ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE ÉLECTRIQUE
M.Ing.

PAR
DAVID BÉLANGER

MODIFICATION DE LA SITUATION PAR SIMULATION AFIN D'AMÉLIORER LES DÉPLACEMENTS DANS UN CADRE TACTIQUE

MONTRÉAL, LE 7 AOÛT 2006

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE :

M. Richard Lepage, directeur de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

M. Luc Pigeon, codirecteur de mémoire
Scientifique de la défense à R & D pour la défense Canada

M. Robert Sabourin, président du jury
Département de génie de la production automatisée à l'École de technologie supérieure

M. Éric Granger, membre du jury
Département de génie de la production automatisée à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC
LE 7 AOÜT 2006
À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

MODIFICATION DE LA SITUATION PAR SIMULATION AFIN D'AMÉLIORER LES DÉPLACEMENTS DANS UN CADRE TACTIQUE

David Bélanger

SOMMAIRE

Depuis la fin du 20e siècle, les villes sont de plus en plus le théâtre d'opérations militaires. Les milieux urbains étant des environnements extrêmement complexes, le commandement et contrôle y est particulièrement difficile. Un simple déplacement peut devenir problématique étant donné le caractère dynamique de l'environnement. Un trajet prévu initialement peut s'avérer impossible à compléter compte tenu de nouvelles menaces détectées pendant le déplacement. La planification d'un nouveau chemin sécuritaire étant longue à accomplir par un opérateur, l'aide d'un ordinateur serait souhaitable. C'est sur cette problématique que se penche notre travail. Plus particulièrement, il sera question de planification d'actions améliorant un déplacement.

Deux actions seront considérées pour améliorer le déplacement d'un véhicule : l'édification d'un barrage routier et l'interception d'un véhicule menaçant. La première devant nécessairement être simulée pour en déterminer les effets, un simulateur de barrage routier a été développé dans ce travail. L'ensemble du planificateur présenté dans ce mémoire prend la forme d'un système multiagent. Les agents composant celui-ci sont les différentes ressources sur le terrain (véhicules de police, forces alliées...), ainsi que le véhicule dont le déplacement est à améliorer.

Un prototype du planificateur a été développé dans ce mémoire. Dans le but de valider la simulation et la planification résultantes de ce prototype, un graphe de la ville de Québec a été utilisé comme environnement expérimental. De plus, des données de trafic ont été générées à partir d'une matrice origine-destination. Les résultats obtenus pour divers scénarios démontrent le plein potentiel du prototype. Le planificateur repère les segments d'un parcours menacés par un adversaire et assigne des unités pour accomplir des actions réglant ces problèmes.

MODIFICATION OF THE SITUATION BY SIMULATION AIMING THE AMELIORATION OF A DISPLACEMENT IN A TACTIC CONTEXT

David Bélanger

ABSTRACT

Since the 20th century, cities become more than often the theater of military operations. Urban areas are really complex environment; therefore command and control of these areas become especially difficult. A simple movement could be problematic due to the dynamic characteristic of the environment. An initial path could become impossible to complete since new threats were reported during the navigation of that path. The evaluation of selecting a new secure path is complex to achieve by an operator, so computers assistance is required. It's on this specific problematic that we devote our effort. More specifically, we will focus on planning actions to improve movement.

There are two actions that will be considerate to improve the movement of a vehicle: implentation of a roadblock and the interception of an enemy vehicle. The first action must be simulated to determine its effects; a roadblock simulator had been developed in this work. The whole planning engine presented in this master's thesis is implemented as a multi-agent system. Agents which compose the system are resources on the field (police vehicles, allies' forces), and the vehicle which the movement is to be improve.

A prototype of the planning engine had been developed in the research. To validate the simulation and the results of the prototyped planning engine, experimentation had been performed on graph data of Quebec City. Further more, traffic data had been generated from a origin-destination matrix. The results gather from several scenarios shown the fully potential of this prototype. The planning engine finds the segments of a path affected by an enemy and assign units to accomplish action task to resolve the problem.

REMERCIEMENTS

Ce travail n'aurait pu avoir lieu sans le Colonel Gregory Burt qui a eu la gentillesse de me recommander et d'établir le contact entre moi et le Dr Luc Pigeon. Merci à ce dernier et à Alain Bergeron de m'avoir accueilli dans leur équipe de recherche et d'avoir codirigé ce mémoire. Merci énorme au Dr Richard Lepage pour avoir accepté de diriger ce travail. Merci à Richard Carbone et Dany Francis pour l'assistance technique.

Merci à Recherche et Développement pour la Défense Canada d'avoir subventionné ce travail et de m'avoir donné accès à leurs installations.

Un gros remerciement à Sophie pour les nombreuses heures passées à relire un mémoire portant sur un sujet si loin de ses centres d'intérêt. Merci à Pierre et Antonine pour leur soutien. Mélanie…

TABLE DES MATIÈRES

Page

SOMMAIRE i
ABSTRACT ii
REMERCIEMENTS iii
TABLE DES MATIÈRES iv
LISTE DES TABLEAUX vi
LISTE DES FIGURES vii
LISTE DES ABRÉVIATIONS ET DES SIGLES viii
INTRODUCTION 1
CHAPITRE 1 PLANIFICATION D'ACTIONS ET SIMULATION DE TRAFIC : ÉTAT DE L'ART 5
1.1 Algorithme de planification 5
1.1.1 Systèmes utilisant des algorithmes génétiques 6
1.1.2 Systèmes multiagents 8
1.2 Simulation du trafic urbain 9
1.2.1 Matrice origine-destination 10
1.2.2 Assignation du trafic 11
1.2.3 Méthode de saisie de données réelles 13
1.2.4 Simulation d'un barrage routier 15
1.3 Conclusion 15
CHAPITRE 2 RÉSEAU ROUTIER ET GÉNÉRATION DES DONNÉES DE TRAFIC 17
2.1 Réseau routier 17
2.2 Génération de données de trafic 19
2.2.1 Matrice OD 19
2.2.2 Fonction vitesse-véhicules 21
2.2.3 Algorithme d'assignation 23
2.3 Données réelles de la ville de Québec 24
2.4 Conclusion 25
CHAPITRE 3 SIMULATION D'UN BARRAGE ROUTIER 26
3.1 Choix de l'algorithme de simulation 26
3.2 Description de l'algorithme de simulation 27
3.2.1 Estimation de la matrice OD 27
3.2.2 Soustraction des véhicules 30

3.2.3 Réassignation 31
3.3 Réalisation du simulateur 33
3.4 Conclusion 33
CHAPITRE 4 PLANIFICATION D'ACTIONS 35
4.1 Planificateur de haut niveau 35
4.2 Planificateur de bas niveau 37
4.3 Structure du module OptiEvents 39
4.3.1 Observateur d'environnement 40
4.3.2 Unificateur de plans 42
4.3.3 Communication 46
4.4 Fonctionnement complet du module OptiEvents 46
4.5 Implémentation du module OptiEvents 48
4.6 Conclusion 50
CHAPITRE 5 RÉSULTATS ET DISCUSSION 51
5.1 Génération des données de trafic 51
5.2 Simulateurs de barrage routier 55
5.3 Planificateur 61
5.4 Conclusion 67
CHAPITRE 6 CONCLUSION ET RECOMMANDATIONS 69
ANNEXE 1 Implémentation JAVA de la génération de trafic automobile 73
ANNEXE 2 Implémentation JAVA de la simulation d'un barrage routier 77
ANNEXE 3 Implémentation JAVA du module OptiEvents 79
ANNEXE 4 Algorithme de visualisation 83
BIBLIOGRAPHIE 85

LISTE DES TABLEAUX

Page

Tableau I	Exemple de matrice origine-destination	10
Tableau II	Valeur Q et JA pour les différents types de route	22
Tableau III	Correspondance entre la classification routière 4 dans le graphe et les types de route selon Akcelik	22
Tableau IV	Erreur relative entre les résultats de la simulation de trafic et les données réelles de la ville de Québec	53

LISTE DES FIGURES

Page

Figure 1 Vue d'ensemble du projet Scipio. 2
Figure 2 Méthode de représentation des nombres. 12
Figure 3 Schéma de l'algorithme d'assignation. 24
Figure 4 Fonction floue utilisée pour représenter la perception des automobilistes 29
Figure 5 Résultat de l'assignation. 32
Figure 6 Schéma de l'algorithme du simulateur de barrage routier 33
Figure 7 Structure générale du module *OptiEvents*. 40
Figure 8 Fusion de plans. 44
Figure 9 Schéma algorithmique du module *OptiEvents*. 49
Figure 10 Chargement du réseau automobile. 52
Figure 11 Région de simulation. 56
Figure 12 Agrandissement de l'endroit où a lieu le barrage routier. 56
Figure 13 Temps 1. Visualisation du trafic avant la formation du barrage routier... 57
Figure 14 Temps 2. Visualisation du trafic après la formation du barrage routier... 58
Figure 15 Visualisation de la TTDM, avant l'application du barrage routier. 59
Figure 16 Visualisation de la TTDM, après l'application du barrage routier 60
Figure 17 Premier scénario : une *menace* et deux *ressources*. 64
Figure 18 Second scénario : deux *menaces* aperçues à différents moments et trois *ressources*. 67
Figure 19 Schéma des classes utilisées dans l'implémentation de l'assignation du trafic automobile 75
Figure 20 Schéma des classes utilisées dans l'implémentation du simulateur de barrage routier. 78
Figure 21 Schéma des classes utilisées dans le module *OptiEvents*. 82

LISTE DES ABRÉVIATIONS ET DES SIGLES

RDDC	Recherche et développement pour la défense Canada
MTQ	Ministère des Transports du Québec
UE	*User equilibrium* (Équilibre des usagers)
FUZ	*Fuzzy user equilibrium* (Équilibre flou des usagers)
SUE	*Stochastic user equilibrium* (Équilibre stochastique des usagers)
HUE	*Hybrid user equilibrium* (Équilibre hybride des usagers)
MRC	Municipalité régionale de comté
OD	Origine destination
PHN	Planificateur de haut niveau
PBN	Planificateur de bas niveau
TP	Tronçons problématiques
OE	Observateur d'environnement
UP	Unificateur de plans

INTRODUCTION

Les opérations militaires contemporaines se déroulent en grande partie dans des milieux urbains. Dans cet environnement complexe et hautement dynamique, le commandement et contrôle y est très difficile. À ce titre, pensons à l'opération américaine à Mogadishu (Somalie, 1993), où les soldats avaient d'énormes difficultés à se déplacer dans un réseau routier perturbé par différents groupes ennemis. Ce type d'opération pratiquement désastreuse a amené le milieu scientifique militaire à concevoir des outils d'aide au commandement. Bien que ce sujet de recherche soit diversifié, un volet important retiendra notre attention dans le cadre de ce travail : la génération de plans.

Ces thématiques revêtent d'ailleurs un intérêt particulier pour l'équipe de recherche parrainant ce mémoire, laquelle travaille depuis quelques années en ce sens. Ce groupe de chercheurs, travaillant pour Recherche et développement pour la défense Canada (RDDC), a pour principal objectif de concevoir un système de commandement et contrôle en milieu urbain qui serait, dans l'avenir, destiné aux Forces armées canadiennes. Un tel système comporte plusieurs modules qui sont illustrés à la figure 1.

Le cœur de ce projet est le module *A.I.Think*. Il est responsable de la base de données et de l'apprentissage du système. Quant à *UrbanWish*, il est en quelque sorte l'interface entre la machine et l'utilisateur. Comme son nom l'indique en anglais, il reçoit les « souhaits » du commandant. *CounterDeception*, pour sa part, servira à valider l'information du système alors qu'*UrbanDispatch* sera responsable de recevoir les informations des différents capteurs sur le terrain et d'afficher le champ de bataille. *Optipath* doit trouver, à la demande de l'utilisateur, le chemin le plus court entre deux points sur un réseau routier. Finalement, *OptiEvents* est responsable de trouver des actions susceptibles d'améliorer les chemins trouvés par *Optipath*.

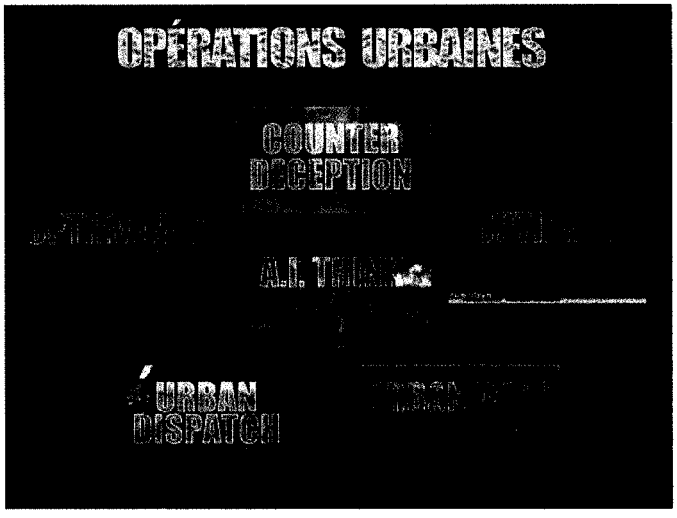


Figure 1 Vue d'ensemble du projet Scipio.

Les six modules structurant le projet Scipio : *A.I.Think* est responsable de la base de données et de l'apprentissage du système, *UrbanWish* est l'interface machine-utilisateur, *CounterDeception* valide l'information, *UrbanDispatch* reçoit les informations, *Optipath* trouve des chemins dans un graphe et *OptiEvents* trouve des actions qui modifient la situation.

Dans le cadre de ce mémoire, les deux derniers modules sont davantage pertinents. *Optipath* est à ce jour le module le plus développé. Plus complexe que les systèmes retrouvés dans certaines voitures de luxe, lorsqu'il recherche un chemin le plus court, *Optipath* tient compte autant du type de véhicule utilisé que des trois facteurs caractérisant les tronçons du réseau :

a. la distance;
b. le temps de parcours (nécessite la notion de trafic automobile);
c. la sécurité (présence de menace).

L'innovation proposée par l'équipe de RDDC réside surtout dans les deux derniers facteurs (temps de parcours et sécurité). Le commandant utilisant le système indique quelles sont ses préférences à l'égard des trois facteurs précédemment nommés. Ainsi, il peut choisir le chemin le plus rapide, le plus court, le plus sécuritaire ou un compromis entre ces trois facteurs.

La recherche du chemin le plus court a, jusqu'à présent, considérée l'environnement tel qu'il est, sans l'altérer. À cet égard, le module *OptiEvents* devient intéressant. Son objectif est de trouver des actions susceptibles de modifier la situation (environnement) afin d'améliorer la pondération des différents facteurs du réseau pour ainsi minimiser le coût du déplacement. De plus, ces améliorations doivent être planifiées dans un délai court (voir en temps réel) afin de faire une utilisation opportune des ressources sur le terrain. Ceci étant, cet ouvrage se concentrera donc sur ce module.

L'objectif principal de ce mémoire est de concevoir une architecture pour le module *OptiEvents*. Un prototype basé sur cette architecture doit démontrer la faisabilité du module sans pour autant être sa version opérationnelle. En effet, étant donné le temps et les moyens limités relatifs au programme de maîtrise, seules deux actions seront développées. L'architecture devra cependant être facilement modifiable pour éventuellement intégrer davantage d'action.

Dans un premier temps, une revue de la littérature a été effectuée. Dans un second temps, afin de prouver la faisabilité et l'efficacité du module *OptiEvents*, une première action a été développée, soit un barrage routier. Une seconde action, plus simple et plus facile à implémenter dans un système en temps réel, a été par la suite étudiée : l'interception d'un véhicule menaçant. Finalement, un planificateur utilisant ces actions a été développé. Des résultats concluants, découlant de ces actions, pourraient nous permettre de formuler l'hypothèse voulant que davantage d'actions amélioreront, dans l'avenir, les plans engendrés par le prototype actuel. D'un point de vue méthodologique, notons que les algorithmes seront, dans le cadre de ce mémoire, programmés en JAVA, langage utilisé dans le système existant.

De manière plus précise, la revue de littérature sera exposée au premier chapitre alors qu'un deuxième se consacrera à l'environnement d'expérimentation. Il sera alors question du réseau routier et de la méthode utilisée pour générer les valeurs de trafic. Le

simulateur de barrage routier retiendra notre attention au chapitre 3. Quant à la planification des actions et aux effets de la seconde action (l'interception d'un véhicule menaçant), ils seront discutées dans le chapitre suivant. Finalement, un cinquième chapitre présentera les résultats obtenus lesquels seront par la suite discutés.

CHAPITRE 1

PLANIFICATION D'ACTIONS ET SIMULATION DE TRAFIC : ÉTAT DE L'ART

Le présent chapitre consiste en un état de l'art des principaux thèmes qui seront abordés dans ce travail. Différents travaux portant sur la planification d'actions seront présentés. Par la suite, il sera question de la simulation de trafic. Deux concepts seront ainsi développés : matrice origine-destination et assignation du trafic. Finalement, un simulateur de barrage routier sera discuté.

1.1 Algorithme de planification

Le cœur de ce mémoire consiste en la réalisation d'un algorithme de planification. Il s'agit de coordonner une ou plusieurs actions permettant d'améliorer un déplacement routier. Plusieurs méthodes existent dans le domaine de l'intelligence artificielle pour arriver à cette fin [1-5]. Toutefois, dans le cadre de ce mémoire, la problématique observée diffère des situations habituelles où l'on fait appel à l'intelligence artificielle et ce, à plusieurs égards :

a. la situation est dynamique : posées au même endroit et à deux moments distincts, deux actions identiques auront possiblement des résultats différents;
b. les résultats des actions posées ne sont pas bien définis : la majorité des actions doivent être simulées afin de calculer leurs effets sur le réseau;
c. le but n'est pas bien défini : bien que les points de départ et d'arrivée soient clairs, le chemin emprunté ne l'est pas. Il variera selon les résultats des actions.

