	$\{true\}$	$true$	default
o_s	$\{true\}$	$true$	default

TABLE 5.1 – Paramètres utilisés pour la norme du jeu *no norms*.

p_{ref}	$\mathcal{D}_p$	v_{d_p}	g_p
v_{max}	[100, 140] km/h	120 km/h	$\mu = 120,\ \sigma = 10$
t_s	[1.0, 2.0] s	1.5 s	$\mu = 1.5,\ \sigma = 0.25$
r_o	[0.0, 1.0]	0.5	$\mu = 0.5,\ \sigma = 0.25$
r_s	[0.9, 1.1]	1.0	$\mu = 1.0,\ \sigma = 0.025$
o_p	$\{true\}$	$true$	default
o_s	$\{true\}$	$true$	default

TABLE 5.2 – Paramètres utilisés pour la norme du jeu *normal only*.

Une section de 11 km est utilisée pour les expérimentations, pour des questions de performances. Cette section ne comporte pas de voie d'insertion ni de sortie, afin de préserver un flux constant de véhicules sur le parcours. Trois détecteurs sont placés aux kilomètres 2.2, 6 et 10.8 par rapport au début de la section utilisée (détecteurs 1, 2 et 3). Ils permettent en particulier d'enregistrer la vitesse des véhicules, la voie sur laquelle ils se trouvent et leur norme initiale. Les mesures ne sont effectuées que dans un sens de circulation.

Des générateurs de trafic sont positionnés au début de la portion, et un « puits » est positionné à la fin de celle-ci afin de supprimer les véhicules arrivant au bout du parcours. Les générateurs créent une demande de trafic de 3000 véhicules par heure au début de la section, ce qui correspond à un flux dense. Après la création initiale des véhicules, le modèle de trafic de SCANeR™ les prend en charge tout au long de la simulation. Chaque simulation a une durée de 2h30. Les données présentées dans les graphes de la Section 5.2.2 sont les valeurs moyennées de 5 simulations distinctes.

Choix des normes Afin d'évaluer l'apport des normes dans la simulation, trois jeux de normes ont été utilisés pour les différentes simulations. Le premier jeu, *no norms*, est une norme restrictive : tous les paramètres sont réduits à un singleton, les violations étant interdites. Procéder de cette manière revient à désactiver le modèle de différenciation comportementale : tous les véhicules créés ont le même paramétrage. Les valeurs des paramètres sont les valeurs par défaut assignées par l'éditeur de scénario, présentées dans la Section 4.1.3 (p. 107) et rappelées dans le Tableau 5.1. De cette façon, la simulation correspond à un scénario dans lequel le concepteur aurait simplement ajouté les véhicules, sans prendre le temps de modifier manuellement leur paramétrage. Il s'agit de la condition de référence pour notre évaluation : cette configuration est celle que l'on retrouve dans la plupart des scénarios créés par les utilisateurs.

Dans le second jeu de norme, *normal only*, une seule norme est utilisée. Elle correspond à la norme N_{normal} présentée dans la Section 4.2.2 (p. 112), pour laquelle on étend le domaine de définition de la vitesse maximale de 100 à 140 km/h au lieu de se restreindre à l'intervalle

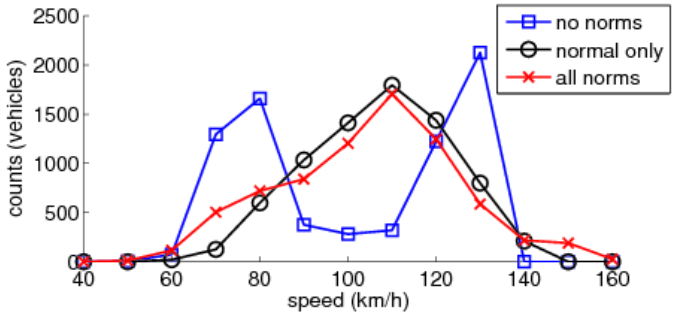


FIGURE 5.2 – Distribution des vitesses en fonction de l'ensemble de normes choisies.

[110, 130]. Les paramètres de la normes sont rappelés dans le Tableau 5.2. Lors de l'utilisation de ce jeu de normes, 100 % des véhicules sont générés avec la norme définie.

Enfin, le troisième jeu de normes, *all norms*, est basé sur les trois normes définies dans la Section 4.2.2 (p. 112) : N_{prudent}, N_{normal} et N_{agressif}. Lors de la simulation, les véhicules sont créés par les générateurs de trafic suivant les proportions suivantes : 10 % de conducteurs prudents, 80 % de normaux et 10 % d'agressifs.

5.2.2 Apports du modèle

Nous décrivons ici les résultats des expérimentations réalisées. Nous présentons tout d'abord quels apports les normes ont introduites dans la simulation, puis dans quelle mesure les comportements obtenus correspondent au paramétrage utilisé.

Apports sur la variété La distribution de la vitesse des véhicules au niveau du second détecteur, au kilomètre 6, est représentée sur la Figure 5.2 [26]. Quand aucune norme n'est utilisée (jeu de normes *no norms*), les vitesses enregistrées sont soit basses, entre 70 et 90 km/h (46 % des véhicules), soit élevées, autour de 130 km/h (40 % des véhicules). La courbe présente une forme en « dos de chameau », peu de véhicules adoptant des vitesses moyennes, entre 90 et 130 km/h. Cette distribution peut s'expliquer par le fait que dans ce cas, le comportement des véhicules est trop similaire pour leur permettre de s'adapter aux petites fluctuations du trafic : la voie de gauche reste lente, la droite rapide, et les véhicules effectuent peu de changements de voie ou de dépassements. La vitesse moyenne des véhicule reste basse, à environ 91.5 km/h.

Lorsqu'une norme est introduite, avec le jeu de normes *normal only*, la distribution des vitesses est plus équilibrée : 60 % des véhicules se déplacent à une vitesse comprise entre 90 et 115 km/h, et 30 % entre 115 et 140 km/h. La forme de la courbe se rapproche de celle d'une gaussienne centrée autour de 110 km/h. Dans ce cas, les véhicules se comportent de manière plus dynamique, effectuant des changements de voie et ne restant pas confinés sur la même tout au long du parcours. Par ailleurs, la vitesse maximale des véhicules est en moyenne plus faible que dans le cas *no norms* : elle est répartie entre 100 et 140 km/h, centré sur 120, au lieu d'être constante à 130. Pourtant la vitesse moyenne des véhicules est de 100.4 km/h, contre 91.5 km/h dans le cas précédent : cette augmentation montre que la variété des comportements permet d'améliorer le déplacement des véhicules sur le réseau.

26. Notons que les résultats obtenus sur les autres détecteurs sont similaires.

	no norms	normal only	all norms
vitesse moyenne (détecteur 2)	91.5 km/h	100.4 km/h	103.7 km/h
temps de parcours	5 min 35 sec	4 min 56 sec	5 min 14 sec

TABLE 5.3 – Vitesse moyenne au niveau du détecteur 2 et temps de parcours moyen sur la section.

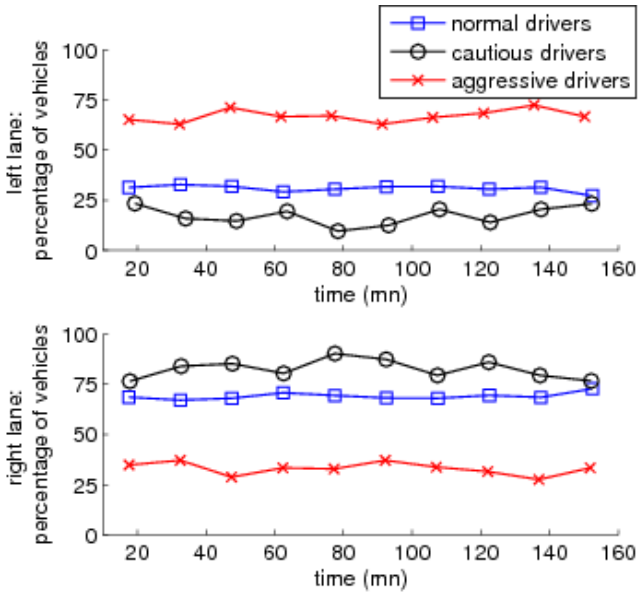


FIGURE 5.3 – Répartition des véhicules entres les voies gauche et droite de l'autoroute.

Dans le troisième cas, lorsque trois normes sont utilisées, la distribution présente un aspect similaire à celle obtenue avec une seule norme. La vitesse moyenne des véhicules est de 103.7 km/h : en augmentant la diversité des paramétrages, les véhicules peuvent encore mieux exploiter l'occupation de la route pour augmenter leur vitesse. Notons que la distribution s'élargit dans les hautes et les basses vitesses, du fait des ensembles de définition utilisés pour la génération. L'introduction de normes supplémentaires permet bien de renforcer les profils extrêmes, comme on peut le voir aux reprises de la courbe autour de 70 et de 150 km/h. L'effet obtenu correspond à l'objectif poursuivi.

Calculé entre les détecteurs 1 et 3, le temps de parcours des véhicules sur la section fournit des résultats complémentaires intéressants (Tab. 5.3). L'introduction d'une norme fait décroître la valeur moyenne du temps de parcours des véhicules de -11.6 %. Cela correspond aux observations précédentes concernant la vitesse : avec plus de variété, les véhicules occupent le réseau routier de manière plus efficace, ce qui accélère leur déplacement moyen par rapport au cas *no norms*. Toutefois, lorsque des comportements plus extrêmes sont introduits avec l'utilisation de normes plus nombreuses (cas *all norms*), le temps de parcours ne diminue pas, mais au contraire augmente de +6.1 % par rapport au cas *normal only*. Dans le même temps, la vitesse moyenne mesurée au niveau des détecteurs augmente pourtant (+3.3 % au détecteur 2 par exemple). Cette observation s'explique par le fait que bien que certains véhicules prennent des vitesses maximales plus élevées, la présence de véhicules lents limite en moyenne leur progression sur le réseau. Par exemple, un véhicule lent effectuant un dépassement a un impact sur le temps de parcours de nombreux autres véhicules.

Représentativité des comportements La Figure 5.3 présente la répartition des véhicules entre la voie de gauche et la voie de droite de l'autoroute, en fonction de leur norme de référence. Les mesures correspondent au détecteur 2, dans le cas *all norms*. Elles sont représentées en pourcentage du nombre total de véhicules ayant traversé le détecteur pendant la durée considérée, ici des intervalles de 10 minutes.

On observe que la majorité des conducteurs agressifs se trouvent sur la voie de gauche (71 % en moyenne sur la durée de la simulation), alors que la plupart des conducteurs prudents restent sur la voie de droite (82 % en moyenne). Les conducteurs normaux sont répartis à 72 % - 28 % entre la voie de gauche et la voie de droite. Conformément au paramétrage qui leur a été assigné, les conducteurs agressifs, qui conduisent plus vite et doublent plus facilement, avec moins de marges de sécurité, se trouvent sur la voie de gauche. Au contraire, les prudents restent sur celle de droite, leurs caractéristiques ne les poussant pas à changer de voie. Enfin, le flux sur la voie de gauche représente 34 % du flux total de véhicules, ce qui montre que les conducteurs, s'ils en ont la possibilité, auront tendance à se rabattre à droite.

Synthèse L'introduction de normes dans le trafic de SCANeR™ permet donc bien de répondre aux besoins fonctionnels initiaux : introduire de la variété et de la cohérence dans le comportement des véhicules. En effet, l'ajout de normes plus extrêmes se traduit par l'apparition dans la simulation des comportements correspondants. Par ailleurs, lorsque que la dynamicité du trafic est accrue par l'ajout de conducteurs adoptant des vitesses élevées ou faibles, des phénomènes de trafic apparaissent, comme la diminution de la vitesse moyenne de traversée d'une section. Enfin, les conducteurs adoptent un comportement représentatif de la norme qui leur correspond.

L'utilisation des normes permet donc d'augmenter la variété des comportements observés dans la simulation. De plus, la diversité des paramétrages à l'origine de ces comportements est générée automatiquement, sans intervention manuelle au cours de la simulation et en limitant la complexité de la phase de conception. L'augmentation du nombre de normes permet par ailleurs de spécifier la simulation de manière plus précise, en privilégiant certains comportements ou en en introduisant de nouveaux.

5.2.3 Apprentissage des normes

Le module « Traffic Analysis » nous a permis de réaliser une première évaluation de la partie observation et analyse du modèle. Dans cette première étape, nous avons décidé de baser les expérimentations sur des données enregistrées depuis la simulation. De cette façon, les normes initiales ayant conduits aux enregistrements sont connues. Les données utilisées proviennent de la simulation utilisant le jeu de normes *all norms*. Les trois normes N_{prudent}, N_{normal} et N_{agressif} sont utilisées, les véhicules étant générés dans les proportions 10 % - 80 % - 10 %.

Le graphe de gauche de la Figure 5.4 représente les vitesses maximales et les temps de sécurité enregistrés par le détecteur 2. Ces paramètres étant statiques, ils correspondent à la valeur initiale affectée au véhicule lors de sa création. Conformément aux spécifications, les domaines des vitesses maximales sont disjoints, et les domaines des temps de sécurité se recouvrent partiellement. Le jeu de données comportant deux paramètres, le réseau de Kohonen comporte $(2+1)^2 = 9$ neurones, qui correspondent à autant de normes inférées. Les losanges représentés sur la figure sont les poids des différents neurones, représentant les valeurs par défaut des paramètres de la norme. On constate que sur un ensemble peu bruité de données comme celui utilisé ici, les normes inférées permettent bien de représenter les comportements des agents.