Ceci étant, la majorité des planificateurs conventionnels ne sont pas applicapbles dans le contexte de la recherche. Néanmoins, certaines méthodes documentées dans la littérature

ont retenu notre attention puisqu'elles semblent mieux adaptées aux trois contraintes précédemment citées.

1.1.1 Systèmes utilisant des algorithmes génétiques

Un algorithme génétique est une technique utilisée pour trouver des solutions à certains problèmes d'optimisation. Elle utilise des techniques inspirées de la sélection naturelle telles que la *sélection*, la *recombinaison* et la *mutation*. Chacune des solutions peut être représentée par un chromosome dans lequel est codée une solution du problème. À partir d'une population initiale de chromosomes, on sélectionne les meilleures solutions sur lesquelles sont effectuées les opérations de *recombinaison* et de *mutation*. Ces opérations fournissent une nouvelle génération de chromosome qui sera à son tour sujet aux trois opérations précédentes. Répétant ces étapes à plusieurs reprises, le principe de sélection naturelle est imité, trouvant de génération en génération, de meilleures solutions. Un mécanisme de niche peut être utilisé afin d'assurer que la population recherche dans différents espaces de solution.

Dans le domaine militaire, un planificateur du nom de FOX-GA a été décrit par Hayes et al. [6]. Conçu pour assister les services de renseignements militaires, il contribue à la prise de décision sur un champ de bataille conventionnel (en campagne). En d'autres mots, il élabore des plans de bataille. FOX-GA a l'avantage d'offrir au commandant en place un plus grand nombre de plans alternatifs que les méthodes traditionnelles dans ce domaine : soit jusqu'à une quinzaine de plans au lieu des trois usuels (axe du centre, de gauche et de droite). Sa capacité de générer un grand nombre de plans jugés adéquats dans un court délai représente donc l'une des forces de ce planificateur. Un algorithme génétique est nécessaire à FOX-GA pour obtenir un tel résultat. L'algorithme contient un mécanisme de niche afin d'assurer la diversité des plans proposés. L'algorithme permet d'éliminer certaines combinaisons illégales, c'est-à-dire donnant des plans

impossibles, en réparant les chromosomes erronés suivant des contraintes de faisabilités [7]. La justesse des plans est évaluée au moyen de simulations effectuées contre six plans ennemis créés par un expert. Ceux-ci n'étant pas évolutifs, le planificateur trouvera la faille dans les six plans et l'exploitera. Ceci constitue un handicap puisque l'ennemi n'effectuera pas nécessairement l'un des six plans suggérés.

Le générateur de plan co-évolutionnaire proposé par Suantak et al. [8] vient pallier à cette faiblesse. Contrairement au modèle proposé précédemment, il permet une évolution à la fois des plans amis et ennemis. Ainsi, deux systèmes utilisant chacun un algorithme génétique s'opposent, générant tour à tour un meilleur plan exploitant les failles de son opposant.

Une autre approche intéressante tend à utiliser un algorithme génétique combiné à un système d'inférence flou [9]. Le planificateur s'intéresse davantage au caractère multi-objectif des plans de bataille. L'implémentation floue permet au commandant de considérer graduellement les objectifs, en ajoutant un nouvel objectif lorsqu'il est satisfait du précédent. Les auteurs ont également eu recours à un système co-évolutionnaire. Par contre, cette technique fait davantage appel à la participation d'un opérateur.

Dans l'ensemble des méthodes utilisant un algorithme génétique citées plus haut, la clé réside dans le codage des actions. Ces actions sont toujours limitées et leur application est possible dans un nombre d'endroits restreint. Par exemple, dans le cas d'un générateur de plan de bataille, le terrain est séparé en trois axes. Ainsi, il ne reste qu'à déterminer quelles unités opéreront dans chaque axe, limitant ainsi le choix à trois endroits. De plus, le résultat de ces actions est invariable dans le temps. Dans le cadre actuel du projet, les actions et les lieux sont beaucoup plus variés. Chacune des unités peut effectuer un grand nombre d'actions dans une multitude de points de décision

(chacune des rues d'une ville) à des moments différents. Une autre méthode doit donc être empruntée puisque le chromosome codé serait trop volumineux.

1.1.2 Systèmes multiagents

La littérature fait état d'une méthode de planification de plus en plus utilisée pour résoudre des problèmes complexes. Il s'agit des systèmes multiagents. Comme son nom l'indique, un système multiagent est composé de plusieurs d'agents autonomes interagissant entre eux dans le même environnement. Un agent peut être un robot, un programme informatique, un processus ou encore un être humain. L'important est que tous puissent communiquer entre eux efficacement. Lorsque les agents coopèrent entre eux, ils forment un système capable de formuler des plans dans des environnements très complexes. En d'autres mots, si plusieurs agents formulent des plans pour eux-mêmes, ils peuvent par la suite les partager et les coordonner afin de créer un plan global de qualité. Certains auteurs [10-12] ont qualifié un tel processus de *coordination* ou *synergie*. Ce type d'architecture apparaît particulièrement intéressant dans le cas qui nous préoccupe puisqu'il permettrait de coordonner plusieurs unités.

Afin de maximiser la coordination entre les agents et plus particulièrement la fusion de plans venant de deux ou de plusieurs agents, une *hiérarchisation* des plans est couramment utilisée [10;13;14]. Lorsque qu'un plan effectué par un agent donne des résultats supérieurs ou semblables au plan venant d'un deuxième agent et que les coûts sont moindres, ces deux plans peuvent être fusionnés. Ainsi, le meilleur agent effectuera le plan et le second agent pourra être libéré de ce travail et ainsi remplir d'autres tâches. Sachant qu'un plan est constitué d'une série d'actions dont le nombre est parfois élevé, le plan devient difficile à définir et, par conséquent, à fusionner. Par contre, en utilisant une représentation à plusieurs niveaux d'abstraction (ou *représentation hiérarchique*), il est possible de travailler avec les informations les plus pertinentes et de descendre, au

besoin, dans la hiérarchie (vers le niveau le moins abstrait). Cette technique facilite grandement le calcul.

Un point commun remarqué dans l'ensemble des travaux précédents est la nécessité d'implémenter un simulateur de résultats. En effet, contrairement aux planificateurs classiques [15], la complexité des situations et leurs caractères dynamiques font qu'il est impossible d'anticiper le résultat des actions. Celles-ci doivent donc être simulées.

Les systèmes multiagents sont très prometteurs pour résoudre le problème dont fait objet ce travail. Les travaux recensés ont montré l'efficacité de ce type d'architecture dans des environnements complexes et dynamiques tel le nôtre. De plus, la nécessité de fusionner plusieurs plans simples provenant de différentes unités en un plan global se prête très bien au système multiagents.

1.2 Simulation du trafic urbain

Pour simuler le trafic d'un réseau routier, deux grandes catégories de simulation se distinguent : la simulation macroscopique et la simulation microscopique. Comme leur nom l'indique, ces deux techniques se distinguent par leur niveau d'abstraction. Lors d'une simulation microscopique, toutes les actions de chacun des véhicules sont simulées [16;17]. Pour ce faire, le simulateur fait appel à une panoplie de formules mathématiques représentant diverses manœuvres allant du freinage au changement de ligne. Bien que cette technique soit d'une précision élevée, elle demande un calcul très complexe pouvant poser des problèmes dans un système en temps réel. Elle est de ce fait peu intéressante dans le cadre de cette recherche. Pour sa part, la simulation macroscopique ne s'avère pas aussi précise, seulement, elle demande moins de temps quant au calcul à effectuer. Elle requiert un niveau d'abstraction ne dépassant pas le nombre de véhicules dans un tronçon. Cette technique fait appel aux notions de *matrice*

origine destination (OD), d'assignation du trafic et de validation à l'aide de données réelles.

1.2.1 Matrice origine-destination

La littérature recensée révèle que pour simuler le trafic sur un grand réseau, l'assignation du trafic à partir d'une matrice origine destination (OD) est une technique macroscopique couramment utilisée. La matrice OD est un tableau en deux dimensions. Les lignes de celui-ci représentent les points d'origines, les colonnes représentent les destinations et les valeurs du tableau le nombre de véhicules allant de l'origine à la destination correspondante. Par exemple, dans le tableau suivant, 25 véhicules se dirigent du secteur 1 au secteur 2 alors que 33 véhicules se déplacent d'un point du secteur 3 vers un second point du secteur 3 :

Tableau I

Exemple de matrice origine-destination

Destination → / Origine ↓	**1**	**2**	**3**
1	12	25	75
2	35	14	45
3	77	100	33

Généralement, les matrices OD d'une ville entière sont construites par recensement. Cette tâche laborieuse est réalisée par le ministère des Transports du Québec (MTQ) dans toutes les grandes villes de la province, à tous les 5 ans [18]. Évidemment, ces données s'avèrent difficilement accessibles dans une ville étrangère. Heureusement,

différentes techniques ont été recensées dans la littérature qui permettent d'extraire cette matrice à partir de données sur l'état du trafic dans le réseau [19-22] .
Nanda et al. [23] se distinguent puisqu'ils utilisent des valeurs floues de trafic en entrée. Malheureusement, les résultats de leurs travaux demeurent imprécis. De plus, l'extraction complète d'une matrice OD floue constitue une opération demandant un temps de calcul relativement important.

1.2.2 Assignation du trafic

Une fois la matrice OD connue, la seconde étape dans l'obtention de données de trafic consiste à assigner le trafic dans le réseau. Il s'agit de représenter la perception qu'ont les conducteurs du réseau pour les diriger par la suite vers le chemin qu'ils auraient pris, c'est à dire le chemin qu'ils perçoivent le plus court. Cette étape est réalisable à partir d'un processus itératif où une portion du trafic contenu dans la matrice OD est assignée à chaque incrément. Les valeurs du réseau sont modifiées à chaque incrément et un équilibre est ainsi créé dans l'ensemble des chemins du réseau. Lors des premiers incréments, les conducteurs auront tendance à utiliser les grandes artères, ceux-ci étant généralement plus rapides dû à une limite de vitesse supérieure. Par contre, quelques incréments plus tard, ces artères seront congestionnées et des routes secondaires seront favorisées. À la fin de l'assignation, lorsque l'équilibre est atteint, l'ensemble des chemins devrait avoir un temps de parcours similaire.

Quatre catégories d'assignation semblent possibles. Chacune d'elles se distingue par le type de nombre utilisé pour représenter le temps de parcours perçu. La première catégorie, *user equilibrium* (UE) utilise des nombres discrets [24]. Il s'agit de la méthode la plus simple et la plus rapide. L'hypothèse voulant que les conducteurs aient une information parfaite du réseau est le principal défaut de cette méthode. La seconde catégorie, *fuzzy user equilibrium* (FUE), comme son nom l'indique, fait appel à des

nombres flous [16;19;25;26]. Quoiqu'un peu plus lente d'exécution que la méthode UE, elle représente mieux la réalité. En effet, autant la perception qu'ont deux conducteurs du réseau est différente, autant le sera leur décision : deux conducteurs partant du même endroit et se dirigeant au même point ne prendront pas nécessairement le même chemin. La troisième catégorie, *stochastic user equilibrium* (SUE), utilise des lois de la probabilité pour évaluer les temps de parcours [27-29]. Elle possède sensiblement les mêmes avantages et inconvénients que la méthode FUE. Une dernière technique, appelée *hybrid user equilibrium* (HUE), s'avère une combinaison des deux précédentes [30]. Un nombre flou est translaté sur l'échelle du temps en fonction d'une variable aléatoire. Les résultats provenant de cette technique ne montrent pas d'avantages significatifs par comparaison aux méthodes précédentes. La figure suivante illustre les quatre méthodes d'assignation.

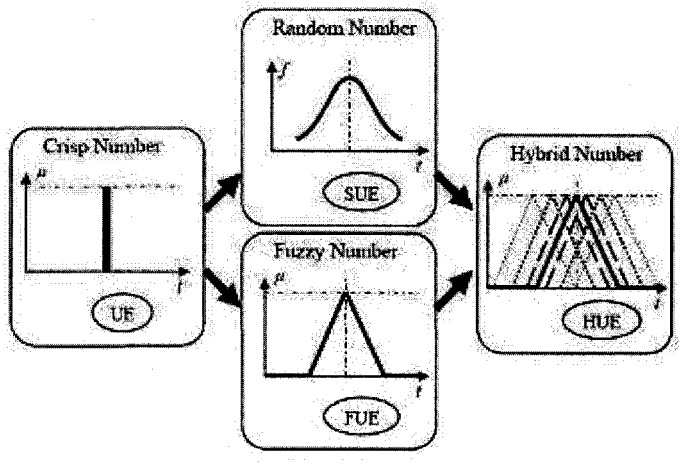


Figure 2 Méthode de représentation des nombres.

Les quatre méthodes de représentation des nombres utilisées dans l'assignation du trafic automobile : *user equilibrium* (UE), *fuzzy user equilibrium* (FUE), *stochastic user equilibrium* (SUE) et *hybrid user equilibrium* (HUE). UE utilise des nombres discrets, FUE utilise des nombres flous, SUE utilise des lois de la probabilité et HUE un mélange des deux derniers. [30]

Une variante intéressante au problème de l'assignation du trafic est proposée par Liao et Wang [31]. Les auteurs ont élaboré un modèle mathématique pour assigner le trafic lorsque les données relatives au temps de parcours et au nombre de voitures sont floues. Ce dernier aspect peut devenir particulièrement intéressant dans un système macroscopique où nous retrouvons une approximation du nombre de véhicules. Par contre, la démarche proposée est très complexe en calcul.

Lors de ce travail, l'*UE* sera utilisée pour créer les données de trafic dans la ville. L'utilisation d'une autre méthode aurait de beaucoup alourdi une tâche déjà très longue à exécuter. Par contre, dans la simulation d'un barrage routier, une perception floue sera utilisée pour améliorer l'exactitude des résultats.

1.2.3 Méthode de saisie de données réelles

Pour bien générer le trafic, il est nécessaire d'obtenir des données réelles de celui-ci à différents points précis du réseau. Ces données serviront soit à valider les résultats obtenus par assignation, soit à extraire la matrice OD lorsque celle-ci n'est pas disponible par recensement. À cet égard, plusieurs techniques sont répertoriées dans la littérature.

L'utilisation de capteurs infrarouges en bordure des routes a été privilégiée par quelques chercheurs [32;33]. Ce type de capteurs est peu dispendieux, fonctionne bien dans diverses conditions météorologiques (neige, pluie, etc.) et sa précision semble meilleure que d'autres capteurs fonctionnant avec les ondes ultrasons ou encore les micro-ondes. Par contre, les capteurs nécessitent d'être posés au-dessus de la chaussée, à divers endroits du réseau routier. Ce dernier point apparaît particulièrement difficile à réaliser dans une zone de combat et inefficace lorsqu'on souhaite utiliser le système en arrivant dans un nouveau théâtre d'opérations.

L'utilisation d'images optiques [34;35] et de radars [36;37] provenant de satellites ou d'autres capteurs aériens et servant à évaluer le déplacement de véhicules est également documentée. Le radar permet une utilisation de jour comme de nuit et ce, indépendamment des conditions météorologiques. Par ailleurs, ce type de système est toujours en développement et nécessite la manipulation de matériel de haute précision (satellite sous les trois mètres de précision et avion de reconnaissance). De plus, les recherches actuelles n'ont pas prouvé la capacité du système à détecter des véhicules dans une zone urbaine à haute densité. Une telle technique a été appliquée de manière limitée, soit uniquement sur une autoroute ou en région rurale. Son utilisation est, pour le moment, laissée de côté, mais pourrait s'avérer pertinente dans un avenir rapproché.

D'autres systèmes font appel à des équipements spécialisés qui doivent être implémentés dans les voitures, permettant que celles-ci soient reconnues par des détecteurs longeant les routes. Tel est le cas d'un projet proposé par Niver et al. [38] utilisant des cartes de péage équipées d'un émetteur. Il est entendu que ce type de système n'est viable que dans une ville « alliée ». Parallèlement, l'installation de détecteurs le long des routes implique des coûts substantiels, sans parler du temps nécessaire d'installation. Ces raisons expliquent que cette technique ne soit propice que sur certaines autoroutes.

Parmi les moyens les plus fréquemment utilisés, on retrouve les systèmes à boucles inductives qui doivent être déposés sur la chaussée ou enterrés sous celle-ci. Les boucles inductives ont pour principale application celle de prendre des lectures occasionnelles d'un point précis du réseau. Cette technique n'est toutefois pas viable dans le cas qui nous préoccupe puisqu'elle nécessite plusieurs détecteurs devant être distribués sur l'ensemble du réseau routier.

Plus intéressante semble être l'évaluation du trafic par un opérateur. Bien qu'elle soit moins précise que les détecteurs, cette technique donne l'avantage d'être facilement utilisable, autant par des troupes au sol que par un opérateur observant le réseau depuis

des images aériennes. Pour que ce système soit facile à utiliser et demeure efficace, l'utilisation de la logique floue est pertinente. Lu [39;40] propose à cette fin une méthode utilisant un *neuro-fuzzy adaptatif system*. Cette technique est adéquate lors d'une simulation macroscopique puisqu'elle fournit des données floues sur le trafic dans le réseau.