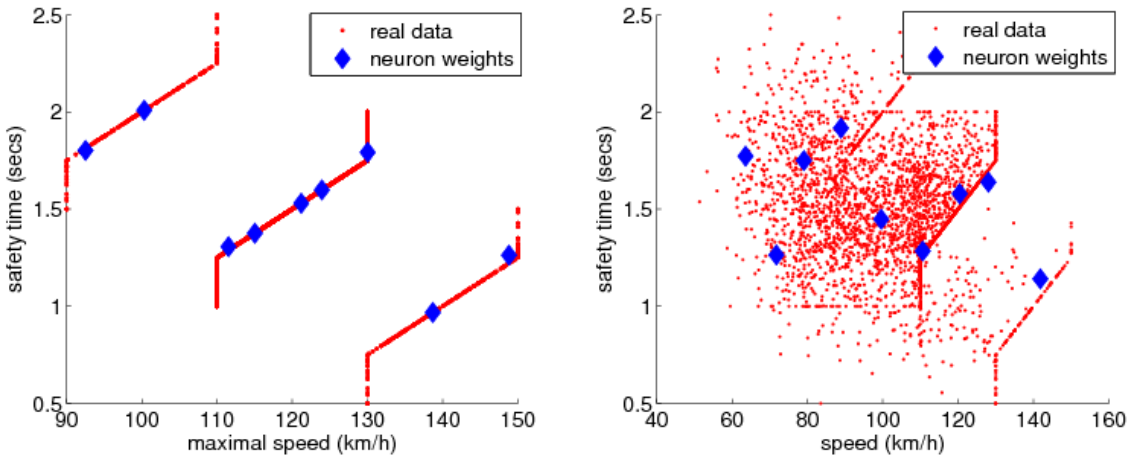


FIGURE 5.4 – Inférence de normes : à gauche en basant sur la vitesse maximale et le temps de sécurité ; à droite sur la vitesse observée et le temps de sécurité.

Cependant, une limite de l'approche apparaît au niveau de la définition des domaines de définition des paramètres. En effet, ceux-ci sont construits à l'aide des valeurs extrêmes prises par les paramètres dans les agents appartenant à cette norme. Autrement dit, dans un cas comme celui-ci, l'espace dans lequel sera généré un paramétrage est un rectangle autour du point représentant la valeur par défaut. Comme on peut le constater sur la figure, cela ne représente pas exactement les données, et la dispersion induite par la génération augmente dans les zones où les normes représentent des comportements correspondant à des paramétrages plus éloignés (ici pour les conducteurs prudents et les conducteurs agressifs). Notons que deux pistes sont envisageables pour y remédier. La première est d'ajouter au processus une décomposition en composantes principales, qui permettrait de restreindre l'espace de sortie des paramètres. La seconde est l'exclusion des valeurs extrêmes lors de la construction des domaines de définition, afin de limiter la sur-représentation de comportements trop éloignés de la moyenne.

La partie droite de la Figure 5.4 représente le temps de sécurité en fonction de la vitesse observée dans la simulation. Les données se rapprochent ainsi plus de données qui peuvent être obtenues à partir d'observations réelles, comme celles provenant de détecteurs à boucle d'induction. Les normes obtenues après apprentissage représentent ici encore bien l'espace de données observées. On peut par ailleurs noter que la limite liée à la construction des domaines de définition, présentée dans le paragraphe précédent, sera à la fois plus limitante, car les valeurs extrêmes sont plus éloignées, tout en ayant potentiellement une influence plus faible, du fait de la forte dispersion des données.

Les résultats obtenus sont donc prometteurs, et l'application sur un jeu de données réelles permettra de valider les éléments complémentaires dont pourrait bénéficier le modèle.

5.3 Évaluation du trafic dans scaner™

Dans cette section, nous présentons les évaluations réalisées pour valider le trafic de SCANeR™. Après avoir introduit nos objectifs, nous abordons deux cas classiques en simulation de trafic : la congestion sur autoroute, liée à l'atteinte du maximum de capacité de la route, et l'insertion sur l'autoroute, qui crée des perturbations en amont de la bretelle.



FIGURE 5.5 – Schéma de la base de données utilisée pour l'expérimentation de la congestion sur autoroute.

période	1	2	3	4	5	6	7	8	9	10	11
nbre veh	100	250	500	750	1000	1500	1000	750	500	100	100

TABLE 5.4 – Nombre de véhicules générés pendant chaque période de 10 minutes.

5.3.1 Objectif

L'introduction des normes et l'automatisation de la création de flux de véhicules a rendu possible l'évaluation de SCANeR™ en tant qu'outil de simulation de trafic. En effet, il était auparavant nécessaire de créer manuellement l'ensemble des véhicules, ce qui ne permettait pas de réaliser des simulations impliquant des volumes importants de véhicules, comme celles permises par les outils spécialisés dans l'étude du trafic.

Il nous a donc paru intéressant d'évaluer dans quelle mesure SCANeR™ était en mesure de répondre à ce type de besoin. Dans le cadre de l'utilisation qui est faite chez Renault, cela permettrait d'introduire l'application dans des étapes supplémentaires de la conception véhicule. En effet, pour les applications de cartographie et de prévision du trafic qui sont aujourd'hui incluses sur certains véhicules, la simulation du trafic au niveau de l'ensemble du réseau routier est intéressante. Utiliser SCANeR™ pour répondre à ce type de besoin permettrait ensuite d'introduire facilement un conducteur dans la boucle lors de leur évaluation, sans avoir à changer d'outil.

Une autre possibilité est le couplage avec un outil commercial de simulation de trafic. Il s'agirait alors de simuler les flux au niveau du réseau en utilisant ce type d'outil, puis de confier à SCANeR™ la gestion microscopique des véhicules se trouvant à proximité du simulateur. De cette façon, le simulateur est introduit de façon transparente dans la simulation au niveau réseau, ce qui permet de bénéficier des résultats à différents niveaux de granularité.

5.3.2 Congestion sur autoroute

Le premier cas de trafic que nous avons cherché à reproduire est la simulation de congestion sur autoroute. Le phénomène que nous cherchons à reproduire est l'apparition de ralentissements sur l'autoroute : au delà d'une certaine demande de trafic, la capacité maximale du réseau routier ne lui permet plus d'absorber le flux, et des ralentissements apparaissent.

Pour ce faire, nous avons créé une base de données constituée d'un tronçon d'autoroute 2 fois 2 voies, en ligne droite et d'une longueur de 1500 m (Fig. 5.5). Un détecteur est placé à 700 m après début du tronçon. Une demande de trafic variable est générée au début de celui-ci, suivant un échelon variant toutes les 10 minutes, pendant 110 minutes (1 h 50 mins). Cette demande de trafic est détaillée dans le Tableau 5.4. Notons que c'est le développement des générateurs de trafic qui permet d'introduire une telle demande, cette fonctionnalité n'étant pas disponible initialement dans SCANeR™.

Afin d'évaluer les résultats obtenus avec SCANeR™, nous avons réalisé une comparaison avec un outil de simulation de trafic dédié à l'ingénierie du trafic, AIMSUN. La Figure 5.6 présente

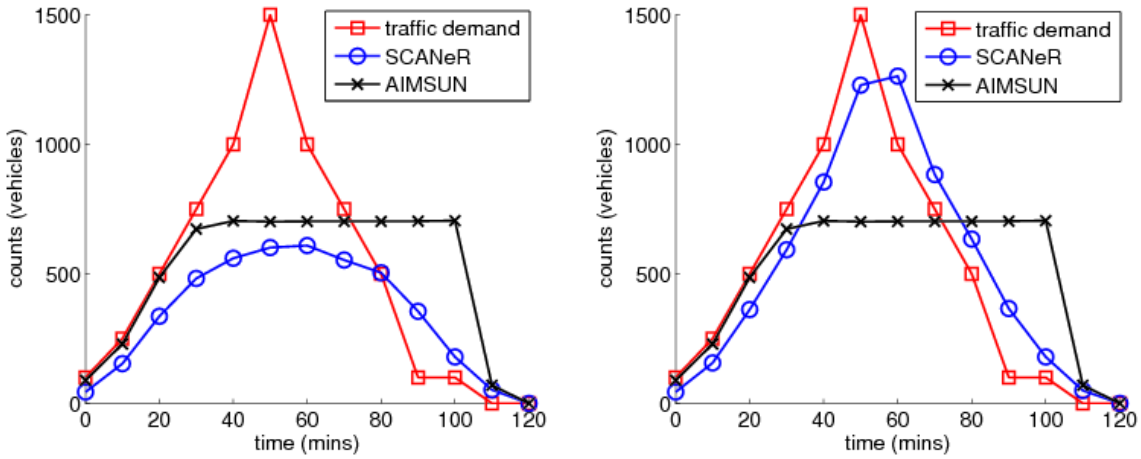


FIGURE 5.6 – Comparaison des flux mesurés avec SCANeR™ et AIMSUN par rapport à la demande, avec deux paramétrages différents pour les normes.

deux graphes représentant la demande de trafic, le flux mesuré au niveau du détecteur dans SCANeR™, et celui mesuré dans AIMSUN. Sur le graphe de gauche, le trafic de SCANeR™ est généré en utilisant uniquement la norme N_{normal} : on constate que le flux mesuré ne correspond pas à la demande de trafic. Le phénomène observé dans la simulation est qu'au-delà d'une certaine demande (entre 500 et 750 véhicules pour 10 minutes, soit 3000 à 4500 véhicules heure) la vitesse moyenne sur le réseau diminue, et que les générateurs ne créent plus de véhicules par manque d'espace : le véhicule créé au pas de temps t est encore présent au dessus du générateur au pas de temps $t+dt$ où ce dernier devrait en générer un nouveau. Pour résoudre ce problème, nous avons modifié la norme utilisée par les véhicules en diminuant le temps de sécurité, et donc la distance inter-véhiculaire moyenne sur le réseau. En utilisant des valeurs comprises entre 0.1 et 2 secondes et en centrant la distribution sur 0.5 s, on obtient le graphe représenté sur la partie droite de la Figure 5.6. Dans ce cas, le volume total de trafic généré correspond bien à la demande. Cette application illustre la flexibilité des normes pour reproduire des situations particulières.

On constate cependant que le phénomène de « stockage » par le réseau, qui apparaît bien avec AIMSUN, n'est pas reproduit avec SCANeR™. Pour ce dernier, le flux de trafic suit la demande, avec un décalage lié au temps de parcours entre le point de génération et l'emplacement du détecteur. Un infléchissement se produit pour la demande maximale, qui est ensuite lissé lorsqu'elle diminue progressivement. En fait, la diminution de la marge de sécurité introduite par la modification de la norme permet aux véhicules de se suivre de manière très rapprochée, ce qui biaise le processus.

Afin de reproduire le phénomène recherché, il est en fait nécessaire de modifier la prise en compte de certains paramètres dans le modèle de décision. En particulier, les véhicules n'amortissent pas assez les variations du flux, ce qui leur permet de repartir très rapidement et évite les ralentissements. De plus, la dépendance temporelle des marges de sécurité peut également être affinée, et être prise en compte différemment suivant la plage de vitesse adoptée par le véhicule. Notons que ces éléments, aujourd'hui en cours de développement, n'étaient pas pris en compte car aucun besoin n'avait jusqu'alors été exprimé. C'est l'extension du champ d'application du logiciel qui les a créés.

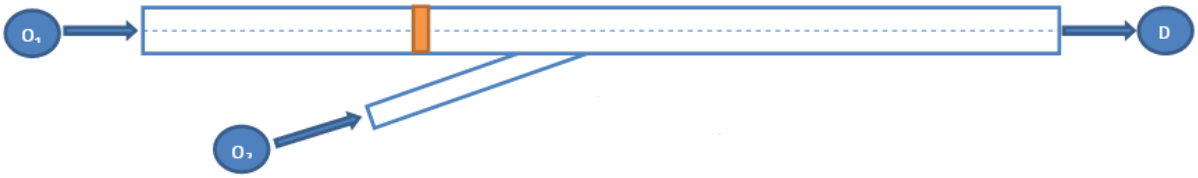


FIGURE 5.7 – Schéma de la base de données utilisée pour l'expérimentation de l'influence d'une voie d'insertion.

période	1	2	3	4	5	6
nbre veh autoroute	100	200	400	800	1600	160
nbre veh voie d'insertion	10	20	40	80	160	16

TABLE 5.5 – Nombre de véhicules générés pendant chaque période de 10 minutes.

5.3.3 Influence des voies d'insertion

Le second cas d'application que nous avons considéré concerne l'influence d'une voie d'insertion sur autoroute. Le phénomène que nous souhaitons reproduire est l'apparition de ralentissements sur la voie principale au delà d'un certain seuil dans le flux de véhicules cherchant à s'insérer.

La base de données créée est une autoroute 2 fois 2 voies, en ligne droite, de 4000 m de longueur. La voie d'insertion se trouve à 2500 m après le début du tronçon, et un détecteur est placé à 2000 m (Fig. 5.7). Une demande de trafic variable est générée au début du tronçon d'autoroute, et une autre au début de la voie d'insertion. Ces demandes sont créées en utilisant une norme N_{normal} modifiée en réduisant le temps de sécurité (cf. Sect. 5.3.2). Les valeurs de la demande de trafic sont précisées dans le Tableau 5.5.

Dans SCANeR™, les véhicules coopèrent afin de faciliter l'insertion des flux entrant sur une autoroute. Pour ce faire, les véhicules de la voie principale déboîtent pour libérer l'espace nécessaire aux véhicules arrivants. Toutefois, au-delà d'une demande de trafic de 400 véhicules par 10 minutes (2400 véhicules heure), nous avons été confronté à une difficulté particulière : le flux sur la voie principale est trop important pour permettre une coopération efficace, et les véhicules arrivants sur la voie secondaire ne parviennent plus à s'insérer. Une situation de blocage est alors rencontrée, les véhicules bloqués s'immobilisant sur la voie d'insertion.

Le cas d'application n'a ainsi pas pu fournir de résultats sur le phénomène recherché, car son bon fonctionnement nécessite des modifications du modèle de décision des véhicules autonomes. L'insertion sur autoroute, tout à fait satisfaisante dans le cadre de l'utilisation classique pour la simulation de conduite, n'est pas en mesure de prendre en compte les demandes de trafic très importantes que nous sollicitons dans ce cas d'utilisation.

5.3.4 Synthèse

En conclusion, les nouveautés introduites dans SCANeR™ à travers les générateurs de trafic, qui permettent de créer une demande de trafic variable, offrent la possibilité d'explorer d'autres usages de l'application. Cependant, celle-ci a été conçue pour répondre aux besoins de la simulation de conduite, et non de la simulation de trafic. Aussi, différentes évolutions sont nécessaires pour permettre d'avancer dans cette voie. Le besoin ayant été identifié, ces évolutions

sont aujourd'hui en cours de développement et seront disponibles dans des versions futures de l'application.

5.4 Discussion

Cette partie concerne la discussion des éléments présentés dans ce travail. Nous abordons tout d'abord la discussion du modèle de différenciation comportementale proposé, avant de nous intéresser à son application à la simulation de trafic.

5.4.1 Discussion du modèle de différenciation comportementale

Deux points particuliers du modèle de différenciation comportementale proposé dans les Chapitres 2 et 3 peuvent être discutés.

Représentation des comportements Tout d'abord, l'objectif de ce travail était d'améliorer le réalisme du comportement des agents dans la simulation. L'approche que nous avons proposée s'intéresse à leur variété et leur conformité, et repose sur l'utilisation des paramètres existants des agents. Il se pose donc la question de savoir dans quelle mesure ces paramètres représentent le comportement des agents.

Nous pensons qu'il n'existe pas de réponse absolue à cette question, très liée à l'outil de simulation utilisé. Ces paramètres, qui peuvent d'ailleurs être enrichis si nécessaire, fournissent un moyen d'accès simple et intuitif aux comportements des agents, même s'ils ne permettent peut-être pas de les maîtriser entièrement. Ce critère répond de manière suffisante au besoin que nous avions fixé.

Notons par ailleurs que dans ce travail les choix réalisés ont été basés sur des contraintes opérationnelles, qui ont poussé l'exploitation du modèle dans la direction qui l'a amené à l'application présentée dans le Chapitre 4. En particulier, les implémentations réalisées, que ce soit pour la création de créatures d'espèces différentes ou la simulation de trafic, s'appuient uniquement sur des paramètres numériques du modèle de décision des agents. Ce choix a été fait pour deux raisons. Tout d'abord, il permettait de coupler de manière non-intrusive le modèle avec l'application SCANeR™. Le développement du créateur de créatures visait à démontrer l'utilisation du modèle sur un autre cas, mais avec une architecture identique à celle déjà mise en œuvre pour le trafic, ce qui nous a conduit à utiliser également uniquement des paramètres numériques. Ensuite, l'utilisation de tels paramètres permet aux utilisateurs finaux d'exploiter intuitivement les possibilités du modèle, par exemple en modifiant les normes à leur guise, comme nous l'avons nous même fait dans la Section 5.3 en adaptant les temps de sécurité. C'est un avantage important, en particulier au vu des besoins industriels que nous avions à satisfaire. Enfin, nous avions fait le postulat de ne pas modifier le moteur de simulation, et donc d'utiliser les paramètres existants en nous accommodant des limites qu'ils pouvaient présenter, comme leur interdépendance.

Validation des apports du modèle Un second point de discussion concerne la validation des apports du modèle. Dans la Section 5.2, nous avons présenté une évaluation de ses apports dans le cadre de la simulation de trafic. Elle a permis de montrer que le modèle permet bien d'introduire aisément de la variété dans la simulation, et de représenter les comportements des

agents à travers les normes. Cependant, il reste difficile de démontrer ces apports en dehors d'un cadre applicatif. Le modèle pourrait pourtant bénéficier de la construction de critères permettant de quantifier son apport en fonction du type de simulation utilisé ou de la situation à reproduire. Cela présenterait de plus l'avantage de fournir une référence, sans avoir à se reposer sur des mesures fortement dépendantes de la simulation à laquelle est couplée le modèle.

5.4.2 Discussion de l'application à la simulation de trafic

Dans le cas de l'application du modèle à la simulation de trafic, les principaux points de discussion concernent les choix réalisés pour le paramétrage du modèle : choix des normes, choix des paramètres utilisés dans les normes et choix des valeurs pour ces paramètres. Nous abordons également les questions relatives à la validation et à l'exploitation des violations.

Choix des normes La première question concerne le choix des normes utilisées dans le cadre de l'application. Comme nous l'avons présenté dans la Section 4.2.2.2 (p. 112), nous nous sommes appuyés sur des expérimentations présentées dans la littérature pour décider de proposer par défaut trois normes, correspondant à trois styles de conduite [Wright et al., 2002] : prudent, normal et agressif. Nous pensons que ces trois normes forment un ensemble suffisant pour répondre à la majorité des besoins. De plus, leur paramétrage peut facilement être modifié par les utilisateurs pour répondre précisément à leurs attentes. Par exemple, elles permettent de s'adapter au contexte dans lequel est réalisée la simulation, en augmentant l'agressivité pour une simulation réalisée en Italie par rapport à la France.

Choix des paramètres utilisés dans les normes Les contraintes opérationnelles qui nous ont conduit à décider d'introduire le modèle en tant que module externe au trafic ont contraint le paramétrage utilisable (cf. Sect. 4.2.2.1 p. 111) : les paramètres disponibles étaient ceux du modèle de trafic. Nous avons par ailleurs fait le choix de définir tous les paramètres dans toutes les normes, afin de faciliter la lisibilité de la configuration par les utilisateurs. Il aurait en effet été possible de créer une hiérarchie plus développée, par exemple en factorisant certains paramètres (l'observation de la priorité prend la même valeur dans les normes N_{prudent}, N_{normal} et N_{agressif}). Nos choix visent à simplifier la compréhension et l'utilisation du modèle, afin de le rendre accessible à tous les utilisateurs.

Choix des valeurs de paramètres Les valeurs des paramètres ont été choisies empiriquement, en se basant d'une part sur les valeurs généralement utilisées dans les simulations existantes avec SCANeR™, et d'autre part sur notre connaissance du fonctionnement interne du modèle de trafic. Une première évaluation de ce paramétrage a montré qu'il permettait de reproduire des comportements qui correspondent à ce qui était attendu. Cependant, plusieurs améliorations peuvent être apportées à ce niveau, améliorations que nous aborderons dans la partie concernant les perspectives (Sect. 5.5 p. 139) : la calibration avec des données réelles, et la calibration sur simulateur. Notons que d'une manière générale le choix des valeurs associées au paramétrage de chacun des styles reste arbitraire, même si un processus de calibration peut amener à l'affiner progressivement.

Validation des apports du modèle Comme nous l'avons vu dans la Section 5.1 (p. 125), évaluer les apports du modèle dans une simulation mettant en jeu des éléments subjectifs est un problème difficile. L'approche que nous avons utilisée, basée sur des mesures statistiques du trafic ou l'observation ponctuelle du paramétrage des agents, n'est pas pleinement satisfaisante, même si elles permet d'obtenir des résultats intéressants. En particulier, l'utilisation de mesures statistiques masque certaines des particularités du trafic, qui sont pourtant visibles par observation directe de la simulation, comme l'augmentation du dynamisme du trafic (multiplication des changements de voies et des dépassements) ou le comportement dangereux de certains conducteurs agressifs. De nouveaux indicateurs permettant de mesurer ces éléments pourraient être introduits, afin d'améliorer la quantification des apports du modèle. Notons que nous avons choisi de ne pas recourir à la confrontation de la simulation à des observateurs humains. En effet, la subjectivité de cette méthode nous paraît trop limitante, et elle ne permet de prendre compte qu'un paramétrage particulier, alors que notre objectif est de faciliter la multiplication des paramétrages, pour prendre en compte par exemple les utilisateurs internationaux. Nécessitant une revalidation systématique, cette méthode nous paraît trop contrainte, même s'il est vrai qu'elle peut apporter des éléments intéressants.

Utilisation des violations Enfin, les modules développés ne mettent pas à profit les possibilités offertes par les violations. En effet, comme nous l'avions vu dans la Section 4.2.3.3 (p. 117), elles ne correspondent pas à l'usage ciblé dans les utilisations actuelles. Cependant, elles permettraient d'introduire facilement de la diversité, et de confronter les conducteurs à des comportements non spécifiés, ce qui augmentera le champ d'application de la simulation. Des conducteurs ivres ou perdant le contrôle de leur véhicule peuvent en effet être introduits par leur intermédiaire.

5.5 Perspectives

Dans cette dernière partie, nous présentons tout d'abord les perspectives du modèle, avant de nous intéresser à celles de son application à la simulation de trafic. Pour finir, nous abordons plusieurs évolutions complémentaires du modèle de trafic qui permettent d'introduire des fonctionnalités complémentaires.

5.5.1 Perspectives du modèle de différenciation comportementale

Au regard du développement actuel du modèle de différenciation comportementale et de l'outil le mettant en application, deux axes de perspectives principaux peuvent être distingués. Le premier concerne la mise en application et l'évaluation des possibilités de l'outil qui n'ont pas été mises à profit dans les applications réalisées pour l'instant. Le second axe s'intéresse au développement des aspects de création automatique de normes.

Exploiter le potentiel de l'outil Un certain nombre de fonctionnalités fournies par le modèle n'ont été que peu ou pas exploitées dans les applications implémentées. Parmi celles-ci, les propriétés des normes, les violations et les institutions présentent des possibilités intéressantes. Les propriétés des normes permettent d'envisager l'introduction de normes spécifiques au contexte, qui seraient activées ou désactivées en fonction de la situation. Un jeu de normes spécifiques

pourrait alors être utilisé suivant qu'un véhicule se trouve en agglomération ou sur une voie rapide, le changement s'effectuant à la volée sur la base d'un paramètre présent dans la base de données. Par ailleurs, les violations n'ont été que peu exploitées dans les applications réalisées. Il serait intéressant de développer leur usage, l'introduction de comportements non-spécifiés dans la simulation pouvant se montrer particulièrement intéressante. Enfin, les institutions n'ont pour l'instant été utilisées que comme conteneur de normes. Tout comme les propriétés, il est pourtant envisageable de les exploiter pour adapter l'usage des normes en fonction du contexte. L'utilisation des propriétés nécessite une adaptation du modèle de la simulation, alors que les institutions permettent de substituer un jeu de norme à un autre, ce qui offre de la flexibilité et une utilisation aisée.

Un autre axe pour l'exploitation de l'outil est de l'appliquer sur des paramètres qui ne soient pas numériques, comme par exemple des règles de simulation. Il pourrait alors servir à gérer l'ensemble de la simulation en lieu et place du modèle de décision. Par ailleurs, le modèle peut être utilisé comme un outil indépendant. L'usage principal que nous avons présenté est la maîtrise du paramétrage des agents de la simulation. Il peut toutefois aussi bien être utilisé pour générer tout type de paramétrage, comme la configuration globale de la simulation.

Automatiser la création des normes Une des limites conceptuelle de notre approche est la description que nous utilisons pour la description des comportements, basée sur les paramètres de l'application. Elle résulte d'un choix de conception, visant à simplifier les tâches de paramétrisation du modèle. Introduire une description plus haut niveau suppose que les utilisateurs possèdent des connaissances en systèmes normatifs, en plus de connaissances en scénarisation de simulation de trafic. Au vu de notre cible, cela n'était pas envisageable.

Des travaux récents concernant l'implémentation des normes se sont intéressés à des méthodes de génération automatique, afin de simplifier la tâche des concepteurs [Torres da Silva, 2008]. L'idée est de permettre aux concepteurs de ne plus avoir besoin d'être spécialistes à la fois dans le langage de description des normes, et dans celui utilisé pour l'implémentation.

Une extension intéressante du modèle dans notre cas est de suivre le processus inverse de celui mené dans [Torres da Silva, 2008] : nos utilisateurs connaissent le domaine applicatif, mais pas le monde normatif. L'idée est donc qu'ils puissent définir paramètres et domaines de définition comme ils en ont l'habitude aujourd'hui, mais qu'un système normatif avec des règles strictes soit généré automatiquement à partir de ces domaines et paramètres. Cela permet d'appliquer l'ensemble des méthodologies disponibles dans le domaine normatif, et de s'appuyer sur les résultats théoriques, ce que n'offre pas l'approche actuelle. Les normes pourraient ainsi être décrites en utilisant une spécification plus précise. Notons que l'automatisation de la configuration des normes va déjà dans cette direction, et qu'il s'agit ici de l'étape suivante dans le processus de simplification d'utilisation du modèle.

5.5.2 Perspectives de l'application à la simulation de trafic

En ce qui concerne les perspectives relatives à l'application à la simulation de trafic, trois axes principaux sont envisageables : l'amélioration du paramétrage du véhicule, la comparaison avec une situation réelle de trafic et l'application du modèle aux piétons.