1.2.4 Simulation d'un barrage routier

Peu de chercheurs ont étudié l'effet d'un barrage routier sur un réseau. Parmi eux, quelques-uns seulement tiennent comptes de la rapidité de calcul lors de cette simulation puisque leur but est rarement l'implémentation dans une application en temps réel. Seuls Govind et al. [24] ont fixé le temps comme contrainte. Ils ont élaboré une heuristique capable d'évaluer la matrice OD relative à un lien particulier du réseau. De cette façon, lorsqu'un incident survient sur un segment, il est possible de calculer approximativement où se dirigent les voitures passant par ce segment et de rediriger seulement ces véhicules vers un autre chemin. Cette technique macroscopique à l'avantage de grandement diminuer le temps de calcul. Le principal obstacle lié à l'assignation du trafic à partir de cette méthode est qu'il ne montre pas les effets immédiats d'une modification du réseau. Admettons qu'un tronçon routier soit bloqué, cette technique montrera le trafic après qu'un équilibre ait été obtenu. Les données du trafic des premières secondes suivant l'obstruction de la route sont alors perdues!

1.3 Conclusion

Ce chapitre a présenté différents travaux ayant pour sujet la planification d'actions ainsi que la simulation du trafic automobile. Certains de ces travaux seront utilisés dans la réalisation du prototype dont fait objet cette étude. Par exemple, la simulation macroscopique du trafic sera employée pour générer les véhicules dans notre

environnement d'expérimentation. C'est de cela qu'il sera question dans le chapitre suivant.

CHAPITRE 2

RÉSEAU ROUTIER ET GÉNÉRATION DES DONNÉES DE TRAFIC

Le présent chapitre expose d'une part le portrait du réseau routier ayant servi aux expérimentations effectuées dans le cadre de ce mémoire. D'autre part, la méthodologie suivant laquelle les données liées au trafic ont été obtenues sera présentée. Ce chapitre contient également une présentation de ces données.

2.1 Réseau routier

L'ensemble des algorithmes de ce mémoire a été expérimenté sur un graphe généré à partir du réseau routier de la ville de Québec. Les données ont été fournies par l'entremise de la ville de Québec sous la forme d'un fichier de représentation spatiale (*ShapeFile*). Les données contenues dans ce fichier sont représentées sous forme vectorielle, donnant lieu à une grande précision géographique. Le réseau comprend plus de 21 000 tronçons et 10 000 vertex. Chacun des tronçons qui forment le réseau est caractérisé par les attributs suivants :

a. nom de la rue / ville (quartier);
b. classification routière;
c. numéro d'identification de la rue ;
d. longueur ;
e. vitesse maximum autorisée ;
f. direction (sens unique) ;
g. autre (Par exemple : plage de numéro civique…).

Le *nom de la rue* représente son nom réel sur le terrain. Quant à *classification routière*, ce champ représente le type de route. La classification est assez exhaustive. Les indices suivants constituent en fait les différentes valeurs que peut prendre cet attribut :

a. autoroute ;
b. artère principale ;
c. artère secondaire ;
d. collecteur principal ;
e. collecteur secondaire ;
f. local principal ;
g. local secondaire ;
h. local tertiaire.

Le *numéro d'identification* du tronçon est un numéro unique assigné à chacun des tronçons. Il est à noter que dans ces données, aucune distinction n'est faite quant au sens du trafic, à moins que les deux directions possibles ne soient séparées par un terre-plein, par exemple dans le cas d'une autoroute. Ce détail anodin posera quelques problèmes pour l'application des algorithmes. Pour cette raison, il faudra ultérieurement ajouter un nouvel attribut : un numéro sera donné à chacun des sens du trafic. Nous obtiendrons ainsi près de 39 000 tronçons uniques. Ainsi, une rue sera représentée par deux tronçons uniques, à moins qu'il ne s'agisse d'un sens unique.

La *longueur* correspond à la distance entre les deux extrémités du segment alors que la *vitesse maximale autorisée* est la limite légale à laquelle il est permis de circuler. *Direction* est un attribut indiquant si le tronçon est un sens unique, ainsi que la direction du trafic si c'est le cas. Finalement, quelques autres attributs étaient accessibles, sans toutefois être pertinents dans la poursuite de ce travail.

Les données vectorielles contenues dans un fichier à représentation spatiale sont particulièrement efficaces lorsqu'il s'agit d'une application d'affichage. Par contre, pour

l'utilisation dont il est question dans le cas présent, il est beaucoup plus commode de travailler avec un graphe multidirectionnel. La conversion vers ce type de représentation est effectuée par un algorithme existant indépendamment de ce projet. Le résultat, un graphe de plus de 38 000 segments, sera grandement utilisé dans la suite de ce travail. Ses segments possèdent les mêmes attributs que ceux du fichier à représentation spatiale décrit précédemment.

2.2 Génération de données de trafic

Bien que les données du réseau routier fournies par la ville de Québec soient plutôt complètes, il manque cependant un élément essentiel à la simulation d'un barrage routier : les données de trafic. Pour les générer, la technique macroscopique d'assignation à partir d'une matrice OD a été utilisée.

2.2.1 Matrice OD

La matrice OD provient du Ministère des Transports du Québec [18]. À tous les cinq ans, un recensement est fait dans les principales villes du Québec afin d'établir le profil des conducteurs. Pour la ville de Québec, le sondage téléphonique a été conduit sur un échantillon de plus de 27 000 résidents. Parmi les différentes questions posées, les informations suivantes apparaissent particulièrement pertinentes pour l'obtention de la matrice OD :

a. origine du déplacement;
b. destination ;
c. heure de départ ;
d. motif.

Ce sondage a été réalisé sur l'ensemble des territoires suivants :

a. communauté urbaine de Québec;
b. municipalité régionale de comté (MRC) de La Côte-de-Beaupré;
c. MRC de La Jacques-Cartier;
d. MRC de L'Île-d'Orléans;
e. MRC de Portneuf;
f. les 3 MRC de la Rive-Sud de Québec.

Les résultats sont par la suite classifiés selon 13 grands secteurs (ville ou MRC) ou encore selon 67 secteurs municipaux (ville, quartier, partie d'un quartier). Cette deuxième classification s'avère davantage intéressante pour la poursuite du présent travail puisqu'elle offre plus de précision. Ce faisant, une matrice OD de 67 lignes par 67 colonnes pour chacun des moments de la journée est observable. Dans le cas présent, les données de trafic à l'heure de pointe du matin sont celles ayant retenu notre attention. Pour ce faire, 8 matrices ont été fournies par le MTQ : chacune d'elles représente une période de 30 minutes allant de 6 à 10 heures du matin.

Un inconvénient qu'il faut noter est à l'effet que le graphe utilisé ne couvre pas tous les secteurs traités par la matrice OD. Par exemple, les 3 MRC de la Rive-Sud ne sont pas représentées dans le graphe, pas plus que la MRC de La Jacques-Cartier. Il a donc fallu modifier la matrice pour qu'elle concorde avec le graphe tout en minimisant les pertes de véhicules. Par exemple, les MRC de la Rive-Sud ont été fusionnées en un seul point d'OD. Les véhicules voyageant de ces MRC devant nécessairement passer par l'un des deux ponts, un vertex reliant ceux-ci a été ajouté au graphe. Le nouveau secteur d'OD regroupant les 3 MRC de la Rive-Sud est représenté par ce nouveau vertex. Quant à la MRC Jacques-Cartier, elle a simplement été enlevée. Ce choix a été fait en sachant qu'il introduirait un biais dans l'analyse subséquente. Son incidence apparaît mineure compte tenu du nombre réduit de véhicules provenant ou se dirigeant vers cette région. Modifiée, la matrice OD est réduite à 43 lignes par 43 colonnes.

2.2.2 Fonction vitesse-véhicules

Une notion importante à définir est celle de la fonction vitesse-véhicules. Cette fonction, constamment utilisée dans ce projet, évalue la vitesse des véhicules circulant dans un tronçon en fonction du nombre de véhicules dans celui-ci. La fonction choisie est celle de Akcelik [41]. Celle-ci fait office de référence dans le domaine :

$$t = t_0 + 0.25\ T_f\ [z + (z^2 + 8J_A\ x\ /\ QT_f)^{0.5}]\} \qquad (2.1)$$

où

t = temps de parcours moyen par unité de distance (en seconde par km) ;

t_0 = temps de parcours minimum (sans trafic) ;

J_A = paramètre de délai ;

$z = x - 1$;

$x = q / Q$ = degré de saturation ;

q = demande (véhicule par heure) ;

Q = capacité (véhicule par heure) ;

T_f = période pendant laquelle la demande est stable.

Cette fonction a l'avantage de tenir compte du délai induit par les intersections selon le type de route. Akcelik [41] fournit des valeurs de délai et de capacité pour différents types de route. Ces mêmes valeurs ont été appliquées aux algorithmes dans le présent travail. Elles sont représentées dans le tableau II.

Ces types de route ne coïncidant pas avec la classification routière du graphe, nous avons dû adapter les deux ensembles. Pour ce faire, nous avons répertorié l'ensemble des possibilités de classification du graphe et lui avons assigné un type de route tel que présenté dans le tableau II. Par exemple, pour la classification routière 4 (Collecteur principal), le graphe contient 6 possibilités de vitesse maximales. Le tableau III montre la modification apportée après l'observation de chacun des cas.

Tableau II

Valeurs Q et J_A pour les différents types de route

Type de route	Description	Q (veh/h/ligne)	J_A
1	Autoroute	2000	0,1
2	Artère (non interrompue)	1800	0,2
3	Artère (interrompue)	1200	0,4
4	Secondaire (interrompue)	900	0,8
5	Secondaire (friction élevée)	600	1,6

Tableau III

Correspondance entre la classification routière 4 (Collecteur principal) dans le graphe et les types de route selon Akcelik [41]

V_{max}	Type de route	Raison
30	4	Route secondaire dans une ville
50	3	Artère ou route majeure dans une ville
70	3	Artère ou route majeure dans une ville
80	2	Route majeure dans une zone à faible densité urbaine
90	2	Route majeure dans une zone à faible densité urbaine
aucune	4	Cas rare. Route mal répertoriée dans un quartier. Nous lui avons fixé 30 km/h comme V_{max}

À la lumière de ces nouvelles valeurs, il est possible de bien simuler le trafic. Cependant il faut définir l'algorithme utilisé pour arriver à cette fin.

2.2.3 Algorithme d'assignation

L'algorithme de génération du trafic nécessite quelques explications. Il s'agit d'une méthode itérative où des vagues successives de véhicules sont assignées par le chemin le plus court. Ainsi, à chacune des itérations, une portion du trafic contenue dans chacune des paires OD de la matrice est assignée suivant le chemin le plus court. Dans le cas qui nous concerne, la matrice OD faisant 43 par 43, 1 849 paires OD sont à assigner à chacune des itérations. La figure 3 illustre la logique avec laquelle procède l'algorithme d'assignation.

L'algorithme comprend deux boucles. Une première boucle agit sur l'ensemble des paires OD. Elle est imbriquée dans une seconde boucle ayant la même fonction, mais pour les incréments de temps. Ne voulant laisser aucun avantage à une paire OD en particulier, l'algorithme est conçu pour calculer l'ensemble des chemins les plus courts avant de modifier les données de trafic. Ceci étant, lors du calcul de son chemin le plus court, une pair OD a des conditions identiques (données de trafic) aux 1 848 autres paires de la même itération.

L'implémentation s'est faite dans une perspective multitâche et est décrite à l'annexe 1. Étant donné l'énormité des calculs nécessaires, l'utilisation d'un ordinateur multiprocesseur était inévitable pour l'obtention de résultats dans un délai raisonnable. La recherche de chemin est traitée comme une tâche, de telle sorte que les 1 849 chemins à trouver pour chacune des itérations peuvent l'être en parallèle. Il en va de même pour la modification des données de trafic. Pour obtenir les résultats de quatre heures de trafic, cela prend environ 4 jours à un superordinateur comprenant 32 processeurs cadencés à 800 MHz.

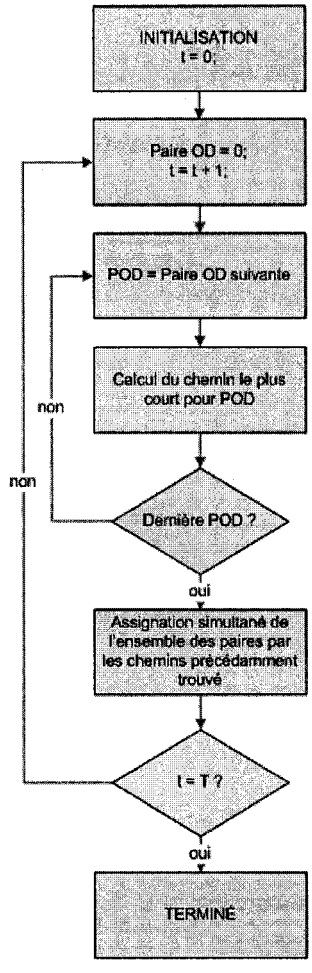


Figure 3 Schéma de l'algorithme d'assignation.

À chaque incrément de temps, un chemin est calculé pour chacune des paires OD avant de modifier les données de trafic de la ville.

2.3 Données réelles de la ville de Québec

Les données réelles utilisées pour valider les résultats ont été fournies par la division du transport de la ville de Québec. Ces données sont des lectures effectuées à l'aide de dispositifs à boucles inductives à des endroits précis sur le réseau routier de la ville. La banque de données comprend plus de 2 700 lectures. Chacune des lectures comporte les champs suivants :

a. date de la lecture;
b. jour de la semaine;
c. localisation;
d. nombre de véhicule de 07h30 à 09h30;
e. autres valeurs de trafic pour différentes heures.

Seule l'heure de pointe du matin est pertinente dans le cas présent puisqu'il s'agit du moment choisi pour simuler le trafic. Par ailleurs, pour la même localisation, plusieurs lectures peuvent avoir été prises à différentes dates s'échelonnant entre 1999 et 2004. Les relevés sont toujours pris durant des jours de semaines et en tentant d'exclure des jours fériés, afin d'obtenir un portrait juste du trafic routier.

2.4 Conclusion

Le présent chapitre a présenté l'environnement d'expérimentation (le graphe de la ville de Québec) et la façon dont les données de trafic ont été générées. Il sera maintenant question des expérimentations ayant directement rapport avec l'objectif de ce mémoire. Rappelons que cet objectif est de modifier notre environnement afin d'améliorer un déplacement. Une façon d'y arriver serait d'effectuer un barrage routier. Cela aurait assurément des conséquences sur le trafic automobile. Le prochain chapitre aborde cette problématique.

CHAPITRE 3

SIMULATION D'UN BARRAGE ROUTIER

Une étape déterminante dans ce projet est le choix des actions à utiliser afin d'améliorer un déplacement. Puisque nous souhaitons nous limiter à peu d'actions et que celles-ci doivent démontrer le plein potentiel du module *OptiEvents*, il est préférable que celles-ci influencent les deux facteurs pondérant les tronçons : la navigabilité (temps de parcours) et la menace. Principalement pour cette raison, notre choix s'est porté sur un barrage routier, situation qui implique ces deux facteurs. Vu la difficulté d'implémenter ce simulateur dans un système en temps réel, la seconde action développée dans le cadre de ce projet consiste en l'interception d'une menace. Cette action étant ne nécessitant pas de simulation, il n'en sera question que dans le prochain chapitre.

3.1 Choix de l'algorithme de simulation

Dans le contexte de ce projet, le simulateur de barrage routier doit avoir comme principales caractéristiques : rapidité d'exécution et précision. La première est imposée par le caractère hautement dynamique de l'environnement dans lequel le système évolue alors que le second l'est par l'aspect non-déterministe de ce même environnement. Le simulateur choisi est celui de Govind et al. [24]. Il est le seul à répondre à toutes nos contraintes, étant le simulateur le plus rapide recensé dans la littérature, même s'il est encore loin d'être viable dans un système en temps réel. Il faudra donc attendre encore quelques années avant que les progrès de l'informatique permettent son utilisation dans un tel environnement. Ce dernier point peut sembler décevant, mais le projet ayant des visées à long terme (10 à 20 ans), cette difficulté pourrait être résolue entretemps.

3.2 Description de l'algorithme de simulation

Tel que déjà mentionné, l'algorithme choisi a l'avantage d'être extrêmement rapide par rapport à ses concurrents. La raison est qu'il approxime la matrice OD aux seuls véhicules se trouvant dans le tronçon bloqué, plutôt que d'évaluer celle de l'ensemble des véhicules du réseau ou du secteur de la simulation. Le calcul de cette matrice est donc beaucoup plus simple. Il en va de même pour la réassignation puisqu'encore là, seuls les véhicules du tronçon bloqué sont réassignés.

Quoique les résultats soient moins précis que certains autres algorithmes, puisqu'il s'agit d'une approximation, le résultat est un processus beaucoup plus rapide dont les étapes sont les suivantes :

a. initialisation ;
b. estimation de la matrice O-D pour le tronçon bloqué ;
c. soustraction des véhicules dans le réseau allant dans le tronçon bloqué ;
d. réassignation du trafic en tenant compte du nouveau réseau et de la connaissance de celui-ci par les conducteurs.

3.2.1 Estimation de la matrice OD

Une fois l'initialisation des variables terminée, la première étape est l'estimation de la matrice OD. Afin d'arriver à ce résultat, il faut en premier lieu sélectionner la région sur laquelle le blocage routier sera simulé. En effet, nous désirons réduire l'effort de calcul en restreignant la zone observée. Il est logique d'opérer d'une telle manière puisque nous savons qu'un blocage a un effet limité sur la zone qui l'entoure. Cette région doit être suffisamment grande pour voir tous les effets de notre action, mais suffisamment petite pour alléger les calculs. De plus, le choix d'une région bordée par le moins de tronçons possibles est important. Ces points seront les origines et les destinations de la future

matrice OD. Plus ils seront nombreux, plus il sera lourd de calculer les opérations futures.