Amélioration du paramétrage Une perspective envisageable est de remplacer le paramétrage actuel du modèle de trafic par un paramètre comportemental unique. La déclinaison de

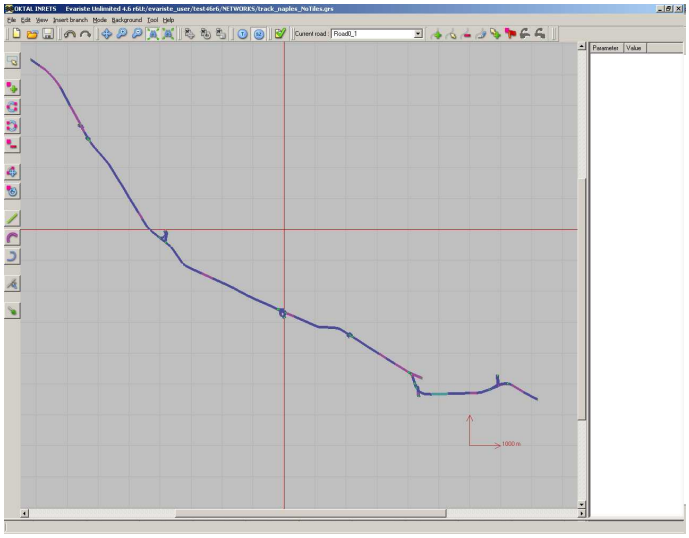


FIGURE 5.8 – Création de la base de données « Napoli – Salerno » dans Evariste à partir de données GIS.

ce paramètre dans le modèle permettrait alors de supprimer l'interdépendance entre les paramètres. Nous proposons d'utiliser un paramètre de *tolérance*, qui serait décliné sur l'ensemble des sous-modèles de trafic. Le comportement des conducteurs peut en effet être décrit de manière simple en utilisant un critère de performance (avec quelle précision la tâche est-elle exécutée ?) et un critère de tolérance (quelle est la marge d'erreur acceptée ?). Par exemple, si la tolérance est trop faible pour une performance moyenne, le comportement est risqué. De manière générale, les modèles peuvent être décrits sous la forme :

$$F(tolerance, performance)$$

Dans le cas du suivi de véhicule par exemple, la tolérance est être décrite par le temps à la collision.

La calibration de ce paramètre peut être réalisée par une expérimentation sur simulateur, confrontant le conducteur à chacune des tâches associées aux sous-modèles de conduite. Avec un échantillon représentatif des différents styles de conduite étudiés, il est alors possible de calibrer le paramètre pour les différentes normes. Notons qu'une des limitations de cette approche est que le paramétrage sera spécifique au pays où est mené la calibration, et que sa robustesse n'est pas assurée.

Comparaison avec un cas réel Une autre perspective intéressante concerne la validation de la simulation de trafic avec un cas réel. Comme nous l'avons vu dans la Section 5.3, cette étape de validation nécessite cependant certaines modifications sur le modèle de trafic de SCANeR™, modifications qui sont aujourd'hui en cours de réalisation. Les premiers éléments destinés à cette validation ont cependant déjà été préparés. Ainsi, une base de donnée représentant une portion d'autoroute italienne entre Naples et Salerne a été recréée dans SCANeR™ (Fig. 5.8), ainsi que les demandes de trafic correspondant aux mesures réelles. La complexité de la base, d'une longueur de 22 km et incluant 7 échangeurs complets, dont certains avec des systèmes de gestion à feux, fait de ce projet un objectif particulièrement intéressant.

Application aux piétons Enfin, le modèle de génération peut être appliqué sur les piétons de SCANeR™. Comme nous l'avons vu avec les exemples illustrant le Chapitre 2, des normes définissant le comportement de piétons peuvent être créées. Par ailleurs, dans SCANeR™ les piétons sont gérés comme des véhicules (cf. Sect 4.1.4), et leur associer une norme spécifique est donc un processus peu complexe. Cette perspective est d'ailleurs un projet qui sera intégré à court terme dans l'application.

5.5.3 Évolutions complémentaires du trafic

Au cours des travaux réalisés, plusieurs possibilités de développements complémentaires du modèle de trafic de SCANeR™ sont apparus. Nous présentons ici deux d'entre eux, qui nous ont paru particulièrement prometteurs : la prise en compte des occlusions et l'utilisation de règles informelles dans le processus de décision des véhicules.

Prise en compte des occlusions Un aspect intéressant et encore peu développé dans le domaine de la simulation de conduite concerne la prise en compte des occlusions dans le processus de décision. La gestion des occlusions est principalement utilisée dans le domaine du graphisme, pour accélérer les processus d'affichage en évitant de calculer le rendu de pixel ou d'éléments qui ne sont pas visibles par l'utilisateur. Des tests de visibilité utilisant des sphères ou des boites englobantes sont utilisés, associés à des techniques de lancer de rayon par exemple.

La prise en compte des occlusions augmente la précision de la perception du véhicule, lui permettant a priori d'avoir un comportement plus réaliste dans certaines circonstances. L'apport de cette prise en compte sur le comportement des conducteurs reste cependant à évaluer, en particulier vis à vis du coût de complexité qu'il entraine. La gestion des occlusions impose de pouvoir situer dans la simulation les éléments mobiles, comme les véhicules, mais aussi les éléments fixes du décor, qui ne font traditionnellement pas partie des descriptions logique de l'environnement. Il faut également être en mesure de les qualifier (dimensions, transparence...), puis de déterminer de quelle manière ils influencent le conducteur, avant de pouvoir enfin retranscrire ce mécanisme dans le modèle de décision. Le processus est donc complexe, pour un gain pour l'instant inconnu, et peut avoir un impact fort sur le modèle de décision : qu'en est-il d'un véhicule aperçu, puis qui disparait pendant quelques instant derrière un bâtiment ? De même, quel est le comportement d'un automobiliste coincé derrière un poids lourd ?

A notre connaissance, aucun simulateur ne prend aujourd'hui en compte ce type de paramètre dans le processus de décision des véhicules autonomes.

Prise en compte des règles informelles Une autre réflexion a été menée sur le sujet de l'amélioration du réalisme comportemental des véhicules autonomes. Nous avons choisi de nous intéresser à la prise en compte de nouvelles règles par les véhicules [Lacroix et al., 2009b].

Les règles informelles permettent de prendre en compte le comportement des autres utilisateurs dans le modèle de décision [Björklung & Aberg, 2005]. Le comportement des conducteurs est basé sur différents paramètres en interaction : les règles formelles de trafic (code de la route), des règles informelles (habitudes ou usages qui peuvent être contradictoires avec les règles formelles, comme forcer le passage à un carrefour ou un rond point), le design de la route (qui est souvent à l'origine de l'apparition des règles informelles) et enfin le comportement des autres conducteurs (aussi bien leur comportement actuel que celui qui est anticipé). Les conducteurs

peuvent ainsi être classifiés en quatre groupes distincts : les « agressifs » (les conducteurs cèdent rarement la priorité quelle que soit la situation), les « informels » (28% des conducteurs, dont le comportement dépend essentiellement du design et non des règles formelles : par exemple ils forcent le passage s'ils sont une route principale avec intersection en priorité à droite sur une plus petite route), les « formels » (qui respectent les règles de priorités ; 38% des conducteurs) et enfin les « prudents » (qui tendent à céder la priorité quelle que soit la situation ; 21% des conducteurs).

L'introduction des ces règles dans SCANeR™ repose sur différents ajouts au sein du modèle. Tout d'abord, la granularité de prise en compte de la signalisation doit être augmentée : la violation d'un feu, d'un stop ou d'un céder le passage n'a pas la même gravité. Le paramètre de respect de la signalisation doit être décliné en trois paramètres différents, qui permettront d'augmenter d'autant la variété. Par ailleurs, le statut de booléen utilisé par ces paramètres est inadapté : un conducteur réel ne commet que ponctuellement ce type d'infraction, et non systématiquement. Un paramétrage représentant un pourcentage de chance de réaliser l'infraction sera plus adapté.

La prise en compte du comportement des autres utilisateurs est particulièrement utile à l'approche des intersections. En effet, si un véhicule s'approche en accélérant sans être prioritaire, nous avons tendance à décélérer par précaution. Afin de permettre aux véhicules de réagir à ce type de comportement, le temps d'arrivée à l'intersection et l'accélération des autres véhicules sont utilisés dans le modèle de décision. Le calcul de la vitesse prend ainsi en compte ces critères supplémentaires.

Enfin, un paramètre dynamique a été intégré. Il s'agit d'un paramètre d'agressivité, qui évolue en fonction de l'atteinte de la vitesse désirée et du temps d'attente aux intersections. Si le conducteur reste trop éloigné de sa vitesse désirée pendant une durée trop longue, son agressivité augmente. Si le conducteur attend trop longtemps à une intersection, son agressivité augmente également. Ce paramètre est utilisé pour influencer dynamiquement les autres paramètres de la simulation : une forte agressivité augmente la prise de risque, et entraîne ainsi le choix d'une vitesse maximale plus grande et de marges de sécurité plus faibles. La calibration des différentes fonctions utilisées reste cependant délicate, du fait des influences multiples des différents critères. Notons que ces différents éléments sont actuellement en cours d'intégration dans le modèle.

5.6 Conclusion

Dans ce chapitre, nous nous sommes tout d'abord intéressés à la calibration et la validation du trafic. La méthodologie utilisée en ingénierie du trafic repose sur la calibration des modèles afin de reproduire des situations observées dans la réalité. Elle fournit une référence à partir de laquelle évaluer les modifications introduites dans la simulation. Cette calibration s'appuie sur la collecte de données réelles, qui peut être réalisée par des systèmes placés sur le réseau, comme les détecteurs à boucle d'induction ou les caméras vidéos, ou par des systèmes embarqués dans les véhicules. Enfin, la validation du trafic dans le cadre des simulateurs de conduite nécessite de prendre en compte le comportement des véhicules autonomes. Elle repose toutefois souvent sur des évaluations subjectives qui n'offrent pas une solution réellement satisfaisante.

Nous présentons dans un second temps les évaluations menées sur les apports du modèle dans le cadre de SCANeR™. Différents jeux de normes ont été évalués, permettant d'illustrer l'augmentation du réalisme du trafic. Le modèle apporte à la simulation la variété souhaitée, et

sa flexibilité permet d'introduire aisément différents comportements. Par ailleurs, le paramétrage utilisé permet de représenter de manière satisfaisante les comportements ciblés. Les méthodes de classification permettant d'inférer le paramétrage des normes ont également été évaluées, donnant des résultats prometteurs. L'introduction de mécanismes complémentaires pourra sans doute encore les améliorer.

Les développements réalisés au cours de la thèse, et tout particulièrement les apports des générateurs de trafic, permettent d'envisager l'extension des usages de SCANeR™à la simulation de trafic. Nous avons présenté les premières expérimentations menées dans ce sens, qui visaient à reproduire des phénomènes comme l'apparition de ralentissement sur autoroute lorsque la capacité maximale du réseau est atteinte ou lorsque le flux d'une voie d'insertion dépasse un seuil critique. SCANeR™ étant dédié à la simulation de conduite, des adaptations du modèle de trafic prenant en compte les flux importants demandés pour ce type d'expérimentation sont encore nécessaires afin de pouvoir obtenir des résultats satisfaisants. Les modifications adéquates sont aujourd'hui en cours, ce besoin étant en émergence et ayant été précisé au cours des travaux.

Le modèle et son application sont ensuite discutés. En particulier, un point important concerne la question de savoir si des paramètres de configuration peuvent représenter le comportement des agents. La validation des apports du modèle en dehors de son application à une simulation est également un sujet d'intérêt. En ce qui concerne l'application au trafic, les différents éléments de paramétrage, comme le choix des normes, celui des éléments les composant, ou encore les valeurs utilisées peuvent être discutés, même s'ils résultent de choix opérationnels ou basés sur des travaux connexes.

Enfin, dans une dernière partie, nous présentons les perspectives de nos travaux. Du point de vue du modèle, deux axes principaux peuvent être poursuivis. Le premier concerne l'exploitation des éléments qui ont été peu mis à profit, comme les violations pour augmenter la variété ou les propriétés des normes pour s'adapter au contexte. Le second axe est celui de la création automatique de normes, qui pourrait être exploitée avec profit pour faciliter la configuration du modèle. Du point de vue de l'application, les perspectives concernent en particulier l'amélioration du paramétrage sur lequel est basé les normes, la validation avec un cas d'utilisation réel et l'application du modèle aux piétons. Différentes améliorations du modèle du trafic sont également envisageables, comme la prise en compte des occlusions ou des règles informelles.

Conclusion

Le Centre Technique de Simulation de Renault développe et exploite des simulateurs de conduite, utilisés pour des applications allant de la mise au point de systèmes d'aide au conducteur aux études d'ergonomie. L'objectif applicatif de cette thèse, effectuée en convention Cifre chez Renault en partenariat avec le Laboratoire d'Informatique Fondamentale de Lille, était d'améliorer le réalisme comportemental du trafic routier afin de renforcer l'immersion des conducteurs. Dans les approches centrées individus, la simulation de trafic dans notre cas, la variété et la cohérence du comportement des agents sont des points cruciaux pour le réalisme. Dans ce travail, nous proposons un modèle de différenciation comportementale les prenant simultanément en compte et permettant d'atteindre notre objectif.

Présentée dans le Chapitre 1, la simulation de trafic routier est un problème complexe, abordé dans des domaines comme l'ingénierie du trafic ou la psychologie de la conduite. Les modèles de simulation utilisés se placent à différents niveaux, en fonction des besoins. Dans le cas des simulateurs de conduite, le conducteur humain est placé au cœur d'un trafic simulé. Celui-ci doit donc être le plus réaliste possible, et impose l'utilisation d'une simulation microscopique, qui permet de prendre en compte au niveau local les interactions entre les véhicules. Dans SCANeR™, l'application développée par Renault pour ses simulateurs de conduite, le trafic est un système multi-agents. En effet, les approches centrées individus sont bien adaptées à ce type de besoin, car elles permettent d'obtenir un modèle explicatif des phénomènes observés, et utilisent un vocabulaire similaire à celui du domaine simulé. Différents travaux abordent la prise en compte de la variété et de la cohérence de comportements ou de paramétrages. Cependant, les approches proposées sont le plus souvent spécifiques au domaine considéré, comme dans le cas de la simulation de foule ou de conducteurs virtuels, ou bien ne cherchent pas à enrichir les comportements mais uniquement à les contraindre, comme dans le cas de l'administration système.

Ce constat nous a conduit à proposer dans le Chapitre 2 un modèle de différenciation comportementale qui soit à la fois flexible et générique. Issues des sciences juridiques et sociales, les approches normatives sont aujourd'hui utilisées dans le contexte des systèmes multi-agents. Elles permettent de réguler les actions des agents et de spécifier un cadre organisationnel. De plus, elles possèdent des similitudes avec les éléments que nous cherchons à simuler : les comportements sur la route correspondent à des normes sociales. Nous avons donc proposé un modèle décrivant les comportements des agents par des normes, en se basant sur leurs capacités descriptives, plutôt que sur leurs capacités prescriptives. Nous les utilisons afin de spécifier les paramètres de la simulation, en les associant avec des domaines de définition. Chaque agent est associé à une norme, qui permet de générer son paramétrage. Lors de cette génération, les paramètres de l'agent sont associés à des valeurs issues des domaines spécifiés dans la norme. L'algorithme utilisé permet également de créer des agents déviants ou en violation avec la norme. Le modèle

fournit les outils permettant de contrôler la fréquence de leur apparition et l'amplitude de leur écart à la norme.

Introduit au Chapitre 3, le troisième axe du modèle concerne l'observation et l'analyse des normes dans la simulation. Plusieurs fonctionnalités permettent en effet d'enrichir et de faciliter son utilisation : l'inférence de norme, qui permet de déterminer quel ensemble de normes est en pratique utilisé par les agents, la classification du comportement des agents, qui permet de visualiser si leur comportement effectif est conforme aux spécifications, et enfin la calibration automatique de la configuration, qui permet de créer une configuration des normes à partir de données enregistrées. Ces trois axes permettent d'atteindre l'objectif visé : créer de la variété tout en contrôlant la conformité des agents. La variété est obtenue à travers la construction des normes, en mettant à profit les définitions des paramètres et leur assemblage afin de créer une grande diversité de normes. La génération des paramètres dans les domaines spécifiés introduit de la variété au sein de chacune d'elle. Par ailleurs, les violations introduisent un autre niveau de variété qui permet d'obtenir des comportements non spécifiés. La conformité est contrôlée par l'utilisation d'une norme régimentée, qui ne peut être violée, et d'un paramètre spécifiant l'écart maximal à la norme autorisé. Fixé par l'utilisateur, il permet de forcer l'application à respecter le paramétrage des normes en toutes circonstancs, de le laisser évoluer librement, ou encore de fixer un seuil de violation à ne pas dépasser. Le modèle de différenciation comportementale peut être utilisé comme un outil extérieur à la simulation, se basant sur ses paramètres externes, ou en tant que module interne, permettant alors de contrôler l'ensemble du paramétrage. Il présente deux avantages principaux : il est non-intrusif si les paramètres de la simulation sont accessibles de l'extérieur de celle-ci, et il permet de réaliser la configuration en dehors de l'agent, ce qui facilite l'utilisation, l'administration et la maintenance. L'outil est également générique, ce qui est illustré par la présentation d'exemples de deux cas d'application, la simulation de foule et la génération de créatures d'espèces différentes.

Dans le Chapitre 4, nous présentons l'application du modèle proposé à la simulation de trafic, avec la suite logicielle utilisée et développée chez Renault, SCANeR™. Dans celle-ci, le module dédié à la simulation du trafic est un système multi-agents, dans lequel chaque véhicule est un agent fonctionnant suivant une boucle « perception – décision – action ». Leur comportement est influencé par des paramètres pseudo-psychologiques, comme une vitesse maximale ou des marges de sécurité. Ces paramètres externes sont accessibles depuis l'extérieur de la simulation, et nous avons choisi d'appliquer le modèle de différenciation comportementale en nous basant sur eux. Cela permet d'assurer la pérennité des développements et la rétro-compatibilité, qui sont importantes dans notre cas. Les modules développés avaient pour objectif d'introduire des styles de conduite pour les conducteurs virtuels et de faciliter la conception des scénarios. Le premier module permet d'appliquer manuellement le modèle sur un scénario existant, en régénérant le paramétrage des véhicules suivant la norme choisie. Le second module permet de générer dynamiquement des flux de trafic au cours de la simulation, en spécifiant la ou les normes à utiliser ainsi que leurs proportions respectives. Les générateurs de trafic ainsi définis permettent de peupler l'environnement de conduite, par exemple pour préparer une situation initiale ou pour préserver une certaine densité de véhicules autour du conducteur réel. Enfin, le troisième module permet d'analyser les enregistrements de la simulation afin d'inférer les normes utilisées en pratique par les agents. Notons que ces développements sont déjà en partie intégrés dans la version commerciale de SCANeR™.

L'évaluation des apports du modèle est présentée dans le Chapitre 5. Après avoir abordé la problématique de la validation du trafic, nous présentons les résultats obtenus sur différentes expérimentations. L'introduction de normes permet d'augmenter la variété des comportements

observés, et les styles de conduite introduits reproduisent les comportements attendus. Par ailleurs, la génération automatisée de flux de trafic aux propriétés configurables permet d'envisager l'utilisation de SCANeR™ comme outil de simulation de trafic, afin d'élargir son spectre d'utilisation. Nous présentons les premières évaluations réalisées pour valider l'application pour cet usage. Nous discutons ensuite le modèle et l'application que nous en avons faite. En ce qui concerne le modèle, la question principale est de savoir si la spécification de paramètres suffit à caractériser le comportement des agents. Au niveau de l'application, le choix des normes et de leur paramétrage, ainsi que la difficulté à les valider, sont les principales questions en suspend. Nous terminons enfin en présentant les perspectives de nos travaux. Tout d'abord, le modèle peut être étendu pour utiliser des éléments autres que des paramètres, et l'automatisation de la configuration des normes doit être validée plus avant. Pour l'application au trafic, l'utilisation de nouveaux paramètres, l'introduction de nouvelles règles de comportement pour les véhicules, ainsi que l'application du modèle aux piétons font partie des perspectives envisageables.

Bibliographie

[Ajzen & Fishbein, 1977] Ajzen, I. & Fishbein, M. (1977). Attitude-behavioral relations : A theoretical analysis and review of empirical research. *Psychological Bulletin*, 84(5), 888–918.

[Alexiadis et al., 2004] Alexiadis, V., Colyar, J., Halkias, J., Hranac, R., & McHale, G. (2004). Next generation simulation program. *ITE Journal*, (pp. 22–26).

[Anderberg, 1973] Anderberg, M. R. (1973). *Cluster Analysis for Applications*. Academic Press, New York.

[Barbuceanu, 1998] Barbuceanu, M. (1998). Agents that work in harmony by knowing and fulfilling their obligations. In *AAAI Conference on Artificial Intelligence* (pp. 89–96). Madison, USA : AAAI Press.

[Barcelò & Casas, 2002] Barcelò, J. & Casas, J. (2002). Dynamic network simulation with AIMSUN. In *International Symposium on Transport Simulation* Yokohama, Japan.

[Barcelò et al., 1999] Barcelò, J., Casas, J., Ferrer, J. L., & Garcia, D. (1999). Modelling advanced transport telematic applications with microscopic simulators : The case of AIMSUN. In M. S. W. Brilon, F. Huber & H. Wallentowitz (Eds.), *Traffic and Mobility, Simulation, Economics, Environment* : Springer.

[Bazzan, 2005] Bazzan, A. L. C. (2005). A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agents Systems*, 10(2), 131–164.

[Bellet et al., 2005] Bellet, T., Bailly, B., Mayenobe, P., Georgeon, O., Tattegrain-Veste, H., Martin, R., Mille, A., & Trassoudaine, L. (2005). Cognitive modelling and simulation of drivers mental models. In *International Workshop on Modelling Driver Behaviour in Automotive Environments* (pp. 165–184). Ispra, Italy.

[Berrou et al., 2005] Berrou, J.-L., Beecham, J., Quaglia, P., Kagarlis, M., & Gerodimos, A. (2005). Calibration and validation of the Legion simulation model using empirical data. In *Pedestrian and Evacuation Dynamics* (pp. 167–181). : Springer.

[Biasillo, 2002] Biasillo, G. (2002). Representing a race track for the AI. In S. Rabin (Ed.), *AI Game Programming Wisdom* (pp. 444–454). : Charles River Media.

[Björklung & Aberg, 2005] Björklung, G. M. & Aberg, L. (2005). Driver behaviour in intersections : Formal and informal traffic rules. *Transportation Research Part F*, 8, 239–253.

[Boella et al., 2006] Boella, G., van der Torre, L., & Verhagen, H. (2006). Introduction to normative multiagent systems. *Computation and Mathematical Organizational Theory, special issue on Normative Multiagent Systems*, 12(2-3), 71–79.

[Boella et al., 2007] Boella, G., van der Torre, L., & Verhagen, H. (2007). Introduction to normative multiagent systems. In G. Boella, L. van der Torre, & H. Verhagen (Eds.), *Dagstuhl Seminar Proceedings : Normative Multi-agent Systems* (pp. 1–7). Dagstuhl, Germany.

[Boella et al., 2008] Boella, G., van der Torre, L., & Verhagen, H. (2008). An interactionist view on normative multiagent systems. *Autonomous Agents and Multi-Agent Systems, special issue on Normative Multiagent Systems*, 17(1), 1–10.

[Boer, 1999] Boer, E. (1999). Car following from the driver's perspective. *Transportation Research Part F*, 2, 201–206.

[Bou et al., 2007] Bou, E., López-Sánchez, M., & Rodríguez-Aguilar, J. A. (2007). Adaptation of autonomic electronic institutions through norms and institutional agents. In P. Noriega, J. Vázquez-Salceda, G. Boella, O. Boissier, V. Dignum, N. Fornara, & E. Matson (Eds.), *Engineering Societies in the Agents World VII*, volume 4457 of *LNCS* (pp. 300–319). : Springer.

[Brackstone & McDonald, 1999a] Brackstone, M. & McDonald, M. (1999a). Car-following : a historical review. *Transportation Research Part F*, 2, 181–196.

[Brackstone & McDonald, 1999b] Brackstone, M. & McDonald, M. (1999b). What is the answer ? and come to that, what are the questions ? *Transportation Research Part F*, 2, 221–224.

[Broersen et al., 2001] Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., & van der Torre, L. (2001). The BOID architecture : Conflicts between beliefs, obligations, intentions and desires. In *International Conference on Autonomous Agents* (pp. 9–16). Montreal, Canada : ACM.

[Bruyninckx, 2006] Bruyninckx, H. (2006). Real-time and embedded guide. `http://people.mech.kuleuven.be/~bruyninc/rthowto/rtHOWTO.pdf`, consulté le 20/04/2009.

[Burgess & Couch, 2006] Burgess, M. & Couch, A. (2006). Modeling next generation configuration management tools. In *Large Installation System Administration Conference* (pp. 131–147). Washington D.C., USA.

[Cal3D, 2009] Cal3D (2009). Cal3D home page. `http://cal3d.sourceforge.net`, consulté le 17/03/2009.

[Caliper, 2009] Caliper (2009). TransCAD home page. `http://www.caliper.com/TransCAD/`, consulté le 15/04/2009.

[Carles & Espié, 1999] Carles, O. & Espié, S. (1999). Database generation system for road applications. In *Driving Simulation Conference* (pp. 87–103). Paris, France.

[Caro et al., 2007] Caro, S., Cavallo, V., Boer, E., & Vienne, F. (2007). The influence of fog on motion discrimination thresholds in car following. In *4th International Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design* (pp. 446–451). Stevenson, USA.

[Carpenter & Grossberg, 1987] Carpenter, G. & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37, 54–115.

[Castelfranchi et al., 2000] Castelfranchi, C., Dignum, F., Jonker, C. M., & Treur, J. (2000). Deliberative normative agents : Principles and architecture. In *Intelligent Agents VI. Agent Theories, Architectures and Languages*, volume 1757 of *LNCS* (pp. 364–378). : Springer.

[Cayford et al., 1997] Cayford, R., Wei-Hua, L., & Daganzo, C. F. (1997). *The Netcell Simulation Package : Technical Description.* Technical Report UCB-ITS-PRR-97-23, University of California, Berkeley, USA.

[Champion, 2003] Champion, A. (2003). *Mécanisme de coordination multi-agent fondé sur des jeux : application à la simulation comportementale de trafic routier en situation de carrefour.* PhD thesis, Université de Valenciennes et du Hainaut-Cambrésis, France.