L'estimation de la matrice OD est l'étape suivante. Il s'agit de déterminer d'où proviennent et où s'en vont les véhicules contenus dans le tronçon bloqué. Cette étape est rapide et effectuée une seule fois. La fonction calculant cette matrice est la suivante :

$$V_{ij}^{k} = V^{k} \frac{P_i A_j \delta_{ij}^{k}}{\sum_{ij} P_i A_j \delta_{ij}^{k}} \tag{3.1}$$

où

V^k = le nombre total de véhicules dans le tronçon bloqué k ;

V^k_{ij} = la proportion de V_k allant des points externes i à j ;

P_i = le nombre de véhicules entrant par le point i ;

A_j = le nombre de véhicules sortant par le point j ;

δ^k_{ij} = 1 si tronçon k fait partie du chemin le plus court entre i et j ; sinon, δ^k_{ij}=0.

Par contre, cette équation possède une imprécision flagrante : δ est soit égal à 1 si le chemin entre les points i et j est le plus court, sinon δ prend la valeur 0, de sorte qu'un seul chemin, le plus court, est considéré. En utilisant une perception floue du chemin le plus court tel que proposé par Liu et al. [42], δ prendrait une valeur comprise dans l'intervalle 0 à 1 et plusieurs chemins pourraient être considérés. La précision serait en conséquence rehaussée, néanmoins le temps de calcul le serait tout autant.

Dans le cas présent, cette perception floue a été incorporée sans grandement alourdir l'algorithme en appliquant la modification suivante : pour chacune des paires OD, sont calculés le chemin le plus court d'une distance *x*, et le chemin le plus court passant par le tronçon bloqué *k*, d'une distance *y*. Une fonction floue est construite autour du premier pour calculer δ^k_{ij}. Ainsi, une perception floue de la vision du réseau par les conducteurs est introduite. La figure 4 illustre ce propos.

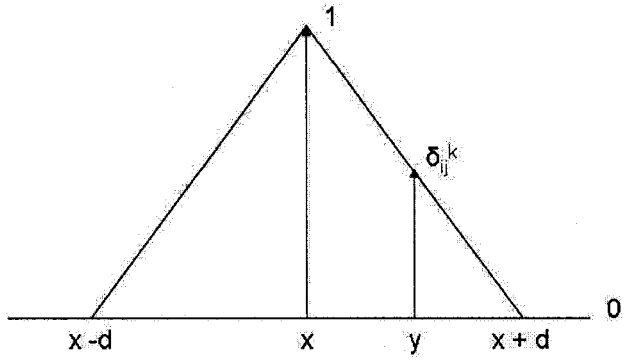


Figure 4 Fonction floue utilisée pour représenter la perception des automobilistes.

Le chemin le plus court est d'une longueur de x alors que le chemin le plus court passant par le tronçon bloqué est d'une longueur de y. δ^k_{ij} est le degré d'appartenance à la fonction.

Lorsque *y* est égal à *x*, il s'agit d'un cas particulier où δ^k_{ij} sera égal à 1. À l'inverse, si *y* est supérieur à $x+d$ alors δ^k_{ij} sera égal à zéro et le chemin ne sera pas considéré. Il ne reste qu'à déterminer une valeur raisonnable pour la valeur *d*. À la lumière des expérimentations réalisées, il a été convenu qu'une valeur de $x+d$ égale à $1{,}20x$ est correcte.

Pour l'instant, ce stratagème ne sert qu'à ajouter certaines paires OD dans le calcul de la matrice OD du tronçon bloqué tout en ajoutant de la précision pour la valeur δ^k_{ij}. Mais jamais, pour une paire OD, on ne trouve plus d'un chemin. En revanche, pour être plus précis, nous devrions trouver tous les chemins ayant δ^k_{ij} plus grand que 0 et cela, pour chacune des paires OD. Or, un algorithme capable de trouver k chemins les plus courts, est très lourd. Il semblait pertinent d'en prendre note tout de même à titre de considération future.

3.2.2 Soustraction des véhicules

En trouvant la matrice OD, nous avons réussi à extraire de l'information concernant les allées et venues des véhicules impliqués dans le tronçon bloqué. Partant de l'hypothèse que le trafic est constant sur un court intervalle de temps, nous pouvons « reculer dans le temps » et recommencer en tenant compte des modifications survenues dans le réseau. Pour ce faire, il faut premièrement enlever du réseau les véhicules contenus dans le tronçon bloqué, ceux s'y dirigeant et ceux y étant passés. La façon d'y parvenir consiste en une méthode similaire à l'assignation développée dans le chapitre précédant. Par contre, dans le cas présent, on enlève les véhicules du réseau plutôt que de les y ajouter.

L'origine et la destination des véhicules contenus dans le tronçon bloqué étant connues, le chemin le plus court entre ces points sera aisé à déterminer. Par contre, dans la recherche du chemin le plus court, les temps pour parcourir les tronçons du réseau ne sont pas calculés en fonction du nombre de véhicule dans le tronçon. Ils sont plutôt fonction du nombre de véhicule dans le tronçon auquel nous enlevons le nombre de véhicules à soustraire dans la présente itération. La raison est simple, nous voulons « revenir en arrière ». Pour ce faire, il faut déterminer quels chemins les plus courts chemins les conducteurs ont observés. Il faut donc se remettre à leur place en enlevant les véhicules de cette itération. Il s'agit là de l'opération inverse de l'assignation où l'on calcule le chemin le plus court et où on ajoute les véhicules. Dans le cas qui nous concerne, on enlève les véhicules et on calcule le chemin le plus court.

Une fois le chemin le plus court trouvé, les véhicules sont enlevés seulement dans les tronçons de ce chemin. Le même exercice est appliqué pour les autres paires OD et ce, pour chacune des itérations. Lorsque ces opérations sont complétées, nous avons réussi à utiliser l'information extraite par l'équation 3.1. Il est maintenant possible de réassigner les véhicules pour trouver le nouvel équilibre du nouveau réseau comportant un tronçon bloqué.

3.2.3 Réassignation

Ayant retiré les véhicules passant par le tronçon bloqué du réseau, il ne reste qu'à les réassigner en tenant compte de notre nouvelle information (origine, destination, tronçon bloqué). Cette technique est, une fois de plus, basée sur l'hypothèse que la quantité de véhicules entrant dans notre réseau est fixe. On peut raisonnablement admettre ceci puisque que le laps de temps pendant lequel a lieu la simulation est relativement court.

Parmi les différentes techniques d'assignation du trafic répertoriées dans la littérature, toutes ont en commun le même défaut : elles attribuent aux conducteurs une connaissance de l'état de l'ensemble des segments du réseau. Ainsi, dès que les véhicules entrent dans le réseau, ils tenteront d'éviter le tronçon bloqué. Dans la réalité, un conducteur ne sait pas qu'une route est bloquée s'il ne la voit pas. Les algorithmes proposés dans la littérature sont donc incapables de fournir des données justes pour les premiers moments suivant la modification du réseau. Ils omettent les premiers bouchons se formant à proximité du tronçon fermé. Ces algorithmes sont seulement capables de fournir l'état de la simulation lorsque l'équilibre est atteint. Pour pallier à cette faiblesse, une technique faisant l'assignation via le chemin original (sans barrage routier) a été développée. À moins que le véhicule ne soit rendu au tronçon bloqué ou que la quantité de véhicules dans le tronçon où il se trouve soit anormalement élevée, il reste sur le chemin calculé sans barrage routier. Cette dernière condition permet aux premiers véhicules de ne pas éviter une congestion créée par notre modification du réseau et de rester fidèle à la réalité. À ce propos, les deux images de la figure 5 illustrent bien l'utilité de notre nouvelle démarche. Celle de gauche démontre les résultats pour une assignation suivant les modèles déjà existants. Les véhicules connaissent, dès leur arrivée dans le réseau, l'état de chacun des tronçons formant celui-ci. Quatre chemins sont envisageables. Ceux en rouge sont irréalistes lors des premiers moments, puisqu'il suppose une connaissance de la modification du réseau par les conducteurs. Par contre, dans la figure de droite, représentant la méthode élaborée aux fins de cette recherche, les

conducteurs ne modifient par leur parcours initial avant d'atteindre le barrage ou un éventuel bouchon de circulation. Ainsi, uniquement deux routes sont considérées (au début), ce qui permet de croire que les résultats sont plus fidèles à la réalité.

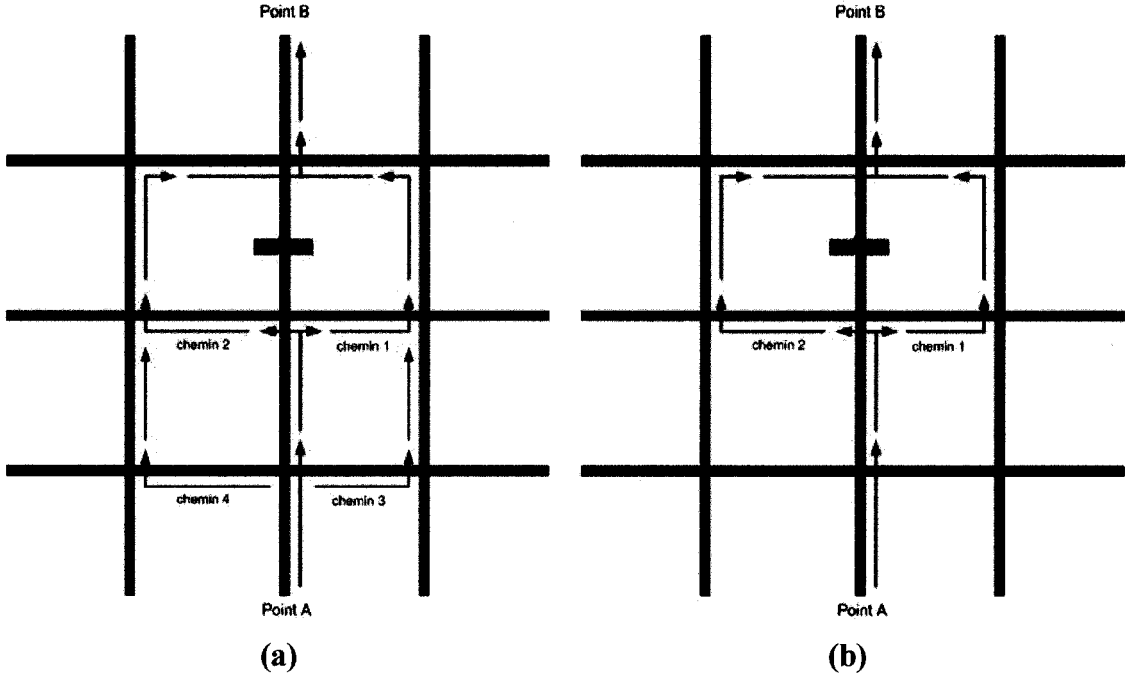


Figure 5 Résultat de l'assignation.

(a) Résultat suivant les modèles existant. Les véhicules connaissent, dès leur arrivée dans le réseau, l'état de chacun des tronçons formant celui-ci. Quatre chemins sont envisageables. Ceux en rouge sont irréalistes lors des premiers moments; (b) Résultat suivant la nouvelle méthode d'assignation. Les conducteurs ne modifient par leur parcours initial avant d'atteindre le barrage ou un éventuel bouchon de circulation. Deux routes sont considérées.

Les résultats de cette partie sont pour le moment très prometteurs. Cependant, le temps de calcul est fortement augmenté par cette technique. Dans l'optique d'un système en temps réel, cette amélioration est coûteuse, mais dans le cadre d'un projet comme celui-ci, travaillant dans un environnement hautement dynamique, il serait impensable de faire autrement vu la précision des effets recherchés.

3.3 Réalisation du simulateur

La réalisation du simulateur s'est faite à l'aide d'une seule classe. Sa mise en œuvre est purement séquentielle suivant les étapes décrites précédemment dans ce chapitre et illustrées par la figure 6. Les détails sont présentés dans l'annexe 2.

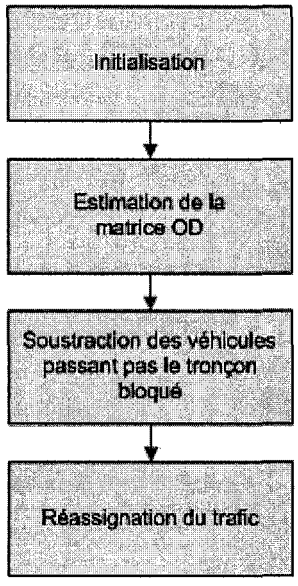


Figure 6 Schéma de l'algorithme du simulateur de barrage routier.

Quatre étapes linéaires le composent : initialisation, estimation de la matrice OD, soustraction des véhicules, réassignation.

3.4 Conclusion

Ce chapitre a décrit le simulateur d'une première action pouvant modifier les conditions du réseau : un barrage routier. Avec ce simulateur, nous pouvons décrire les effets d'une telle action en un endroit précis. Néanmoins, pour atteindre l'objectif de ce travail, le prototype final devra être en mesure de déterminer où effectuer ces actions en fonction du déplacement à améliorer. De plus, le prototype devra trouver qui effectuera ces

actions. C'est pour répondre à ces interrogations que le planificateur présenté au chapitre suivant a été développé.

CHAPITRE 4

PLANIFICATION D'ACTIONS

L'étape principale de ce travail consiste en la réalisation d'un planificateur d'actions. Rappelons que le but de ce planificateur est l'amélioration d'un déplacement d'un véhicule automobile dans un contexte tactique. Les contraintes imposées par la situation sont la rapidité d'exécution et la précision. Le chapitre précédant traitait du simulateur de blocage routier constituant notre première action envisageable. La seconde action dont il sera question dans ce chapitre est l'interception d'un véhicule menaçant.

Il est nécessaire de savoir planifier les actions en fonction de leurs effets. Pour cette raison le *planificateur de haut niveau* (PHN) entre en ligne de compte. Celui-ci coordonne, fusionne et cherche les actions à effectuer. Il est, en quelque sorte, le cerveau de l'opération. Cette tâche étant cependant trop imposante pour lui seul, le PHN fait appel à des *planificateurs de bas niveau* (PBN) qui sont spécifiques aux actions de chaque unité. Ces deux types de planificateur occupent donc une place importante dans le contenu de ce chapitre.

Finalement, la structure du module *OptiEvents* telle que nous l'avons développée sera exposée au cours des lignes suivantes. Nous pouvons d'ores et déjà convenir que cette structure emprunte la forme d'un système multiagent.

4.1 Planificateur de haut niveau

Le PHN, planificateur relatif à un déplacement, a comme tâche le choix du parcours de l'utilisateur, mais surtout la planification des diverses actions permettant la réalisation de

ce chemin dans un environnement sécuritaire. Il doit coordonner les actions à effectuer avec les différentes ressources sur le terrain.

Lorsqu'un utilisateur entre dans le système et demande un trajet, un PHN est automatiquement créé. Il a comme première fonction de trouver le chemin le plus court pour l'utilisateur et ce, sans se soucier des menaces. Par la suite seulement il évalue les *tronçons problématiques* (TP). Les TP représentent des segments de notre trajet sur lesquels peut se trouver une menace pour l'utilisateur. La recherche de ces segments sera aisée à effectuer grâce aux informations extraites par l'*observateur d'environnement* (OE). Il en sera question plus loin dans ce chapitre.

Si aucun TP n'est trouvé, le chemin est sécuritaire et l'utilisateur peut l'emprunter sans problème. À l'inverse, il lui faudra réagir. Ce n'est pas l'utilisateur qui va poser des actions, mais les *ressources* disponibles. Ces *ressources* sont, en fait, d'autres agents indépendants dans notre système. Ils peuvent être des unités militaires, des véhicules de police ou encore toute autre ressource alliée en contact avec notre système. Pour œuvrer sur les TP, ces agents doivent recevoir des instructions sur les travaux à effectuer. À cet effet, le PHN va leur envoyer une *annonce de tâche*. Il s'agit d'un message transmis à tous les agents leur signalant l'existence de ces TP. À partir de là, les différentes ressources vont, individuellement, calculer des actions capables d'améliorer un ou plusieurs TP. Nous développerons davantage ce sujet plus loin dans ce chapitre lorsqu'il sera question des PBN.

La seconde grande responsabilité du PHN est la coordination des actions. En effet, il est possible, mais rare, qu'une seule action puisse résoudre tous les problèmes. Au contraire, dans la majorité des cas, il faut plusieurs actions combinées ensemble pour arriver au résultat escompté. Les différentes *ressources* produisent divers plans, réglant chacun une partie des problèmes. Il faut trouver une combinaison de plans réglant tous les problèmes si c'est possible, sinon, la meilleure combinaison. C'est le rôle de

l'*unificateur de plans* (UP) qui fait partie du PHN. À ce propos, quelques contraintes et nuances seront observées ultérieurement.

Finalement, le PHN devrait aussi posséder des PBN. En effet, l'utilisateur est possiblement en mesure d'accomplir des actions susceptibles de s'aider, ou encore d'aider d'autre utilisateur. Le prototype résultant de ce travail ne considérant pas le mode multiutilisateur, cette idée n'a pas été développée davantage. Cela demeure matière à réflexion.

4.2 Planificateur de bas niveau

Il a été constaté que le PHN envoie une *annonce de tâche* aux différentes *ressources* leur demandant de trouver des solutions à des problèmes sur certains TP. À ce moment précis, les PBN entrent en jeu. Une ressource peut accomplir un nombre limité d'actions. Dans notre prototype, il y en a deux, effectuer un barrage routier ou encore intercepter une menace. Ainsi, deux PBN différents seront appelés pour chacune des ressources. Ils ont comme mission de trouver des solutions suivant l'*annonce de tâche* du PHN. La logique de réflexion d'un PBN est indépendante de celle d'un autre PBN. Par la suite, les PBN doivent retourner au PHN les plans trouvés pour que celui-ci les fusionne et prenne une décision.