[Champion et al., 1999] Champion, A., Mandiau, R., Kolski, C., Heidet, A., & Kemeny, A. (1999). Traffic generation with the SCANeR™ II simulator : Towards a multi-agent architecture. In *Driving Simulation Conference* (pp. 311–324). Paris, France.

[Champion et al., 2002] Champion, A., Zhang, M.-Y., Auberlet, J.-M., & Espié, S. (2002). Behavioural simulation : Towards high-density network traffic studies. In K. Wang, G. Xiao, L. Nie, & H. Yang (Eds.), *Third International Conference on Traffic and Transportation Studies (ICTTS)* Guilin, Chine : American Society of Civil Engineers.

[Chandler et al., 1958] Chandler, R. E., Herman, R., & Montroll, E. (1958). Traffic dynamics : Studies in car following. *Operations Research*, 6(2), 165–184.

[Ciuffo et al., 2007] Ciuffo, B., Punzo, V., & Torrieri, V. (2007). A framework for the calibration of microscopic traffic flow models. In *Transportation Research Board* Washington DC, USA.

[Ciuffo et al., 2008] Ciuffo, B., Punzo, V., & Torrieri, V. (2008). Comparison of simulation-based and model-based calibrations of traffic-flow microsimulation models. *Transportation Research Record*, 2088, 36–44.

[Cohen, 1993] Cohen, S. (1993). *Ingénierie du trafic. Éléments de théorie du trafic et applications.* Presses de l'École Nationale des Ponts et Chaussées, Paris, France.

[Conte et al., 1999a] Conte, R., Castelfranchi, C., & Dignum, F. (1999a). Autonomous norm acceptance. In *Intelligent Agents V : Agents Theories, Architectures, and Languages*, volume 1555 of *LNCS* (pp. 99–112). : Springer.

[Conte et al., 1999b] Conte, R., Falcone, R., & Sartor, G. (1999b). Introduction : Agents and norms : How to fill the gap ? *Artificial Intelligence and Law*, 7(1), 1–15.

[Corkill, 1991] Corkill, D. (1991). Blackboard systems. *AI Expert*, 6(9), 40–47.

[Couch et al., 2003] Couch, A., Hart, J., Idhaw, E., & Kallas, D. (2003). Seeking closure in an open world : A behavioral agent approach to configuration management. In *Large Installation System Administration Conference* (pp. 125–147). San Diego, USA.

[Cranefield, 2007] Cranefield, S. (2007). Modelling and monitoring social expectations in multi-agent systems. In P. Noriega, J. Vázquez-Salceda, G. Boella, O. Boissier, V. Dignum, N. Fornara, & E. Matson (Eds.), *Coordination, organizations, institutions, and norms in agent systems II*, volume 4386 of *LNCS* (pp. 308–321). : Springer.

[Cremer & Joseph Kearney, 1996] Cremer, J. & Joseph Kearney, P. W. (1996). A directable vehicle behavior model for virtual driving environment. In *AI, Simulation, and Planning in High Autonomy Systems* (pp. 18–25). La Jolla, USA.

[Cremer et al., 1995] Cremer, J., Kearney, J., & Papelis, Y. (1995). HCSM : A framework for behavior and scenario control in virtual environments. *ACM Transactions on Modeling and Computer Simulation*, 5(3), 242–267.

[Dagdelen et al., 2004] Dagdelen, M., Reymond, G., Kemeny, A., Bordier, M., & Maïzi, N. (2004). MPC based motion cueing algorithm : Development and application to the ULTIMATE driving simulator. In *Driving Simulation Conference* (pp. 221–233). Paris, France.

[Danech-Pajouh, 2003] Danech-Pajouh, M. (2003). Les modèles de prévision du dispositif Bison futé et leur évolution. *Recherche Transport Sécurité*, 78, 1–20.

[Darby, 2004] Darby, A. (2004). Racing vehicle control using insect intelligence. In S. Rabin (Ed.), *AI Game Programming Wisdom 2* (pp. 469–484). : Charles River Media.

[Deborne et al., 2008] Deborne, R., Barthou, A., Toffin, D., Reymond, G., & Kemeny, A. (2008). Simulation study of driver stress and performance to an unexpected steering critical event. In *Driving Simulation Conference* (pp. 111–120). Monaco.

[Demazeau, 1995] Demazeau, Y. (1995). From interactions to collective behaviour in agent based systems. In *European Conference on Cognitive Science* (pp. 117–132). Saint Malo, France.

[Dewar, 2002] Dewar, R. E. (2002). Individual differences. In R. Dewar & P. Olson (Eds.), *Human Factors in Traffic Safety* (pp. 111–142). : Lawyers & Judges Publishing.

[Dignum, 1999] Dignum, F. (1999). Autonomous agents with norms. *Artificial Intelligence and Law*, 7(1), 69–79.

[Dignum et al., 2000] Dignum, F., Morley, D., Sonenberg, L. S., & Cavedon, L. (2000). Towards socially sophisticated BDI agents. In S. Kraus, H. Nakashima, & M. Tambe (Eds.), *International Conference on Multi-Agent Systems* (pp. 111–118). Boston, USA : IEEE Press.

[Doniec et al., 2008] Doniec, A., Mandiau, R., Kolski, S. P., & Espié, S. (2008). Anticipation based on constraint processing in a multi-agent context. *Autonomous Agents and Multi-Agent Systems*, 17(2), 339–361.

[Donikian, 1997] Donikian, S. (1997). VUEMS : a virtual urban environment modeling system. In *Computer Graphics International* (pp. 84–92). Hasselt-Diepenbeek, Belgique.

[Dorigo et al., 1996] Dorigo, M., Maniezzo, V., & Colorni, A. (1996.). Ant system : Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26, 29–41.

[Dresner & Stone, 2007] Dresner, K. & Stone, P. (2007). Sharing the road : Autonomous vehicles meet human drivers. In *Int. Joint Conf. on Artificial Intelligence* (pp. 1263–1268). Hyderabad, India.

[Dresner & Stone, 2008] Dresner, K. & Stone, P. (2008). A multiagent approach to automonous intersection management. *Journal of Artificial Intelligence Research*, 31, 591–656.

[Drogoul, 1993] Drogoul, A. (1993). *De la simulation multi-agent à la résolution collective de problèmes. Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents.* PhD thesis, Université de Paris VI, France.

[Duncan, 1995] Duncan, G. I. (1995). PARAMICS wide area microscopic simulation of ATT and traffic management. In *28th ISATA Conference* Stuttgart, Germany.

[Dupuis & Grezlikowski, 2006] Dupuis, M. & Grezlikowski, H. (2006). OpenDRIVE® - an open standard for the description of roads in driving simulations. In *Driving Simulation Conference* (pp. 25–36). Paris, France.

[Elvik, 2004] Elvik, R. (2004). To what extend can theory account for the findings of road safety evaluation studies ? *Accident Analysis and Prevention*, 36, 841–849.

[Espié et al., 1994] Espié, S., Saad, F., Schnetzler, B., Bourlier, F., & Djemane, N. (1994). Microscopic traffic simulation and driver behaviour modelling : the ARCHISIM project. In *Road Safety in Europe and Strategic Highway Research Program* (pp. 22–31).

[Esser & Schreckenberg, 1997] Esser, J. & Schreckenberg, M. (1997). Microscopic simulation of urban traffic based on cellular automata. *International Journal of Modern Physics C*, 8(5), 1025–1036.

[Esteva et al., 2002] Esteva, M., Padget, J., & Sierra, C. (2002). Formalizing a language for institutions and norms. In J.-J. C. Meyer & M. Tambe (Eds.), *Intelligent Agents VIII*, volume 2333 of *LNCS* (pp. 348–366). : Springer.

[Fellendorf & Vortisch, 2001] Fellendorf, M. & Vortisch, P. (2001). Validation of the microscopic traffic flow model VISSIM in different real-world situations. In *Transportation Research Board* Washington D.C., USA.

[Ferber, 1995] Ferber, J. (1995). *Les Systèmes Multi-Agents.* InterEditions.

[Ferber et al., 2004] Ferber, J., Gutknecht, O., & Michel, F. (2004). From agents to organizations : an organizational view of multi-agent systems. In P. Giorgini, J. Müller, & J. Odell (Eds.), *Agent-Oriented Software Engineering IV* (pp. 214–230). : Springer.

[FHWA, 2009] FHWA (2009). NGSIM home page. `http://www.ngsim.fhwa.dot.gov/`, consulté le 15/04/2009.

[Ficheux, 2005] Ficheux, P. (2005). *Linux embarqué*. Eyrolles.

[FIPA, 1996] FIPA (1996). Foundation for intelligent physical agents home page. `http://www.fipa.org/`, consulté le 20/06/2009.

[Fishbein & Ajzen, 1975] Fishbein, M. & Ajzen, I. (1975). *Belief, Attitude, Intention and Behaviour : An Introduction to Theory and Research*. Addison-Wesley.

[Fritzsche, 1994] Fritzsche, H.-T. (1994). A model for traffic simulation. *Transportation Engineering and Control*, 35(5), 317–321.

[Fuller, 1984] Fuller, R. (1984). A conceptualization of driving behaviour as a threat avoidance. *Ergonomics*, 27(11), 1139–1155.

[Funge, 2004] Funge, J. (2004). *Artificial Intelligence for Computer Games : An Introduction*. A .K. Peters.

[Gallant, 1993] Gallant, S. I. (1993). *Neural Network Learning and Expert Systems*. MIT Press.

[Gameindustry.biz, 2006] Gameindustry.biz (2006). Electronics arts montreal director's interview. `http://www.gamesindustry.biz/articles/cost-of-games-is-crazy-says-ea-montreal-boss`, consulté le 5/05/2009.

[Gameindustry.biz, 2008] Gameindustry.biz (2008). Electronics arts montreal director's interview. `http://www.gamesindustry.biz/articles/riccitiello-we-were-torturing-vancouver-studio`, consulté le 5/05/2009.

[García-Camino et al., 2005] García-Camino, A., Noriega, P., & Rodríguez-Aguilar, J. A. (2005). Implementing norms in electronic institutions. In *Int. Joint Conf. on Autonomous Agents and MultiAgent Systems* (pp. 667–673). Utrecht, Netherlands.

[García-Camino et al., 2009] García-Camino, A., Rodríguez-Aguilar, J. A., Sierra, C., & Vasconcelos, W. (2009). Constraint rule-based programming of norms for electronic institutions. *Autonomous Agents and Multi-Agent Systems*, 18(1), 186–217.

[Gazis et al., 1961] Gazis, D. C., Herman, R., & Rothery, R. W. (1961). Nonlinear follow the leader models of traffic flow. *Operations Research*, 9(4), 545–567.

[Georgé & Gleizes, 2005] Georgé, J.-P. & Gleizes, M.-P. (2005). Emergent programming feasibility study using self-organizing intruction-agents. In *Int. Joint Conf. on Autonomous Agents and MultiAgent Systems* (pp. 1139–1140). Utrecht, Netherlands : ACM.

[Gettman & Head, 2003] Gettman, D. & Head, L. (2003). *Surrogate Safety Measures From Traffic Simulation Models*. Technical Report FHWA-RD-03-050, U.S. Department of Transportation, Federal Highway Administration.

[Gipps, 1981] Gipps, P. G. (1981). A behavioural car-following model for computer simulation. *Transportation Research Part B*, 15, 105–111.

[Gipps, 1986] Gipps, P. G. (1986). A model for the structure of lane-changing decisions. *Transportation Research Part B*, 20(5), 403–414.

[Grossi, 2007] Grossi, D. (2007). *Designing Invisible Handcuffs. Formal Investigations in Institutions and Organizations for Multi-Agent Systems*. PhD thesis, Utrecht University, Netherlands.

[Hadouaj et al., 2000] Hadouaj, S. E., Drogoul, A., & Espié, S. (2000). How to combine reactivity and anticipation : the case of conflicts resolution in a simulated road traffic. In *Multi-Agent Based Simulation* (pp. 82–96). Boston, USA.

[Hadouaj et al., 2004] Hadouaj, S. E., Espié, S., & Drogoul, A. (2004). A multi-agent road traffic simulation model : Validation of the insertion case. In *Summer Computer Simulation Conference* (pp. 20–25). San Jose, USA.

[Haj-Salem et al., 1998] Haj-Salem, H., Elloumi, N., & Papageorgiou, M. (1998). METACOR : A dynamic macroscopic modelling tool for urban corridor. In *IMACS/IEEE Multiconference on Computational Engineering in Systems Applications*, volume 3 (pp. 207–212). Nabeul-Hammamet, Tunisia.

[Hancock, 1999] Hancock, P. (1999). Is car following the real question - Are equations the answer ? *Tranportation Research Part F*, 2, 197–199.

[Hübner et al., 2009] Hübner, J., Boissier, O., Kitio, R., & Ricci, A. (2009). Instrumenting multi-agent organisations with organisational artifacts and agents : "giving the organisational power back to the agents". *Autonomous Agents and Multi-Agent Systems*.

[Hübner et al., 2002] Hübner, J., Sichman, J., & Boissier, O. (2002). Moise+ : Towards a structural, functional, and deontic model for MAS organization. In *Int. Joint Conf. on Autonomous Agents and MultiAgent Systems* (pp. 501–502). Bologna, Italy : ACM.

[Helbing et al., 2000] Helbing, D., Farkas, I., & Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature*, 407, 487–490.

[Helbing & Molnar, 1995] Helbing, D. & Molnar, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, 51, 4282–4286.