Les deux actions que nous désirons utiliser, barrage routier et interception d'une menace, possèdent à peu de chose près le même PBN. En fait, l'interception d'une menace est un cas particulier du barrage routier : il ne produit aucun effet sur le trafic. Son action se limite au tronçon où se situe la ressource. Aucune simulation n'est requise pour déterminer son effet sur le réseau routier. De cette manière cette action devient beaucoup plus facile à implémenter dans un système en temps réel.

Voyons d'abord les différentes étapes du PBN:

a. lecture de l'*annonce de tâche*;
b. présélection des tronçons potentiellement intéressants;
c. calcul des effets;
d. envoi des plans au PHN.

La première étape a déjà été exposée, il s'agit de lire les TP provenant du PHN. Quant à la seconde étape, elle est de haute importance puisqu'elle agit à titre de filtre. Le PBN ne peut pas simuler l'effet qu'aura une action particulière sur chacun des tronçons de la ville. Cette tâche serait beaucoup trop fastidieuse pour être viable dans un système en temps réel. Ainsi, il doit restreindre ses recherches aux endroits les plus prometteurs. Par la suite seulement il simulera son action à ces emplacements.

Dans notre cas, la présélection se fait en utilisant un algorithme relativement simple. Il s'agit de trouver les chemins par lesquels la menace peut passer pour atteindre le parcours de l'utilisateur avant celui-ci. C'est sur ces chemins qu'il faudrait agir. Les étapes suivantes sont nécessaires à cette fin.

a. pour chacun des TP, trouver le chemin le plus court entre la menace et ce TP;
b. pour chaque chemin, noter les tronçons faisant partie du chemin;
c. identifier des redondances dans les tronçons.

Ces étapes sont simples, rapides d'exécution et le résultat est étonnamment efficace. Le calcul d'un chemin le plus court étant pratiquement instantané, une fraction de seconde suffit pour terminer la recherche des points prometteurs. Plus la redondance est élevée pour un tronçon, plus une action posée à cet endroit risque d'avoir un effet sur un grand nombre de TP. Bien que les PBN simulent leur action sur l'ensemble des tronçons trouvés par cet algorithme, ils le font par ordre décroissant de redondance, en commençant par les plus prometteurs. Cette façon de faire favorise les chances de trouver une bonne solution dans un court délai.

La prochaine étape de notre PBN est le calcul des effets sur la menace. Pour le barrage routier, il faut calculer l'effet qu'auront les modifications du réseau et du trafic sur les déplacements de la menace. En résumé, il faut répondre à la question suivante : *maintenant que j'ai créé un bouchon de circulation, est-ce que la menace peut encore m'atteindre (arriver avant moi dans un segment de mon parcours)?* Pour répondre à cela, il faut comparer le temps auquel l'utilisateur arrivera dans chacun des tronçons avec ceux de la menace (suivant les modifications apportées au graphe). Il s'agit de la même procédure que celle utilisée par le PHN lors de la détection des TP. Encore une fois, cette tâche sera complétée grâce à l'observateur d'environnement dont il sera question plus loin.

Quant à notre seconde action, l'interception d'une menace, la procédure est encore plus simple. Cette action ne produit aucun effet autre que d'interdire à la menace de ressortir du tronçon où a lieu l'action. Il n'y a donc aucune modification du réseau à simuler. Néanmoins, pour en calculer l'effet, il suffit d'enlever le tronçon du graphe et répondre à la même question (par la même méthode): *est-ce que la menace peut encore m'atteindre?* Les effets trouvés en répondant à cette question font partie d'un plan qui est transmis au PHN.

4.3 Structure du module OptiEvents

Après avoir exploré les deux parties fondamentales du module, soit les planificateurs, il est temps passer au fonctionnement général du module. Sa structure est organisée comme un système multiagent. L'utilisateur et les ressources sur le terrain sont tous des agents indépendants. Ce sont eux qui communiquent ensemble pour planifier les actions à effectuer. La figure 7 illustre la structure générale du module :

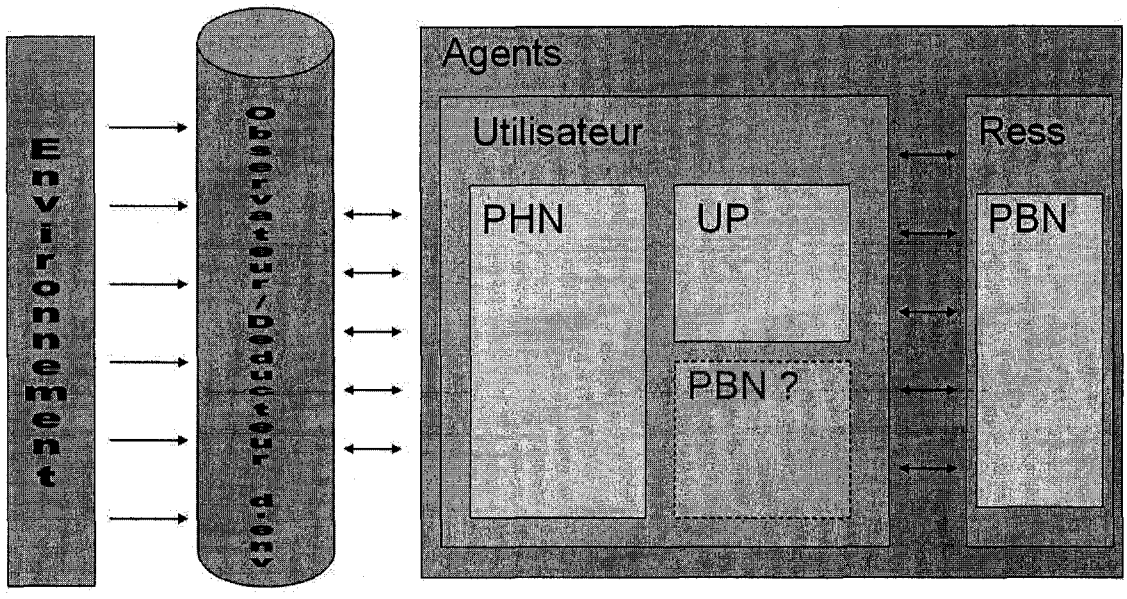


Figure 7 Structure générale du module *OptiEvents*.

L'OE agit à titre de base de données et de récepteur d'information du module. Deux types d'agent sont présents : *utilisateur* et *ressource* (Ress). Le premier a la capacité de créer des PHN et des UP alors que le second peut concevoir des PBN. Une version futur du module devrait permettre aux agents de type *utilisateur* la création de PBN.

Cette figure illustre deux parties qui n'ont pas été encore décrites : l'observateur d'environnement et l'unificateur de plans.

4.3.1 Observateur d'environnement

L'OE sert à la fois de base de données et de structure exploitant les connaissances du système. C'est là que sont emmagasinées toutes les données relatives au graphe, au trafic ainsi qu'aux unités. De manière plus précise l'OE a en mémoire des informations relatives aux *menaces*, aux *ressources* ainsi qu'à l'utilisateur. Pour chacune de ces unités, différents champs sont gardés en mémoire. Parmi ceux qui sont communs aux trois types, notons les suivants :

a. nom de l'unité;
b. dernière position connue;
c. table de temps de déplacement minimal.

Nous pouvons ajouter les champs suivants qui sont spécifiques aux ressources et dans certains cas, à l'utilisateur:

a. destination (aussi applicable aux unités de type utilisateur);
b. trajet (aussi applicable aux unités de type utilisateur);
c. état;
d. employeur;
e. actions possibles.

Le nom des unités est donné par l'ordre où elle arrive dans le système. Par exemple, la première menace répertoriée portera le nom *m1*, la seconde *m2* et ainsi de suite. Il en va de même pour les *ressources* (*r1*, *r2* ...) ainsi que les utilisateurs (*b1*, *b2*....). Ces noms permettent d'identifier à la fois l'unité et son type. Dans le cas présent, il n'y aura qu'un utilisateur (*b1*).

Le champ *dernière position connue*, comme son nom l'indique, est le dernier endroit où l'unité a été aperçue par notre système. Quant à elle, la *table de temps de déplacement minimal* (TTDM) est mise à jour à chaque nouvelle position connue de l'unité. Dans ce but, un algorithme calcule pour chacun des tronçons du graphe, le temps minimal nécessaire pour que l'unité s'y rende. En fait, cette démarche équivaut à trouver le chemin le plus court entre l'unité et chacun des segments qui forment notre graphe. Cet algorithme peut sembler coûteux en temps de calcul, mais il en va tout autrement. En fait, il s'agit d'une variation de l'algorithme de *Dijsktra* [43]: on demande au système de trouver le chemin le plus court partant de la position actuelle, sans toutefois préciser la destination. En enlevant la condition d'arrêt qui est normalement atteinte lorsque la recherche parvient au point d'arrivée, on force l'algorithme à parcourir l'ensemble du

graphe. À chaque vertex rencontré, le système note le temps dans une table. Lorsque le système a fini cette épreuve, la table contient tous les vertex avec le temps le plus court pour y parvenir. Il s'agit de notre TTDM. Ces données sont extrêmement utiles aux différents planificateurs pour trouver les TP et pour vérifier l'effet des actions. En effet, il suffit de comparer les TTDM de deux unités pour trouver laquelle arrivera la première dans un segment de route particulier.

État, *employeur* et *actions possibles* sont trois champs uniques aux unités de type *ressource*. Le premier représente ce que fait l'unité en ce moment : déplacement, interception, attente... *Employeur* représente, le cas échéant, l'utilisateur pour qui travaille la *ressource*. Lorsqu'un agent accomplit une tâche demandée par un utilisateur, il est en quelque sorte son employé. Ce champ pourrait être utile dans l'éventualité d'un système multiutilisateur où des négociations devraient avoir lieu entre plusieurs utilisateurs se partageant les mêmes *ressources*. Toutefois, dans le présent cas, il s'agit simplement d'une référence pour le débogage. Finalement, le champ *actions possibles* est une liste des différentes actions envisageable par l'agent. Il y en aura deux dans le prototype faisant l'objet de ce mémoire.

4.3.2 Unificateur de plans

Il a été plus tôt constaté que les PBN retournent une multitude de plans au PHN. Ces plans ne sont pas toujours parfaits – ils ne répondent pas à tous les problèmes – alors il est préférable de les combiner pour obtenir un meilleur plan. Ce travail est accompli par l'UP.

Auparavant, il est important de s'entendre sur le concept de ce qu'est un plan pour notre système. Il s'agit d'un objet informatique ayant les attributs suivants :

a. pointage;
b. résultat sur les vertex;
c. endroit;
d. ressource utilisée;
e. sous-plans.

Pointage est une valeur qualifiant le plan. Plus cette valeur est élevée, plus le plan améliorera notre situation. Le *pointage* est calculé à partir du nombre de vertex jugés non sécuritaires initialement et qui le sont devenus grâce au plan en question. Par exemple, si le PHN demande d'améliorer la sécurité sur 5 vertex tous affectés par une menace quelconque, le pointage maximum sera de 5.

Néanmoins, le *pointage* n'est qu'une valeur numérique. Elle n'indique pas quel vertex a été amélioré par ce plan et quel vertex ne l'est pas. C'est par l'attribut *résultat sur les vertex* que sera mémorisée cette information. Elle prend la forme d'une *table de hachage*. « Une table de hachage est une structure de données qui permet une association clé-élément [...]. On accède à chaque élément de la table via sa clé. En quelque sorte, il s'agit d'un tableau ne comportant pas d'ordre. »[1] Dans notre *table de hachage*, pour chacun des TP fournis par le HLP, une valeur binaire correspond aux menaces affectant, ou pas, ce TP. Par exemple, si pour le premier TP la valeur binaire est *101*, nous sommes en mesure de savoir qu'avec ce plan, la première et la troisième menace affectent ce TP. Ce type d'écriture permet de faire rapidement des opérations logiques sur deux plans afin de vérifier le résultat d'une éventuelle fusion.

L'attribut *endroit* représente les endroits où auront lieu les actions. En effet, un plan peut être le résultat de plusieurs actions faites à différents endroits par différentes ressources (attribut *ressource utilisée*). Lorsque deux plans sont fusionnés, ils forment un nouveau

---

[1] Selon l'encyclopédie en ligne Wikipédia à http://fr.wikipedia.org/wiki

plan ayant la même forme (même objet), tout en gardant une trace des sous-plans le constituant. Cette trace est contenue dans l'attribut *sous-plans*. La figure 8 reflète bien ce derniers propos :

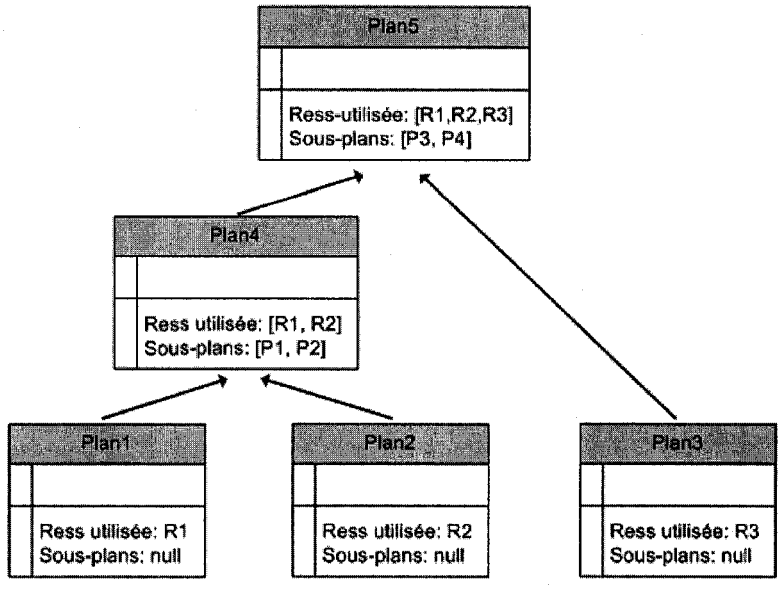


Figure 8 Fusion de plans.

Exemple où trois plans sont fusionnés en un seul. Dans une première étape, les plans 1 et 2 sont fusionnés pour donner le plan 4. Celui-ci est fusionné avec le plan 3 pour donner un plan plus global, le plan 5. Pour chacun des plans, un premier attribut conserve une référence aux plans simples le constituant, alors qu'un second réfère aux ressources utilisées dans ces plans.

Il est temps d'aborder la fusion de plans. Pour être fusionnés en un plan *P*, deux plans, *p1* et *p2*, doivent répondre aux deux conditions suivantes :

a. [pointage (P) > pointage (p1)] ET [pointage (P) > pointage (p2)];
b. (si p1, p2 est réalisable) ET (si p2, p1 est réalisable).

Ces conditions sont simples et intuitives. La première affirme que le plan résultant de la fusion doit être meilleur que chacun des deux plans pris individuellement. Cette condition n'est pas atteinte dans le cas où l'un des plans n'amènerait rien de mieux au second. Prenons un exemple où nous devons éliminer une menace sur 5 tronçons. Si le

premier plan affecte positivement les trois premiers tronçons, alors que le second n'affecte que le premier tronçon, nous sommes en présence d'un cas où les effets du second plan sont un sous-ensemble des effets du premier plan. Ainsi, la condition n'est pas atteinte puisque le plan fusionné n'est pas meilleur que le premier plan, mais seulement équivalent (en termes d'effets). Par contre, dans le même cas, si le second plan affectait positivement les trois derniers tronçons, les deux plans seraient complémentaires et le plan fusionné serait supérieur aux plans pris individuellement. Ce plan constituerait un choix avantageux.

La seconde condition affirme plutôt qu'un premier plan ne doit pas empêcher la réalisation du second pour que ceux-ci puissent être unifiés. Dans le présent prototype, l'hypothèse voulant qu'une *ressource* ne puisse accomplir qu'une action est avancée. La condition est donc équivalente à dire que si un plan fait appel à une *ressource r1*, il ne peut être unifié à un second plan faisant aussi appel à *r1*. Cette hypothèse simplifie beaucoup l'algorithme puisqu'elle enlève la notion de temps pour accomplir une action. Dans une version plus avancée du module *OptiEvents*, il faudrait ajouter cette notion, pour ainsi permettre à une *ressource* de faire plusieurs actions l'une après l'autre. La condition serait donc modifiée pour la suivante : si un plan fait appel à une *ressource r1* à un temps t, il ne peut être unifié à un second plan faisant aussi appel à *r1* au même temps t.

4.3.3 Communication

La communication entre les agents est essentielle dans un système multiagent. Dans notre cas, le mode de communication est relativement simple. Chacun des agents possède une boîte de réception de message. Chacun a accès à la boîte des autres agents et peut y déposer des messages à son gré. Ainsi, lorsqu'un agent veut envoyer un message, il construit un objet informatique standard pour l'ensemble des agents, appelé *message*, et le dépose dans la boîte de réception des destinataires. Si la boîte de réception d'un agent contient des messages, celui-ci les lit par ordre d'arrivée avant de les envoyer dans une seconde boîte contenant les messages ouverts.

Ce mode de communication a été préféré pour des questions de simplicité. Vu l'avancement du prototype et la quantité réduite d'agents, les contraintes de sécurité et de performance ont été négligées.

4.4 Fonctionnement complet du module OptiEvents

Les différentes parties du module ayant été discutées, passons maintenant à leur fonctionnement et aux relations qui les unissent. La meilleure façon d'exposer tous les mécanismes qui agissent dans le module est de procéder par un exemple.

Supposons qu'un utilisateur entre dans le système et demande un chemin entre sa position actuelle et une destination quelconque. Si aucune menace n'est détectée, le système proposera à l'utilisateur le chemin le plus court ou le plus rapide, dépendamment de ses préférences. Ainsi, l'utilisateur se déplace suivant ce chemin. Pendant ce temps, l'OE vérifie la position de l'utilisateur ainsi que des différentes ressources et met à jour les attributs de la base de données. À un certain moment, une *menace* est repérée dans la ville. Automatiquement, le PHN de l'utilisateur vérifie si

cette nouvelle *menace* peut perturber le parcours actuel. Dans ce but, le PHN fait appel aux données contenues dans l'OE. Plus particulièrement, il compare les valeurs des TTDM de l'utilisateur et de la *menace* pour les vertex du parcours de l'utilisateur. Si ceux de la *menace* sont plus petits que ceux de l'utilisateur (si la *menace* peut arriver avant l'utilisateur) alors nous avons possiblement un problème. Il doit donc trouver une façon d'éviter un contact entre les deux. Il envoie un message d'*annonce de tâche* à l'ensemble des agents, leur spécifiant ses demandes. Il spécifie l'ensemble des TP avec la *menace* s'y rattachant et leur demande d'empêcher cette *menace* d'arriver avant lui. Enfin, le PHN spécifie un temps limite pour soumettre les plans.

Les différents agents reçoivent le message. S'ils sont disponibles, ils vérifient s'ils peuvent faire quelque chose pour aider l'utilisateur. Dépendamment du type de *ressources*, ils possèdent certaines actions envisageables. Pour chacune de ces actions, chacun des agents démarre un PBN. Les PBN fournissent les plans les plus simples. C'est la base de l'arbre de plans de la figure 8. Si l'un de ces plans est en mesure de répondre à toutes les demandes de l'*annonce de tâche*, l'agent écrit à nouveau au PHN pour lui envoyer ce *plan complet*. Le PHN prendra alors la décision d'utiliser ou non ce plan. Dans l'affirmative, il renverra un message à l'ensemble des agents pour fermer le contrat contenu dans l'*annonce de tâche* précédente (terminant ainsi les recherches de plan par les PBN) et un second message contenant des ordres pour l'agent ayant trouvé le *plan complet*. Par contre, comme c'est généralement le cas, un plan simple ne pourra pas résoudre l'ensemble des problèmes causés par une *menace*. Il faudra plus souvent qu'autrement fusionner plusieurs *plans partiels* pour en obtenir un plus complexe, mais certes plus efficace. Ainsi, au fur et à mesure que les PBN trouvent des *plans partiels*, ils les envoient à l'UP relatif à l'utilisateur.

Rappelons que l'UP a pour mission de fusionner plusieurs plans simples en un plan plus complexe. Si l'UP trouve un *plan complet*, il l'envoie au PHN pour que celui-ci prenne une décision. Si le PHN décide d'exécuter ce plan, il ferme le contrat et envoie les

ordres d'une manière similaire à celle déjà vue. Par contre, il devra décomposer le plan en sous-plans, pour l'envoyer à toutes les *ressources* impliquées. Ceci est fait grâce à l'attribut *sous-plans* contenu dans l'objet *plan*. Par contre, si l'UP n'arrive pas à trouver un *plan complet*, il devra se contenter du meilleur *plan partiel* trouvé à l'heure limite fixée dans l'*annonce de tâche*. Les plans sont évalués en fonction de leur *pointage*.

Si pendant l'exécution des ordres, des informations supplémentaires sont rapportées, alors les étapes précédentes sont répétées. Par exemple, si une seconde *menace* est repérée, le PHN recommencera en fonction des deux *menaces*. De la même manière, si la première *menace* est repérée à nouveau, le PHN devra encore vérifier sa planification ultérieure. En effet, peut-être certaines actions préalablement amorcées ne sont plus d'aucune utilité compte tenu des derniers renseignements. Peut-être ces *ressources* pourraient être utilisées autrement, d'où l'utilité de recommencer la planification.

4.5 Implémentation du module OptiEvents

L'implémentation du module peut être représentée par la figure 9. Afin d'alléger sa représentation, certaines parties ne sont pas représentées sur ce diagramme. Parmi ceux-ci, notons les différentes classes de recherche du chemin le plus court, les classes sur les données de trafic ainsi que la table de correspondance trafic-temps de parcours.

On remarque en bleu sur la figure 9 les classes définissant les agents. Les deux types d'agent, *ressource* et *utilisateur* héritent tous les deux d'une classe plus générale. Tel que décrit précédemment dans ce chapitre, les agents construisent des objets informatiques de planification (en vert sur la figure) que sont les PHN et PBN. Il faut ajouter une classe d'UP construit par le PHN au besoin. Ce sont en quelque sorte les classes travaillantes, ce sont eux qui exécuteront la majeure partie des algorithmes. Les classes d'agent ne seront utilisées que pour emmagasiner des données sur l'agent à

proprement dit (type d'unité, actions possible...) et pour envoyer et recevoir des messages.

Les classes ayant pour but la communication entre les agents sont reproduites en jaune sur la figure 9. Ces échanges sont faits à l'aide de messages standards. Lorsqu'un planificateur veut envoyer un message, il construit un objet informatique définissant l'envoi puis il demande à l'agent pour qui il travaille de l'acheminer. Ces message sont variés, allant de l'ouverture d'un contrat jusqu'à l'envoie d'un plan. Dans ce dernier cas, un plan construit à l'aide d'une autre classe est ajouté à ce message.

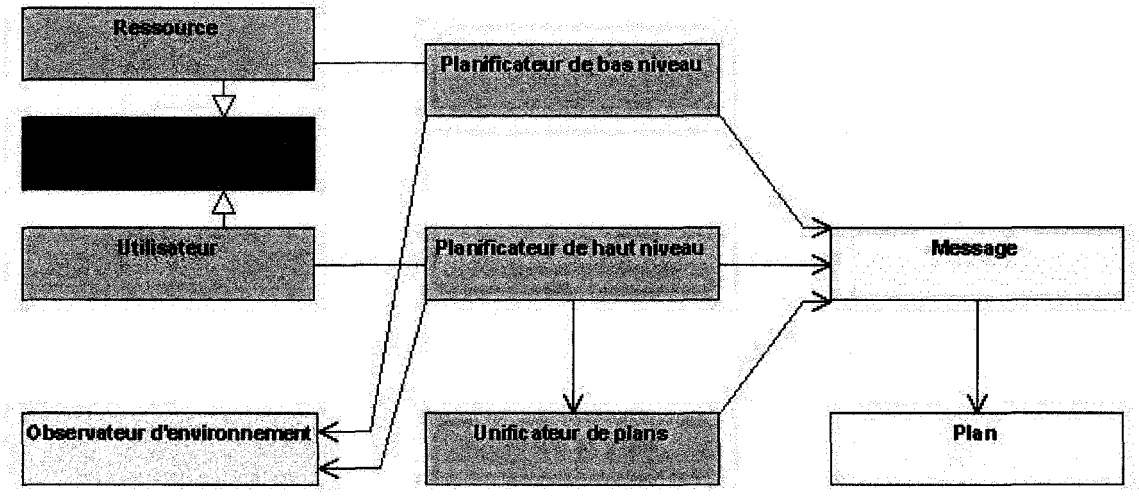


Figure 9 Schéma algorithmique du module *OptiEvents*.

En bleu sont représentées les trois classes servant à définir les agents. Les algorithmes de planification sont caractérisés dans les classes représentées en vert alors que l'OE est représenté en rouge. Les classes jaunes servent à la communication entre les agents.

Finalement, un dernier module fait partie du prototype : l'OE. Son fonctionnement est défini dans une classe à part. Les deux planificateurs ont accès régulièrement à cette classe lors de la planification d'actions. Les détails des différentes classes sont présentés à l'annexe 3.

4.6 Conclusion

Ce chapitre à présenté l'architecture générale du prototype qui fait l'objet de ce mémoire. Celui-ci prend la forme d'un système multiagent et tient compte des ressources disponibles, du temps pour se rendre à l'endroit où doit avoir lieu l'action, ainsi que des résultats de cette dernière. Une seconde action, plus simple à implémenter en temps réel, a été développée : l'interception d'un véhicule menaçant. De plus, le système a été conçu pour être facilement modifiable dans le futur. L'ensemble des algorithmes ayant été présenté, le chapitre suivant traitera des résultats obtenus avec le prototype.

CHAPITRE 5

RÉSULTATS ET DISCUSSION

La présente partie se veut le dévoilement des diverses expérimentations réalisées dans le cadre de ce mémoire. Il importe de rappeler les trois étapes constituant ce travail :

a. la génération de données de trafic;
b. la simulation d'un barrage routier;
c. la planification d'actions.

La planification d'actions renferme certes l'essentiel des résultats assimilables à ce mémoire. Néanmoins, les deux premières étapes doivent également attirer notre attention étant donné qu'elles ont toutes deux une incidence sur les résultats qui préoccupe cette recherche.

5.1 Génération des données de trafic

Pour simuler nos expérimentations, les données de trafic pour une ville entière sont nécessaires. C'est données n'étant pas disponibles, nous avons tenté de les reproduire à partir d'une méthode macroscopique de simulation du trafic automobile : la génération de trafic à partir d'une matrice OD. L'expérimentation consistait donc à reproduire le trafic automobile de la ville de Québec à l'heure de pointe matinale, soit entre 7 h 30 min et 9 h 00.

La figure 10 représente la somme des véhicules du réseau routier simulé. Les résultats utilisés pour la suite des travaux sont ceux compris entre la 60^{e} (7 h 00) et la 180^{e} minute (9 h 00). Les résultats obtenus avant la 60^{e} ne sont pas représentatifs : le réseau était dès

lors en phase de chargement. Quant à ceux obtenus après la 180^{e} minute, ils sont inutilisables tout autant puisque le réseau a cessé de se charger à 9 h 00.

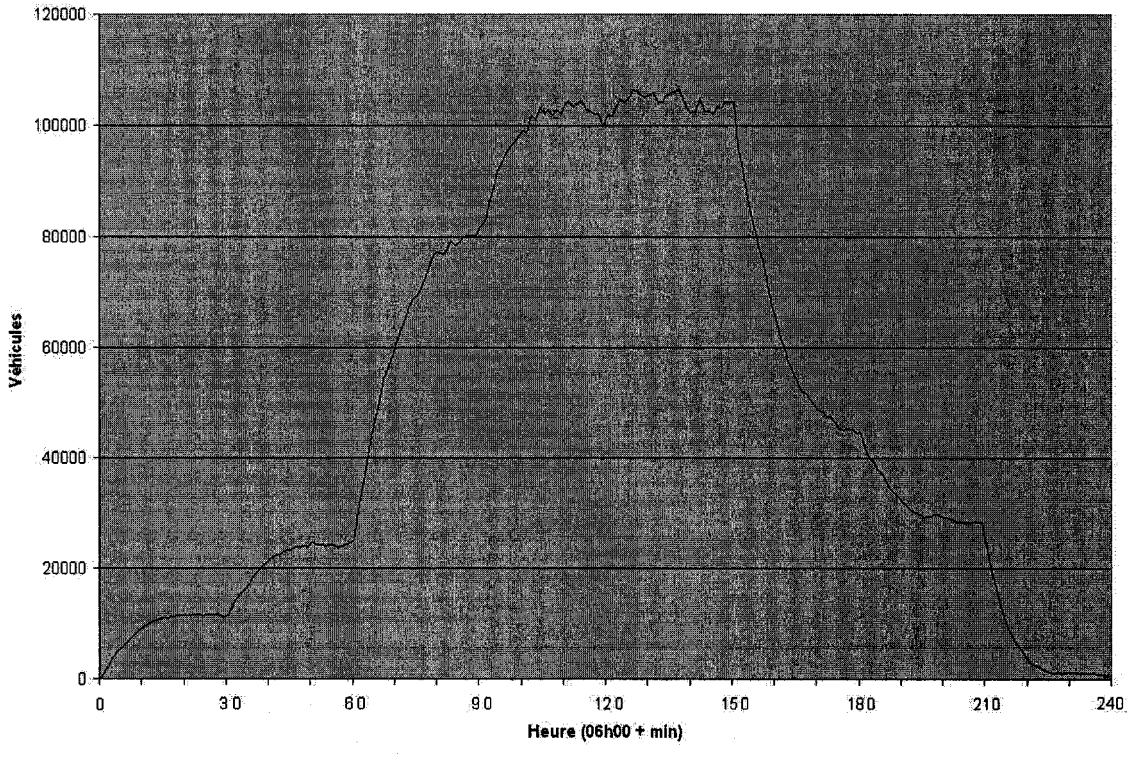


Figure 10 Chargement du réseau automobile.

Somme des véhicules présent dans le réseau routier entre la 6 h 00 et 10 h 00. Le réseau atteint sa congestion maximale entre 7 h 45 et 8 h 45 ou plus de 100 000 véhicules sont dans le réseau. Les résultats obtenus avant 7 h 00 (60^{e} minute) ne sont pas représentatifs : le réseau était dès lors en phase de chargement. Quant à ceux obtenus après 9 h 00 (180^{e} minute), ils sont inutilisables tout autant puisque le réseau a cessé de se charger à 9 h 00.

Les résultats ont été validés avec des comptages réels effectués par la ville de Québec. Le tableau suivant énumère quelques exemples de comparaison.

Tableau IV

Erreur relative entre les résultats de la simulation
de trafic et les données réelles de la ville de Québec

Tronçon	Réel (véhicules)	Simulation (véhicules)	Erreur relative (%)
Charest / Ouest de M. de l'incarnation	5185	2830	45,41
Charest / Est de M. de l'incarnation	4514	2288	49,31
Ch. Ste-Foy / Est de Du Vallon	2007	1848	7,92
Ch. Ste-Foy / Ouest de Du Vallon	2059	2858	-38,80
W-Hamel / Est de M. de l'incarnation	5229	3593	31,29
W-Hamel / Ouest de M de l'incarnation	3729	2192	41,21
1ère Ave / Nord de 18e Ave	1125	1331	-18,31

Notons que ces résultats de simulation sous-estiment généralement les données réelles. Plusieurs raisons peuvent expliquer cela :

a. la matrice OD est erronée;
b. les types de route (nombre de voies) sont partiellement erronés;
c. des secteurs ont été rejetés.

Une première raison, susceptible d'être responsable de ces erreurs, est la matrice OD. Cette dernière est à la base de la simulation. Bien que très précise à plusieurs égards, elle ne tient pas compte de certains types de véhicules. En effet, tous les véhicules provenant de l'extérieur et se dirigeant à l'extérieur de la ville ne sont pas comptabilisés dans la matrice. Pensons aux camions lourds, aux autobus interurbains ou simplement aux voyageurs en transit. De plus, les véhicules commerciaux ne sont pas répertoriés : autobus, taxi, camion, etc. Autant d'exclusions peuvent diminuer grandement le nombre de véhicules présents dans notre réseau simulé et restreindre les résultats obtenus.

De plus, le nombre de voies des tronçons peut être erroné. Les données du fichier à représentation spatiale ne contenaient pas d'indication sur le nombre de voies par route. Il a donc fallu approximé cette valeur à partir du type de route répertorié dans le fichier à représentation spatiale. Or, une telle valeur joue un rôle important dans la fonction d'Akcelik (équation 2.1) qui détermine le temps pour parcourir un tronçon. Un segment contenant deux voies plutôt qu'une, a une capacité supérieure à recevoir un flot de véhicules automobile. Des erreurs quant au nombre de voies ont pu forcer certains automobilistes virtuels à emprunter un chemin différent de celui emprunté dans la réalité, gonflant par le fait même les résultats sur certains tronçons et les diminuant sur d'autres. Cela expliquerait, en partie, les différences observées.

Finalement, tel que discuté au second chapitre, nous avons volontairement omis d'intégrer certaines banlieues pour lesquelles aucun graphe du réseau n'était disponible. Bien que ces régions génèrent peu de véhicules dans le réseau, environ 5% du total, il est évident que ce choix méthodologique tend à diminuer le nombre de véhicules répertorié dans les résultats.

L'objectif premier n'était pas de reproduire avec exactitude l'heure de pointe matinale, mais plutôt de peupler notre réseau afin de tester des algorithmes. Il serait inutile de tester un barrage routier dans une ville sans véhicule, il n'y aurait aucun changement à la situation, sinon qu'un segment de route serait bloqué. À l'inverse, avec les données recueillies par nos expérimentations, bien qu'elles diffèrent de la réalité, les changements apportés par nos actions seront facilement observables. Par conséquent, malgré le fait que les données de trafic dans les segments soient généralement inférieures à la réalité et compte tenu que ces données sont cohérentes, les résultats de simulation sont suffisants pour la poursuite des travaux.

5.2 Simulateurs de barrage routier

La première action choisie est un barrage routier. Ce choix semblait pertinent, puisque l'effet d'une telle action agit sur les deux facteurs pondérant notre graphe : navigabilité et menace. Ceci étant, la simulation d'un barrage routier s'est avérée plus difficile qu'initialement prévu. Deux difficultés majeures ont été rencontrées : la validation et le temps de simulation.

Voyons d'abord les résultats. La simulation a été effectuée sur un quadrilatère de la ville de Québec formé à l'ouest par l'autoroute Henri IV, au nord pas l'autoroute Charest, au sud par le boulevard Laurier et à l'est par la rue Myrand. Le barrage a été simulé sur l'autoroute du Vallon à la hauteur du chemin Ste-Foy. Les figures 11 et 12 illustrent le quadrilatère (en bleu) et le tronçon bloqué (en rouge).

Les figures 13 et 14 ont été générées par un algorithme simple mis au point afin de visualiser divers types de données. Les détails de cet algorithme sont à l'annexe 4. La qualité des images est approximative puisque l'information géographique est tirée du graphe de la ville et non du fichier à représentation spatiale. Ainsi, les routes sont représentées par des segments droits et jamais par des courbes. Les figures 13 et 14 représentent la même région que la figure 12 qui elle, provient du fichier à représentation spatiale.