[Hidas & Wagner, 2004] Hidas, P. & Wagner, P. (2004). Review of data collection methods for microscopic traffic simulation. In *World Conference on Transport Research* Istanbul, Turkey.

[Hohfeld, 1913] Hohfeld, W. N. (1913). Fundamental legal conceptions as applied in judicial reasoning. *Yale Law Journal*.

[Hourdakis & Michalopoulos, 2002] Hourdakis, J. & Michalopoulos, P. G. (2002). Evaluation of ramp control effectiveness in two twin cities freeways. In *Transportation Research Board* (pp. 21–29). Washington D.C., USA.

[Hourdakis et al., 2003] Hourdakis, J., Michalopoulos, P. G., & Kottommannil, J. (2003). A practical procedure for calibrating microscopic traffic simulation models. In *Transportation Research Board* (pp. 130–139). Washington D.C., USA.

[Hughes, 1998] Hughes, J. (1998). Intensive traffic data collection for simulation of congested auckland motorway. In *19th ARRB Transport Research Conference* Sydney, Australia.

[Jayakrishnan et al., 1994] Jayakrishnan, R., Mahmassani, H. S., & Hu, T. Y. (1994). An evaluation tool for an advanced traffic information and management systems in urban networks. *Transportation Research Part C*, 2, 129–147.

[Jones & Sergot, 1993] Jones, A. & Sergot, M. (1993). On the characterisation of law and computer systems : The normative systems perspective. In J. J. C. Meyer & R. J. Wieringa (Eds.), *Deontic Logic in Computer Science : Normative System Specification* (pp. 275–307). : John Wiley & Sons.

[Kanger, 1971] Kanger, S. (1971). New foundations for ethical theory. In R. Hiplinen (Ed.), *Deontic Logic* (pp. 36–58). : D. Reidel Publishing Company.

[Kearney et al., 2006] Kearney, J., Grechkin, T., Cremer, J., & Plamert, J. (2006). Traffic generation for studies of gap acceptance. In *Driving Simulation Conference* (pp. 177–190). Paris, France.

[Kemeny, 1993] Kemeny, A. (1993). A cooperative driving simulator. In *International Training and Equipment Conference* (pp. 67–71). London, UK.

[Kemeny & Panerai, 2003] Kemeny, A. & Panerai, F. (2003). Evaluating perception in driving simulation experiments. *Trends in Cognitive Sciences*, 7(1), 31–376.

[Keskinen et al., 2004] Keskinen, E., Hatakka, M., Laapotti, S., Katila, A., & Peraaho, M. (2004). Driver behaviour as a hierarchical system. In T. Rothengatter & R. Huguenin (Eds.), *Traffic and Transport Psychology* (pp. 9–29). : Elsevier.

[Kohonen, 1995] Kohonen, T. (1995). *Self-Organizing Maps*. Springer.

[Kubera et al., 2008a] Kubera, Y., Mathieu, P., & Picault, S. (2008a). Formalisation et implémentation des interactions pour la simulation centrée individu. *L'objet*, 14(1-2), 9–33.

[Kubera et al., 2008b] Kubera, Y., Mathieu, P., & Picault, S. (2008b). Interaction-oriented agent simulations : From theory to implementation. In M. Ghallab, C. Spyropoulos, N. Fakotakis, & N. Avouris (Eds.), *European Conference on Artificial Intelligence* (pp. 383–387). : IOS Press.

[Lacroix et al., 2008a] Lacroix, B., Mathieu, P., & Kemeny, A. (2008a). A normative model for behavioral differentiation. In *IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology* (pp. 96–99). Sydney, Australia : IEEE Press.

[Lacroix et al., 2008b] Lacroix, B., Mathieu, P., & Kemeny, A. (2008b). The use of norms violations to model agents behavioral variety. In J. Hübner, E. Matson, O. Boissier, & V. Dignum (Eds.), *Coordination, Organizations, Institutions and Norms in Agent Systems IV*, volume 5428 of *LNCS* (pp. 220–234). : Springer.

[Lacroix et al., 2009a] Lacroix, B., Mathieu, P., & Kemeny, A. (2009a). Generating various and consistent behaviors in simulations. In Y. Demazeau, J. Pavon, J. Corchado, & J. Bajo (Eds.), *Int. Conf. on Practical Applications of Agents and Multi-Agent Systems*, volume 55 of *Advances in Intelligent and Soft Computing* (pp. 110–119). : Springer.

[Lacroix et al., 2007] Lacroix, B., Mathieu, P., Rouelle, V., Chaplier, J., Gallée, G., & Kemeny, A. (2007). Towards traffic generation with individual driver behavior model based vehicles. In *Driving Simulation Conference* (pp. 144–154). Iowa City, USA.

[Lacroix et al., 2009b] Lacroix, B., Rouelle, V., Kemeny, A., Mathieu, P., Laurent, N., Millet, G., & Gallée, G. (2009b). Informal rules for autonomous vehicles in SCANeR™. In *Driving Simulation Conference* (pp. 58–69). Monaco.

[Lamarche & Donikian, 2004] Lamarche, F. & Donikian, S. (2004). Crowd of virtual humans : a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum*, 23(3), 509–518.

[Lefebvre et al., 2006] Lefebvre, T., Kemeny, A., Arquès, D., & Michelin, S. (2006). Speed and motion cues provided by temporal anti-aliasing in driving simulator. In *Driving Simulation Conference Asia* (pp. 211–219). Tsukuba, Japan.

[Legion, 2009] Legion (2009). Legion home page. `http://www.legion.com`, consulté le 15/05/2009.

[Letirand & Delhomme, 2005] Letirand, F. & Delhomme, P. (2005). Speed behaviour as a choice between observing and exceeding the speed limit. *Transportation Research Part F*, 8, 481–492.

[Lindhal, 1977] Lindhal, L. (1977). *Position and Change : a Study in Law and Logic*. Dordrecht, Holland : D. Reidel Publishing Company.

[Macret, 2009] Macret, M. (2009). *Génération de comportements pour agents virtuels*. Technical report, Rapport de projet Impact, École Centrale de Lille.

[Maim, 2009] Maim, J. (2009). *Generating, Animating, and Rendering Varied Individuals for Real-Time Crowds*. PhD thesis, École Polytechnique Fédérale de Lausanne, Suisse.

[Mandiau et al., 2007] Mandiau, R., Champion, A., Auberlé, J.-M., Espié, S., & Kolski, C. (2007). Behaviour based on decision matrices for a coordination between agents in a urban traffic simulation. *Applied Intelligence*, 28(2), 121–138.

[Mathieu et al., 2001] Mathieu, P., Routier, J.-C., & Urro, P. (2001). Un modèle de simulation agent basé sur les interactions. In *Modèles Formels de l'Interaction* (pp. 407–417).

[Michael et al., 2005] Michael, L., Parkes, D. C., & Pfeffer, A. (2005). Specifying and monitoring market mechanisms using rights and obligations. In *Agent-Mediated Electronic Commerce VI*, volume 3435 of *LNCS* (pp. 188–201). : Springer.

[Michon, 1985] Michon, J. (1985). A critical view of driver behavior models : what do we know, what should we do ? In L. Evans & R. Schwing (Eds.), *Human Behavior and Traffic Safety* (pp. 485–530). Plenum Press.

[Moses & Tennenholtz, 1995] Moses, Y. & Tennenholtz, M. (1995). Artificial social systems. *Computers and Artificial Intelligence*, 14(6), 533–562.

[Musse & Thalmann, 2001] Musse, S. & Thalmann, D. (2001). Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics*, 7, 152–164.

[Naatanen & Summala, 1974] Naatanen, R. & Summala, H. (1974). A model for the role of motivational factors in drivers' decision making. *Accident Analysis and Prevention*, 6, 243–261.

[Nagel & Schreckenberg, 1992] Nagel, K. & Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *J. Phys. I*, 2, 2221–2229.

[Okazaki, 1979] Okazaki, S. (1979). Study of pedestrian movement in architectural space, part 1 : Pedestrian movement by the application on of magnetic models. *Trans. of A.I.J.*, 283, 111–119.

[Oktal, 2007a] Oktal (2007a). Database and Road Tools user manual, release 2.20. SCANeR™ documentation.

[Oktal, 2007b] Oktal (2007b). RND Editor user manual, release 2.20. SCANeR™ documentation.

[Oktal, 2009] Oktal (2009). SCANeR™ Simulation Engine home page. `http://www.scaner2.com`, consulté le 11/03/2009.

[Olstam, 2002] Olstam, J. (2002). Generation and simulation of surrounding vehicles in a driving simulator. In *Driving Simulation Conference* (pp. 167–176). Paris, France.

[Olstam, 2005] Olstam, J. (2005). Simulation of rural road traffic for driving simulators. In *Transportation Research Board* Washington D.C., USA.

[Olstam & Espié, 2007] Olstam, J. & Espié, S. (2007). Combination of autonomous and controlled vehicles in driving simulator scenarios. In *International Conference on Road Safety and Simulation* Rome, Italy.

[Paris et al., 2006] Paris, S., Donikian, S., & Bonvalet, N. (2006). Environmental abstraction and path planning techniques for realistic crowd simulation. *Computer Animation and Virtual Worlds*, 17(3-4), 325–335.

[Pavón et al., 2008] Pavón, R., Díaz, F., & Luzón, V. (2008). A model for parameter setting based on bayesian networks. *Engineering Applications of Artificial Intelligence*, 21, 14–25.

[Picard et al., 2004] Picard, G., Bernon, C., & Gleizes, M.-P. (2004). Cooperative agent model within ADELFE framework : An application to a timetabling problem. In *Int. Joint Conf. on Autonomous Agents and MultiAgent Systems* (pp. 1506–1507). New York, USA : ACM.

[Priez et al., 1998] Priez, A., Brigout, C., Petit, C., & Boulommier, L. (1998). Driver behavior in a throttle off situation. In *Enhanced Safety Vehicles* (pp. 546–551). Windsor, Canada.

[PTV, 2004] PTV (2004). *VISSIM Overview*. Technical report, PTV Planung Transport Verkehr AG, Karlsruhe, Germany.

[Punzo & Simonelli, 2005] Punzo, V. & Simonelli, F. (2005). Analysis and comparison of microscopic traffic flow models with real traffic microscopic data. *Transportation Research Record*, 1934, 53–63.

[Rao & Georgeff, 1995] Rao, A. & Georgeff, M. (1995). BDI agents : From theory to practice. In *International Conference on Multi-Agent Systems* (pp. 111–118). San Francisco, USA : AAAI Press.

[Raz, 1975] Raz, J. (1975). *Practical Reason and Norms*. Hutchinson, London.

[Reymond et al., 2001] Reymond, G., Kemeny, A., Droulez, J., & Berthoz, A. (2001). Role of lateral acceleration in driving : Experiments on a real vehicle and a driving simulator. *Human Factors*, 43(3), 483–495.

[Reynolds, 1987] Reynolds, C. W. (1987). Flocks, herds, and schools : a distributed behavior model. In *14th Annual Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH 1987* (pp. 25–34). : ACM.

[Reynolds, 1999] Reynolds, C. W. (1999). Steering behaviors for autonomous characters. In *Game Developers Conference* (pp. 763–782). San Francisco, USA.

[Reynolds, 2004] Reynolds, C. W. (2004). OpenSteer web site. `http://opensteer.sourceforge.net/`, consulté le 17/03/2009.

[Ruskin & Wang, 2002] Ruskin, H. J. & Wang, R. (2002). Modelling traffic flow at an urban unsignalized intersection. In *International Conference on Computational Science* (pp. 381–390). Amsterdam, Netherlands.

[Russell & Norvig, 1995] Russell, S. & Norvig, P. (1995). *Artificial Intelligence : A Modern Approach*. Prentice Hall.

[Saad, 1992] Saad, F. (1992). Contribution of observation and verbal report techniques to an analysis of road situations and drivers' activity. In *Traffic Transport and Psychology* (pp. 183–192).

[Salvucci, 2001] Salvucci, D. D. (2001). Predicting the effects of in-car interfaces on driver behavior using a cognitive architecture. In *Human Factors in Computing Systems : CHI 2001 Conference Proceedings* (pp. 120–127). New York, USA.

[Salvucci et al., 2001] Salvucci, D. D., Boer, E. R., & Liu, A. (2001). Toward an integrated model of driver behavior in a cognitive architecture. *Transportation Research Record*, 1779, 9–16.

[Saunier, 2005] Saunier, N. (2005). *Incidence de la régulation d'un carrefour à feux sur le risque subi par les usagers - Apprentissage d'indicateurs par sélection de données dans un flux.* PhD thesis, École Nationale Supérieure des Télécommunications, France.

[Savarimuthu et al., 2008] Savarimuthu, B., Cranefield, S., Purvis, M., & Purvis, M. (2008). Role model based mechanism for norm emergence in artificial agent societies. In J. Sichman, J. Padget, S. Ossowski, & P. Noriega (Eds.), *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, volume 4870 of *LNCS* (pp. 203–217). : Springer.

[Schelling, 1972] Schelling, T. (1972). *A process of residential segregation : Neighborhood tipping.* Lexington Books, Lexington.

[Schwartberg & Couch, 2004] Schwartberg, S. & Couch, A. (2004). Experience in implementing an HTTP service closure. In *Large Installation System Administration Conference* (pp. 213–230). Atlanta, USA.

[Shao & Terzopoulos, 2007] Shao, W. & Terzopoulos, D. (2007). Autonomous pedestrians. *Graphical Models*, 69, 256–274.

[Shoham & Tennenholtz, 1995] Shoham, Y. & Tennenholtz, M. (1995). On social laws for artificial agent societies : Off-line design. *Artificial Intelligence*, 73, 231–252.

[Sommer, 1969] Sommer, R. (1969). *Personal Space : The Behavioral Basis of Design.* Prentice Hall.

[Summala, 2005] Summala, H. (2005). Traffic psychology theories : Towards understanding driving behaviour and safety efforts. In G. Underwood (Ed.), *Traffic and Transportation Psychology* (pp. 383–394). : Elsevier.

[Taylor, 2003] Taylor, N. B. (2003). The CONTRAM dynamic traffic assignment model. In *Networks and Spatial Economics*, volume 3 (pp. 297–322). : Springer.

[Teknomo, 2002] Teknomo, K. (2002). *Microscopic Pedestrian Flow Characteristics : Development of an Image Processing Data Collection and Simulation Model.* PhD thesis, Tohoku University, Sendai, Japan.

[Terzopoulos & Rabie, 1997] Terzopoulos, D. & Rabie, T. (1997). Animat vision : Active vision in artificial animals. *Videre : Journal of Computer Vision Research*, 1, 2–19.

[Therborn, 2002] Therborn, G. (2002). Back to norms ! On the scope and dynamics of norms and normative actions. *Current Sociology*, 50(6), 863–880.

[Tinnemeier et al., 2009] Tinnemeier, N., Dastani, M., & Meyer, J.-J. (2009). Roles and norms for programming agent organizations. In *Int. Conf. on Autonomous Agents and Multiagent Systems* (pp. 121–128). Budapest, Hungary : ACM.

[Toffin et al., 2007] Toffin, D., Reymond, G., Kemeny, A., & Droulez, J. (2007). Role of steering wheel feedback on driver performance : Driving simulator and modeling analysis. *Vehicle System Dynamics*, 45(4), 375–388.

[Toledo et al., 2003] Toledo, T., Koutsopoulos, H. N., Davol, A., Ben-Akiva, M. E., Andréasson, W. B. I., Johansson, T., & Lundin, C. (2003). Calibration and validation of microscopic traffic simulation tools - Stockholm case study. *Transportation Research Record*, 1831, 65–75.

[Torres da Silva, 2008] Torres da Silva, V. (2008). From the specification to the implementation of norms : an automatic approach to generate rules from norms to govern the behavior of agents. *Autonomous Agents and Multi-Agent Systems*, 17(1), 113–155.

[TSS, 2009] TSS (2009). AIMSUN home page. `http://www.aimsun.com/`, consulté le 15/04/2009.

[Tuomela, 1995] Tuomela, R. (1995). *The Importance of Us : A Philosophical Study of Basic Social Norms*. Standford University Press.

[Vázquez-Salceda et al., 2004] Vázquez-Salceda, J., Aldewereld, H., & Dignum, F. (2004). Implementing norms in multiagent systems. In *Multiagent Systems Technologies*, volume 3187 of *LNAI* (pp. 313–327). : Springer.

[Vázquez-Salceda et al., 2005] Vázquez-Salceda, J., Dignum, V., & Dignum, F. (2005). Organizing multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 11(3), 307–360.

[Verhagen, 2000] Verhagen, H. (2000). *Norm Autonomous Agents*. PhD thesis, The Royal Institute of Technology and Stockolm University, Sweden.

[Volterra, 1931] Volterra, V. (1931). Variations and fluctuations of the number of individuals in animal species living together. *Animal Ecology*.

[von Wright, 1951] von Wright, G. H. (1951). Deontic logic. *Mind*, 60, 1–15.

[Wang et al., 2005] Wang, H., Kearney, J., Cremer, J., & Willemsen, P. (2005). Steering behaviors for autonomous vehicles in virtual environments. In *IEEE Virtual Reality Conference* (pp. 155–162). Bonn, Germany : IEEE Press.

[Wiedemann, 1974] Wiedemann, R. (1974). *Simulation des Straßenverkehrsflusses*. Technical report, Instituts für Verkehrswesen der Universität Karlsruhe, Germany.

[Wilde, 1982] Wilde, G. (1982). The theory of risk homeostasis : Implications for safety and heath. *Risk Analysis*, 2(4), 209–225.

[Willemsen et al., 2003] Willemsen, P., Kearney, J., & Wang, H. (2003). Ribbon network for modeling navigable paths of autonomous agents in virtual urban environments. In *IEEE Virtual Reality Conference* (pp. 79–86). Los Angeles, USA : IEEE Press.

[Wooldridge, 2002] Wooldridge, M. (2002). *An Introduction to Multi-Agent Systems*. John Wiley & Sons.

[Wright et al., 2002] Wright, S., Ward, N. J., & Cohn, A. G. (2002). Enhanced presence in driving simulators using autonomous traffic with virtual personalities. *Presence*, 11(6), 578–590.

[Wylie et al., 1994] Wylie, B. J., Cameron, G. D., White, M. D., Smith, M. A., & McArthur, D. (1994). *PARAMICS : Parallel Microscopic Traffic Simulator*. Technical report, EPCC-TR94-01, Edinburgh, UK.

[Yang & Koutsopoulos, 1996] Yang, Q. & Koutsopoulos, H. N. (1996). A microscopic traffic simulator for evaluation of dynamic management systems. *Transportation Research Part C*, 4, 113–129.

Table des figures

1.1 Exemple de simulateur de trafic macroscopique : TransCAD 7
1.2 Exemple de simulateur de trafic microscopique : AIMSUN 8
1.3 Architecture d'un simulateur de conduite 9
1.4 Création par scénario d'une situation de trafic accidentogène 11
1.5 Représentation multi-couche de l'environnement dans les simulateurs de conduite 13
1.6 Les zones de perceptions dans l'application Archisim 15
1.7 Architecture d'un modèle de simulation de trafic dans le contexte de la simulation de conduite 17
1.8 Exemples de comportements réactifs de navigation : comportement de recherche, évitement d'obstacle et suivi de chemin 26
1.9 Exemples d'émergence de comportements complexes par aggrégation de comportements simples : poursuite, foule suivant un chemin commun 27
1.10 Exemple de simulation de foules de piétons : l'application Legion 28
1.11 Contraintes sur la configuration d'un serveur web : les contraintes de configuration permettant de répondre correctement à une requète http 29
1.12 Utilisation de *closures* pour réduire la complexité de configuration en administration système 30
1.13 Méta-modèle de génération automatique de paramétrage 31
1.14 Génération de diversité graphique dans la simulation de foule 33
1.15 Choix des paramètres définissant des personnalités virtuelles de conducteurs . . . 35

2.1 L'architecture d'agent BOID 43
2.2 Utilisation de lois sociales pour le déplacement de robots 45
2.3 Gestion d'intersection par des normes adaptatives 46
2.4 Schéma descriptif du modèle proposé dans ce travail 47
2.5 Représentation synthétique du modèle de données proposé 53
2.6 Description de la hiérarchie des normes 54
2.7 Classification de la déviance en fonction des paramètres de la norme 68

3.1 Les réseaux de Kohonen : exemple d'une grille de neurones s'auto-organisant suivant un carré . . . 73
3.2 Exemple de structure d'un réseau de Kohonen . . . 74
3.3 Utilisation du modèle proposé en tant que module externe à la simulation . . . 87
3.4 Applet Java démontrant l'utilisation du modèle pour la génération de caractéristiques d'agents . . . 92

4.1 Le simulateur d'éclairage du Centre Technique de Simulation de Renault . . . 96
4.2 Les simulateurs CARDS© 2 et Ultimate de Renault . . . 96
4.3 Architecture de l'application de simulation de conduite SCANeR™ . . . 97
4.4 Capture d'écran de SCANeR™, illustrant une partie des modules disponibles . . . 98
4.5 Le module de création et d'édition de scénario de SCANeR™ . . . 99
4.6 Rendu visuel 2D du trafic routier dans SCANeR™ . . . 101
4.7 Représentation des bases de données dans l'application SCANeR™ . . . 102
4.8 Chaîne de conception et d'exécution des scénarios . . . 103
4.9 Interactions entre les entités simulées . . . 103
4.10 Architecture des agents . . . 104
4.11 Perception des agents . . . 105
4.12 L'automate à état fini utilisé dans le modèle de décision des agents . . . 106
4.13 Intégration des modules développés dans l'architecture existante . . . 114
4.14 Le module « Traffic Designer » : application des apports du modèle aux scénarios existants . . . 116
4.15 Utilisation de générateurs de trafic pour produire un flux de véhicules dans la simulation . . . 117
4.16 Fonctionnement des générateurs de trafic . . . 118
4.17 Le module « Traffic Tools » : visualisation en temps réels de données statistiques 119
4.18 Le module « Traffic Tools » : visualisation des informations relatives aux générateurs de trafic . . . 119
4.19 Le module « Traffic Monitoring » : classification des données et inférence de normes 121

5.1 La base de données de type autoroute utilisée pour les expérimentations . . . 128
5.2 Distribution des vitesses en fonction de l'ensemble de normes choisies . . . 130
5.3 Répartition des véhicules entres les voies gauche et droite de l'autoroute . . . 131
5.4 Inférence de normes . . . 133
5.5 Expérimentation de la congestion sur autoroute : schéma de la base de données . 134
5.6 Comparaison des flux de trafic dans SCANeR™ et AIMSUN . . . 135
5.7 Expérimentation de l'influence d'une voie d'insertion : schéma de la base de données utilisée . . . 136
5.8 Création d'une base de données dans Evariste à partir de données GIS . . . 141

Liste des tableaux

1.1 Modèle centré interaction IODA : exemple de matrice d'interaction 24

2.1 Exemple de définition de paramètres dans la sémantique du modèle proposé. . . 66

3.1 Illustration du fonctionnement des réseaux de Kohonen : poids des vecteurs des neurones à la 500^{e} itération. 76

3.2 Paramètres définissant les normes qui caractérisent les espèces dans un exemple de simulation utilisant le modèle comme « créateur de créatures ». 91

4.1 Les paramètres pseudo-psychologiques du modèle de décision des agents dans l'application SCANeR™. 108

4.2 Paramètres de la norme racine N_{init} pour l'application du modèle proposé à SCANeR™ 112

4.3 Définition des paramètres de la norme N_{prudent}. 113

4.4 Définition des paramètres de la norme N_{normal}. 113

4.5 Définition des paramètres de la norme N_{agressif}. 113

5.1 Paramètres utilisés pour la norme du jeu *no norms*. 129

5.2 Paramètres utilisés pour la norme du jeu *normal only*. 129

5.3 Vitesse moyenne au niveau du détecteur 2 et temps de parcours moyen sur la section. 131

5.4 Expérimentation de la congestion sur autoroute : nombre de véhicules générés pendant chaque période de 10 minutes. 134

5.5 Expérimentation de l'influence d'une voie d'insertion : nombre de véhicules générés pendant chaque période de 10 minutes 136

Résumé : Dans les simulations centrées individu, la variété et la cohérence du comportement des agents sont des critères importants pour le réalisme et la validité de la simulation. Dans ce travail, nous nous sommes intéressés à la prise en compte simultanée de ces deux éléments. Nous proposons un modèle de différenciation comportementale, qui se décline en un outil dont les principaux apports sont d'être générique, non-intrusif, et de permettre une conception en dehors de l'agent.

Le modèle s'articule selon trois axes. Tout d'abord, il décrit les comportements des agents par des normes. Celles-ci fournissent un profil comportemental à la conception, et un contrôle de la conformité à l'exécution. Ensuite, le processus de génération des comportements permet d'autoriser la création d'agents déviants ou en violation. Il influe pour cela sur le déterminisme du mécanisme. Enfin, les normes peuvent être inférées à partir de simulations enregistrées ou de situations réelles, afin d'analyser les résultats des expérimentations et d'automatiser la configuration du modèle.

Nous avons appliqué cet outil à la simulation de trafic dans SCANeR™, l'application développée et utilisée par Renault pour ses simulateurs de conduite. Les développements réalisés au cours de la thèse introduisent dans le trafic des styles de conduite spécifiés sous forme de normes, par exemple des conducteurs prudents ou agressifs. Ils permettent ensuite de peupler l'environnement de manière automatisée. Au delà de l'amélioration subjective du réalisme, les expérimentations réalisées démontrent les apports de l'outil sur la variété et la représentativité des comportements obtenus.

Mots clés : systèmes multi-agents, comportement, variété, conformité, cohérence, norme, génération, simulation, trafic routier

Abstract : In individual-centered simulations, the variety and consistency of agents' behaviors are important for the realism and validity of the simulation. In this work, we addressed the issue of the simultaneous influence of these two elements. We propose a behavioral differentiation model, which provides the basis for a generic and non-intrusive tool allowing an out-of-the-agent design.

The model involves three dimensions. First, it describes the agents' behaviors using norms. They provide a behavioral pattern during conception, and a compliance reference during execution. Then, the generation process of the behaviors allows the creation of deviant or violating agents, by influencing the determinism of the mechanism. Finally, the norms can be inferred from previous simulations records or real data, providing an analysis tool of the results and allowing automating the model configuration.

We applied the model to the traffic simulation in SCANeR™, the driving simulation software developed and used at Renault. The developments carried out during the thesis introduced driving styles in the traffic (e.g. cautious or aggressive drivers) specified using norms. The use of norms allows populating the environment easily and in an automated way. The behavioral realism of the traffic was improved, and the experimentations show how the model contributes to the variety and the representativeness of the produced behaviors.

Keywords : multi-agent systems, behavior, variety, compliance, consistency, norm, generation, simulation, road traffic