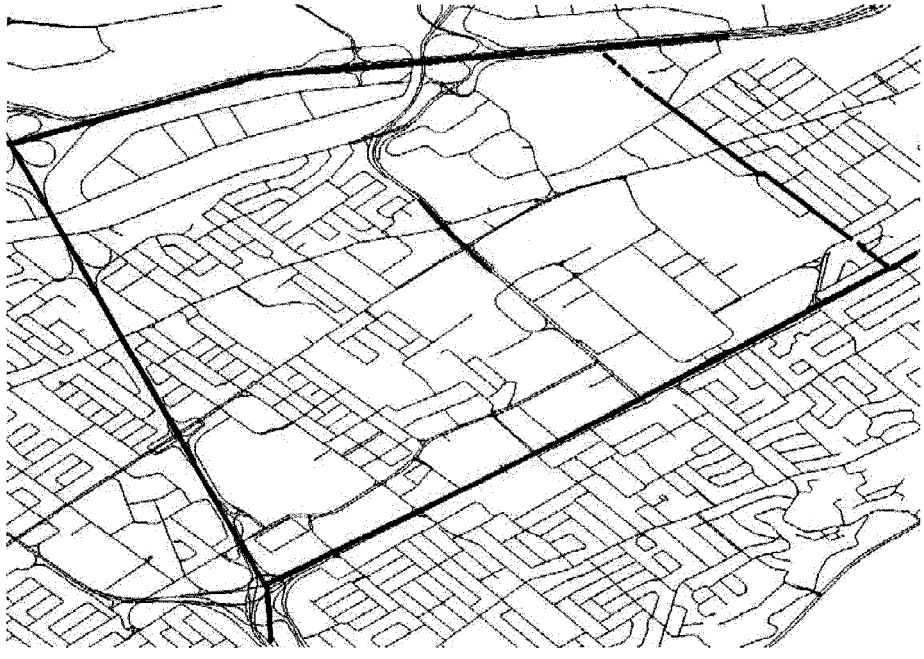


Figure 11 Région de simulation.

Région de la simulation du barrage routier (définie par le quadrilatère bleu); Le barrage routier est effectué sur le tronçon rouge.

Figure 12 Agrandissement de l'endroit où a lieu le barrage routier.

Les résultats de la simulation montrent que le barrage créé sur l'autoroute force les véhicules à utiliser la voie de service. Cette situation engendre une congestion et ralentit dramatiquement la vitesse dans ce segment. Les figures 13 et 14 représentent l'état de la circulation dans le réseau en deux temps, soit avant et après l'apparition du barrage routier. Notons que la couleur verte indique une circulation fluide alors que le rouge signifie une congestion. Dans l'image 14, le tronçon bleu identifie l'endroit où le barrage routier a pris forme : aucun véhicule n'y circule.

Figure 13 Temps 1. Visualisation du trafic avant la formation du barrage routier.

Un tronçon vert indique une circulation fluide alors qu'un rouge représente un tronçon congestionné.

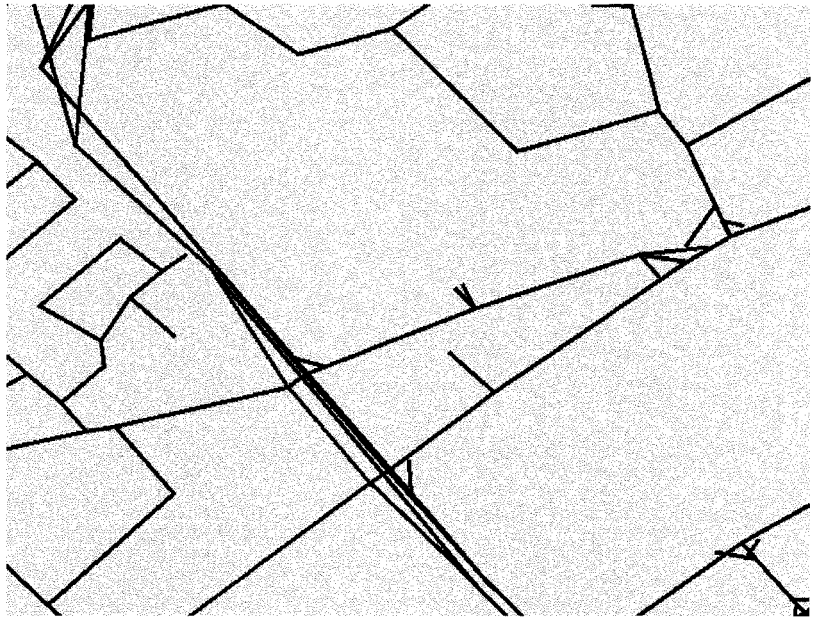


Figure 14 Temps 2. Visualisation du trafic après la formation du barrage routier.

Un tronçon vert indique une circulation fluide alors qu'un rouge représente un tronçon congestionné. Le tronçon bleu est l'endroit où a lieu le barrage routier.

Un l'outil servant à estimer l'emplacement futur d'un véhicule (TTDM) à été étudié au chapitre 4. Cet outil s'avère très utile à cette étape-ci, pour visualiser l'effet du barrage. Imaginons, par exemple, un véhicule qui est sur l'autoroute du Vallon et voudrait emprunter le segment bloqué par la présence du barrage pour se diriger vers le sud. En calculant sa TTDM, nous aurons une bonne idée de l'effet du barrage sur son déplacement. Les figures 15 et 16 illustrent cette logique. Le curseur bleu en haut à gauche des figures indique la position actuelle du véhicule. La couleur des segments indique le temps nécessaire pour s'y rendre à partir du point de départ. Plus le segment est bleu plus ce temps est élevé. La première figure présente les temps de déplacements du véhicule avant l'apparition du barrage alors que la seconde indique les temps approximés après la formation du barrage. Une nette différence est observable. Le véhicule est considérablement ralenti en empruntant la voie de service, rejoignant ainsi le reste du trafic dérouté.

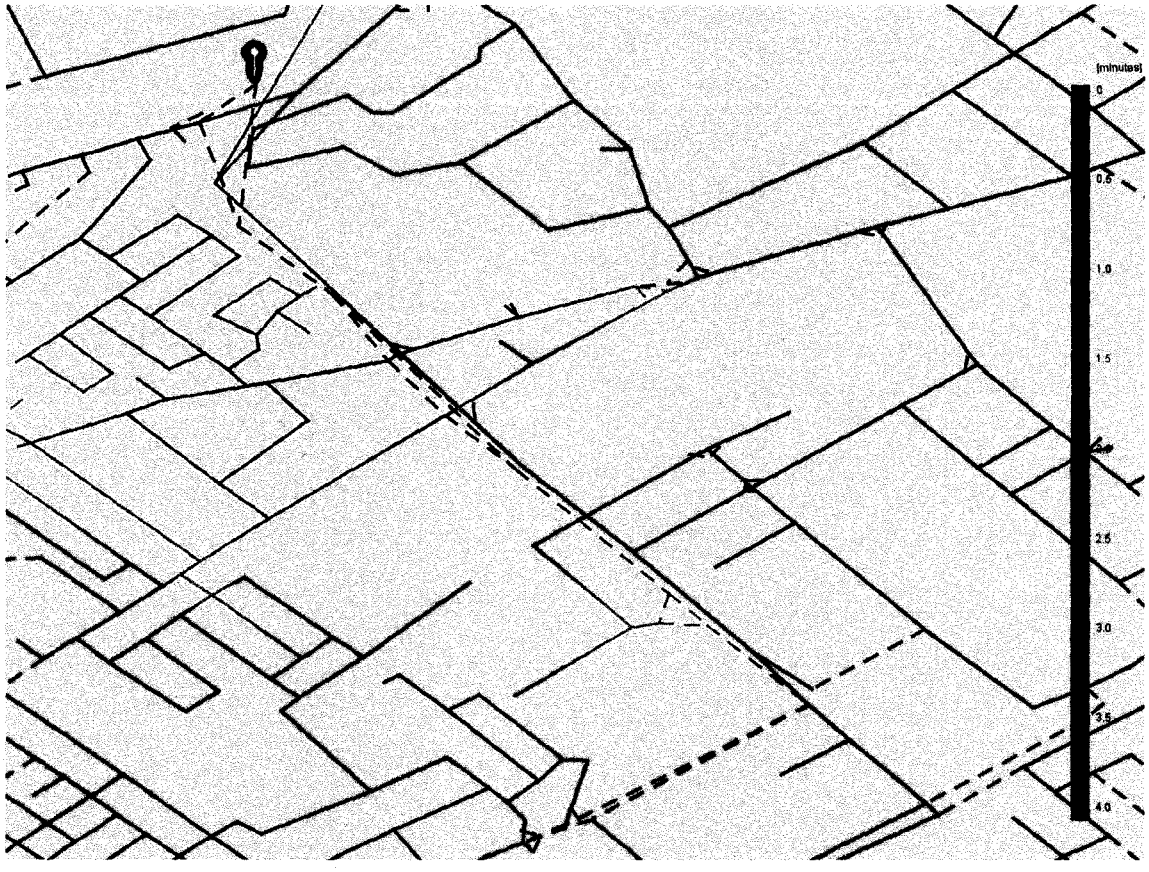


Figure 15 Visualisation de la TTDM, avant l'application du barrage routier.

Pour un véhicule se trouvant sur l'autoroute du Vallon, au nord de l'autoroute Charest (symbole bleu). La couleur des segments indique le temps minimal pour que ce véhicule se déplace jusqu'à cet endroit.

Figure 16 Visualisation de la TTDM, après l'application du barrage routier.

Pour un véhicule se trouvant sur l'autoroute du Vallon, au nord de l'autoroute Charest (symbole bleu). La couleur des segments indique le temps minimal pour que se véhicule se déplace jusqu'à cet endroit.

Le premier obstacle rencontré dans l'élaboration d'un simulateur de barrage routier est la difficulté de valider davantage les résultats. Pour pallier à cette faiblesse, il faudrait recréer dans la réalité un tel barrage et en vérifier les effets. Ce type d'expérimentation était difficile à réaliser dans le cadre de ce mémoire faute de ressources. Par conséquent, les résultats obtenus au cours de cette étude sont restreints par cette contrainte méthodologique.

Une deuxième difficulté est le temps de simulation. En effet, la simulation d'un barrage routier sur un processeur G5 cadencé à 1.8 GHz nécessite 95 secondes. Il est donc impossible de l'implémenter en temps réel pour le moment. Il n'empêche que les résultats obtenus par l'entremise de l'algorithme sont satisfaisants et cela laisse présager le potentiel de son utilisation dans le futur.

5.3 Planificateur

Le planificateur d'action doit fonctionner en temps réel. Pour cette raison, le simulateur de barrage routier ne peut en faire partie pour le moment. Par ailleurs, une seconde action a été développée au cours de ce travail, soit l'interception d'une menace. Ne nécessitant pas de simulation, cette action s'avère beaucoup plus facile à intégrer dans le système proposé.

Un environnement virtuel a aussi été conçu pour simuler le déplacement des *ressources*, des *menaces* et de l'utilisateur. Cet environnement vient combler le besoin d'avoir des véhicules se déplaçant dans la ville, équipés de systèmes de positionnement global. Cette condition était nécessaire pour tester les algorithmes de planification.

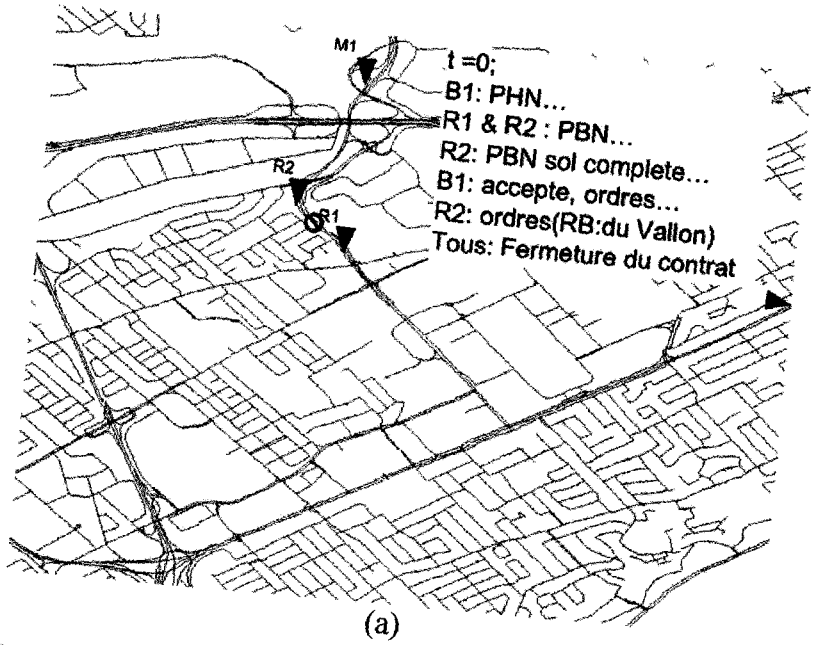
Plusieurs scénarios ont été observés. Deux retiendront l'attention, un simple et un second complexe faisant appel à plusieurs *ressources* essayant de contrer deux menaces distinctes.

Au début du premier scénario, l'utilisateur se déplace sur le boulevard Laurier en direction ouest lorsque le système aperçoit une *menace* qui se trouve sur l'autoroute du Vallon au nord de l'autoroute Charest. La situation est présentée sur la figure 17. L'utilisateur est représenté par la couleur bleu, les *menaces* par la couleur rouge et les *ressources* par le vert. Lorsqu'une *menace* est aperçue, le planificateur de haut niveau

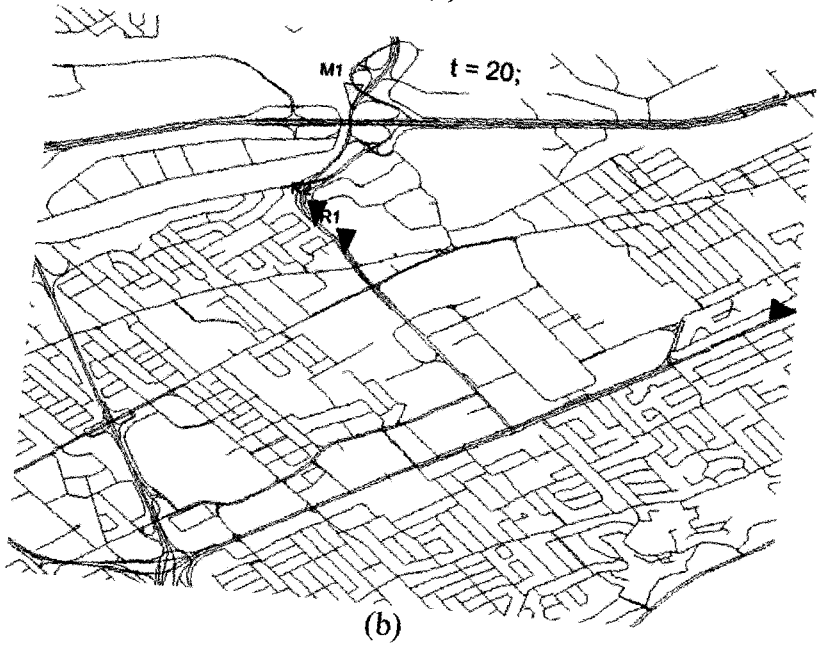
(PHN) identifie les différents endroits où cette menace peut venir intercepter l'utilisateur lors de son parcours. Le PHN demande alors aux différentes *ressources* de planifier des actions pour empêcher cette rencontre. Les deux *ressources* utilisent un planificateur de bas niveau (PBN) simulant une interception de menace. *R2* trouve une *solution complète* en interceptant la *menace* sur l'autoroute du Vallon. Si la *menace* (*m1*) passe par ce tronçon, elle sera interceptée. Sinon, *m1* ne pourra pas atteindre l'utilisateur par un autre chemin. Le PHN accepte et envoie les ordres tout en fermant le contrat. À partir de ce moment, le système ne sait plus où est la *menace*. La seule information qu'il détient est la dernière position connue de *m1*. Ainsi, le plan n'est pas modifié. Finalement, la *menace* est interceptée par *r2* après 80 secondes.

Le second scénario est plus complexe. Une *menace* (*m1*) est repérée sur la rue Dalquier alors que l'utilisateur roule sur le même parcours que précédemment (voir figure 18). Il est clair que *m1* sera beaucoup plus difficile à intercepter étant donné la quantité de rues par lesquelles elle peut circuler. Le PHN envoie son *annonce de tâche* aux trois *ressources*. Celles-ci étant incapables de trouver, chacune de son côté, une *solution complète*, elles renvoient leurs plans respectifs à l'UP. Ce dernier n'est pas plus en mesure de trouver une *solution complète* et choisit la meilleure *solution partielle* à partir des plans qu'il a fusionnés.

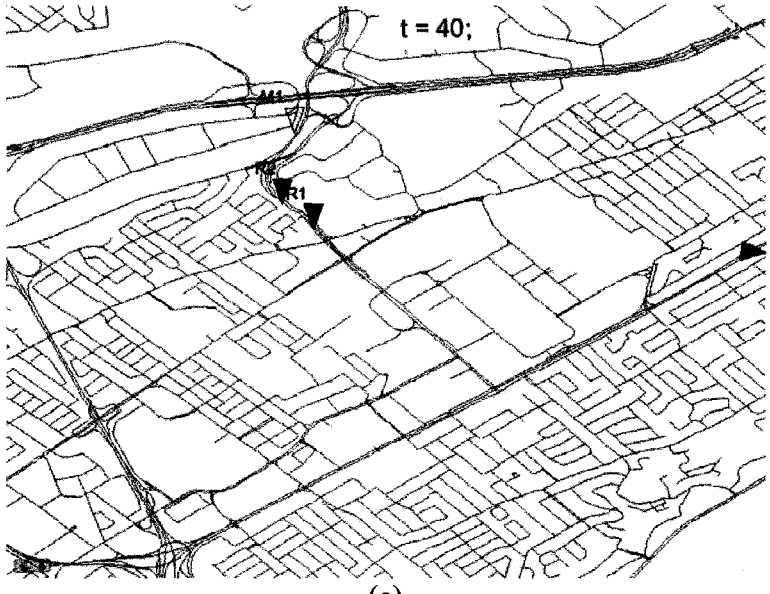
Une minute plus tard, alors que le plan est en cours d'exécution, une seconde *menace* (*m2*) est repérée sur l'autoroute du Vallon. Le PHN renvoie une nouvelle *annonce de tâche* qui tient compte de cette dernière information et de la dernière position connue de *m1*. Un nouveau *plan partiel* est trouvé, *r1* change dès lors l'endroit de son action. Finalement, *r1* réussira à intercepter *m2*, mais *m1* réussira à atteindre l'utilisateur sans être arrêtée.


M1
t =0;
B1: PHN...
R1 & R2 : PBN...
R2: PBN sol complete...
B1: accepte, ordres...
R2: ordres(RB:du Vallon)
Tous: Fermeture du contrat
R2
R1

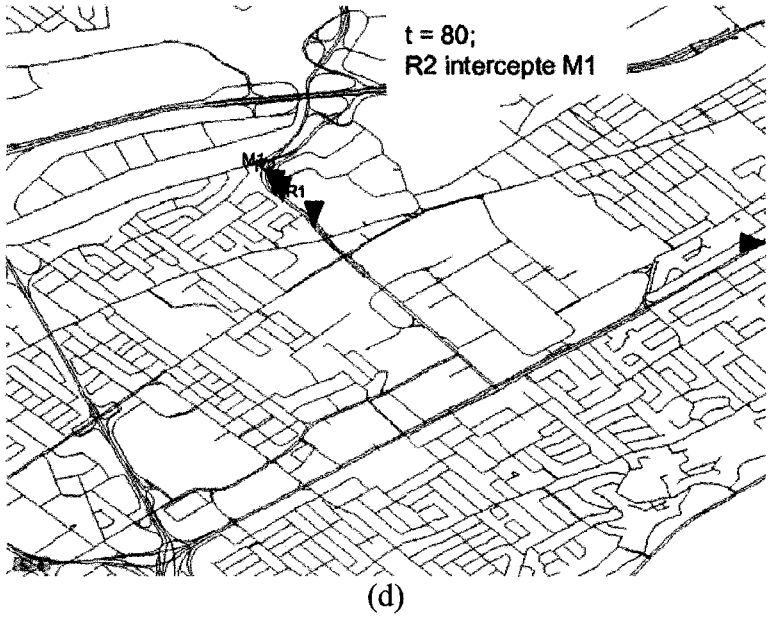
(a)


M1
t = 20;
R1

(b)



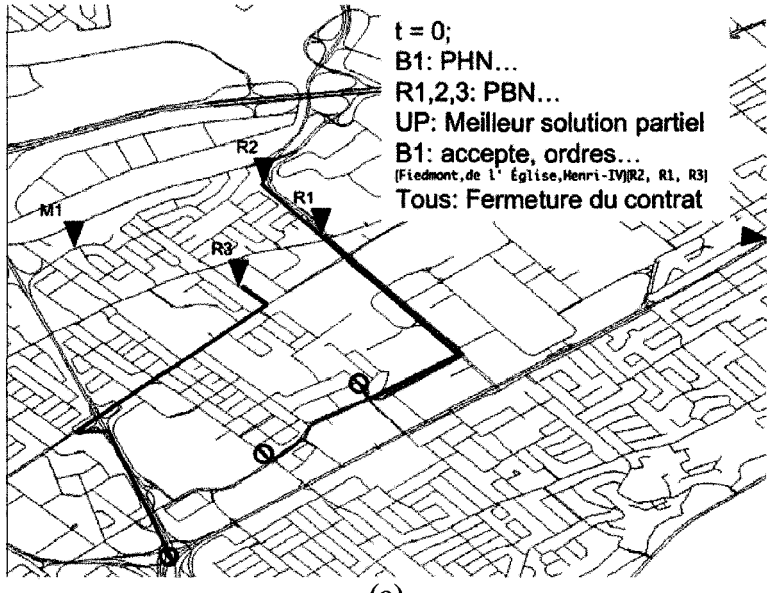
(c)



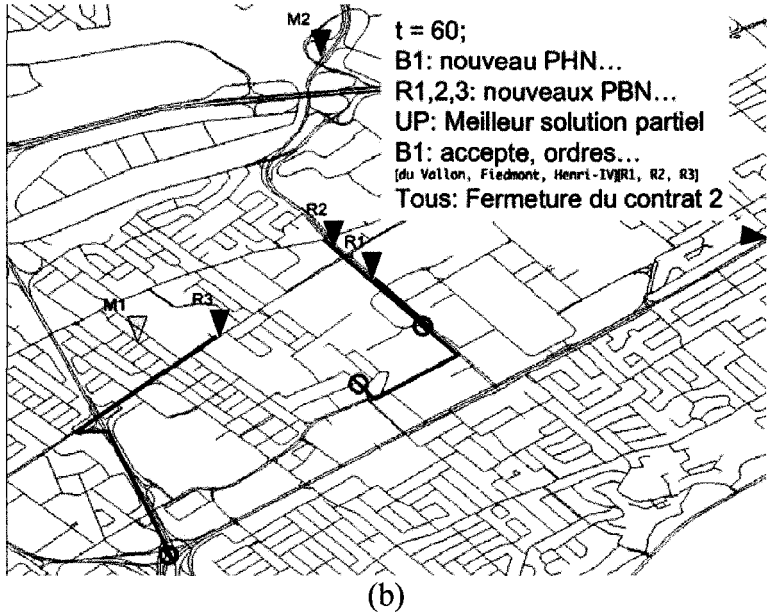
(d)

Figure 17 Premier scénario : une *menace* et deux *ressources*.

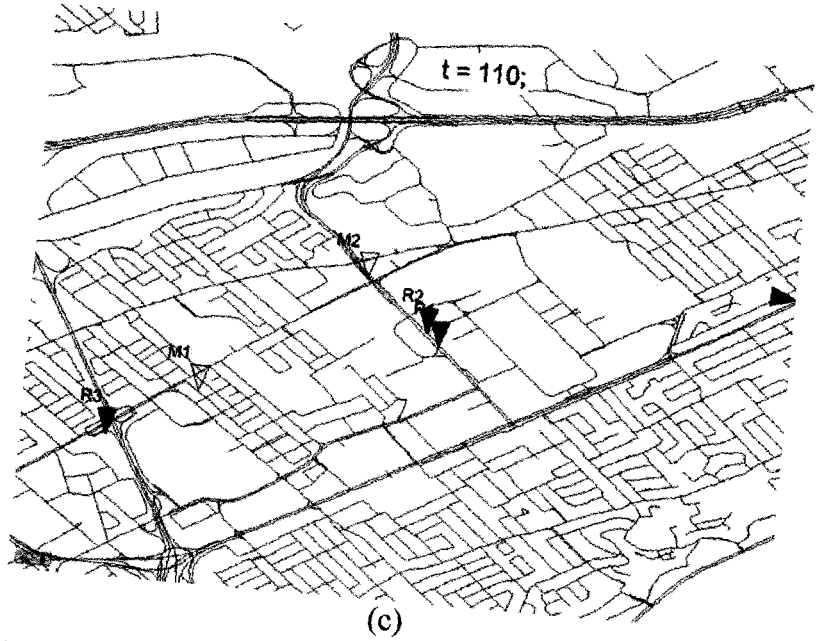
Un plan simple et complet est planifié et exécuté. *R2* intercepte la *menace* après 80 secondes. (a) situation à t = 0 seconde; (b) situation à t = 20 secondes; (c) situation à t = 40 secondes; (d) situation à t = 80 secondes.


t = 0;
B1: PHN...
R1,2,3: PBN...
UP: Meilleur solution partiel
B1: accepte, ordres...
[Fiedmont,de l' Église,Henri-IV][R2, R1, R3]
Tous: Fermeture du contrat
M1
R2
R1
R3

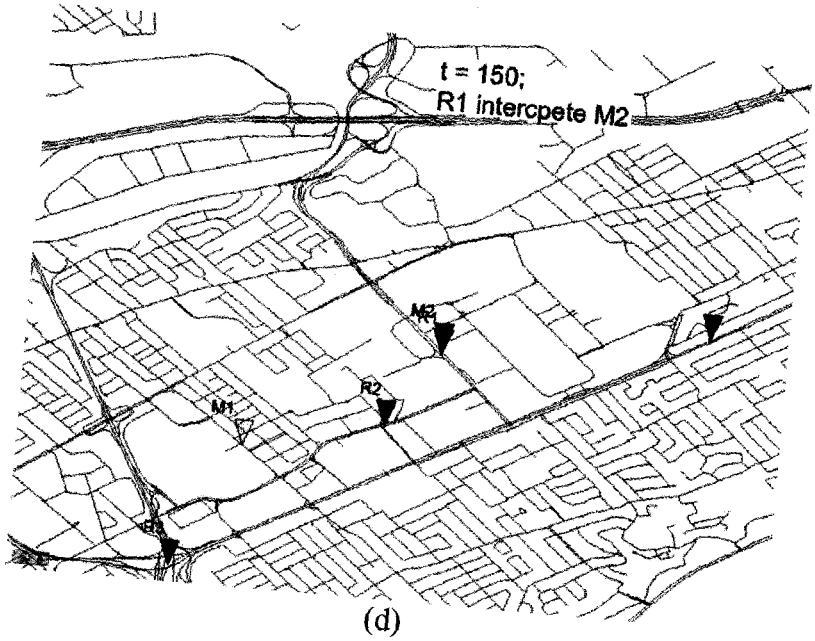
(a)


M2
t = 60;
B1: nouveau PHN...
R1,2,3: nouveaux PBN...
UP: Meilleur solution partiel
B1: accepte, ordres...
[du Vallon, Fiedmont, Henri-IV][R1, R2, R3]
Tous: Fermeture du contrat 2
R2
R1
M1
R3

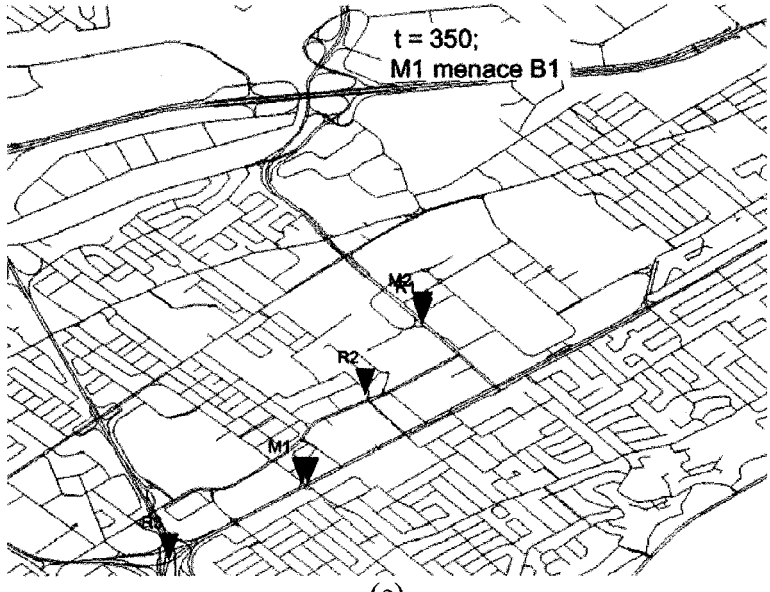
(b)



(c)



(d)



(e)

Figure 18 Second scénario : deux *menaces* aperçues à différents moments et trois *ressources*.

Le meilleur plan partiel est planifié et exécuté jusqu'à la 60^{ième} seconde où une seconde menace est repérée. À ce moment, un nouveau plan est trouvé. Une seule menace sera interceptée. (a) situation à t = 0; (b) situation à t = 60; (c) situation à t = 110; (d) situation à t = 150; (e) situation à t = 350.

Ce dernier scénario démontre la difficulté que peut représenter l'interception d'une ou de plusieurs menaces. Il aurait fallu deux fois plus de ressources afin d'obtenir un *plan complet*. Néanmoins, dans le cas présent, notre prototype a trouvé la meilleure solution possible.

5.4 Conclusion

Pour conclure cette section, le planificateur créé fonctionne et réussit à préparer des plans faisant appel à plusieurs ressources pour contrer de multiples menaces. Il le fait en temps réel avec jusqu'à 3 *ressources*. Il s'agit de la limite du système pour le moment puisque l'UP est incapable de trouver de bons plans dans un délai raisonnablement court

lorsqu'il y a plus de 3 ressources impliquées. Compte tenu de la structure actuelle de l'UP, fusionner des plans deviendrait problématique au-delà de ce nombre de *ressources*. Plusieurs solutions pourraient être envisageables pour éviter cela, sans pour autant avoir été développées dans le prototype. Parmi celles-ci, un système de classement des plans pourrait améliorer légèrement les performances. Ce classement pourrait être fait par ordre de pointage des plans, ou encore par TP améliorés. Distribuer la tâche d'unification des plans aux PBN pourrait être aussi une meilleure façon d'effectuer cette tâche. Dans ce cas, chacun des agents tenterait d'unir ses meilleurs plans avec ceux provenant d'autres agents. Cette implémentation en parallèle augmenterait de beaucoup les performances sur un superordinateur.

CHAPITRE 6

CONCLUSION ET RECOMMANDATIONS

Ce travail avait comme principal objectif le développement d'une architecture pour le module *OptiEvents* du projet *Scipio*. Ce module a pour fonction la création de plans améliorant un déplacement dans le cadre d'opérations militaires en milieu urbain. Un prototype devait montrer le potentiel futur d'une telle architecture.

La première partie de ce mémoire fut de créer un environnement adéquat à la réalisation des expérimentations. Pour ce faire, un graphe du réseau routier de la ville de Québec a été utilisé. Les données de trafic pour chacun des segments n'étant pas disponibles, quoiqu'essentielles, le réseau routier a été peuplé de véhicules à l'aide d'une méthode macroscopique : la simulation à partir d'une matrice origine-destination. Bien que le résultat de cette étape ne représente pas avec exactitude un moment précis de la journée, les données de trafic sont cohérentes et suffisantes pour tester les différents algorithmes. Il serait primordial de représenter avec précision la réalité dans l'éventualité où nous voudrions valider les résultats sur le terrain. Or, aux fins de ce mémoire, cette option a été rejetée, étant donné la complexité logistique que cela demandait. Néanmoins, il sera indispensable pour le projet Scipio d'obtenir ces données de trafic pour utiliser adéquatement les différents modules le constituant. Bien qu'il ne s'agisse pas d'un objectif de ce travail, ce mémoire a tout de même montré la complexité de générer ce genre d'information. De surcroît, ce travail avait l'avantage de travailler dans une ville connue et d'avoir des données très complètes pour simuler le trafic automobile. À l'inverse, dans le cas du projet Scipio, la ville sera la plupart du temps étrangère et les données de base seront absentes. Il serait donc préférable d'opter pour une méthode de collecte de données ayant lieu directement sur le terrain plutôt que de tenter de les générer. À cette fin, l'utilisation d'une flotte de véhicules tests circulant dans les rues de la ville serait envisageable, mais non sans certains problèmes. Une méthode utilisant des

photos aériennes ou satellites serait sans doute plus complexe à développer, mais les résultats seraient assurément plus précis et couvriraient plus rapidement l'ensemble de la ville.

Ce travail a aussi amené la réalisation d'un simulateur de barrage routier montrant les effets de cette action sur le trafic automobile et éventuellement sur un véhicule ennemi. Bien que cet algorithme soit plus lourd en calcul que cela était prévu, les résultats sont très satisfaisants. Tel que proposé, les simulations ont montré que cette action affecte directement le tronçon où elle est posée et indirectement les tronçons environnants par la présence accrue de véhicules déroutés du segment bloqué. Les effets d'un tel barrage sont exactement ceux que l'on souhaitait dans le module *OptiEvents* : ils sont capables de ralentir une menace se dirigeant vers une unité alliée. Deux bémols sont cependant à noter. Premièrement, il est impossible d'intégrer ce simulateur dans un système en temps réel étant donné sa complexité. Deuxièmement, une validation des résultats avec la réalité serait nécessaire afin d'assurer la justesse des effets simulés. En effet, bien que les résultats apparaissent satisfaisants, certains réglages sont à faire. La seule façon d'y parvenir serait de valider ce travail avec des données sur le terrain. Pour y parvenir, il faudrait premièrement avoir des données exactes sur l'état du trafic en temps normal. Par la suite, il serait possible d'utiliser des travaux routiers comme un premier test. Les travaux routiers sont similaires aux barrages routiers quant à leurs effets : ils bloquent un tronçon. Par contre, dans le cas des travaux, les automobilistes ont souvent accès à l'information portant sur la modification du réseau avant leur déplacement. Ainsi, les automobilistes peuvent modifier leur parcours dès le départ. Pour cette raison, la meilleure méthode serait de reproduire, avec l'accord des forces de l'ordre locales, un véritable barrage routier.

Finalement, l'étape essentielle de ce mémoire a été la conception d'un prototype pour le module *OptiEvents*. Puisqu'il était impossible d'utiliser le simulateur de barrage routier, le prototype a employé une seconde action, celle-ci ne nécessitant pas de simulation :

l'interception d'un véhicule ennemi. Toutefois, la structure est en mesure d'accueillir d'autres actions, tel un barrage routier, si ceux-ci rencontrent les exigences de performance nécessaire. Ce mémoire a mis en évidence le potentiel du système actuel en utilisant différents scénarios simulés par ordinateur. La structure actuelle du prototype est celle d'un système multiagent, favorisant ainsi les performances et la polyvalence des situations. Les agents sont les différentes unités sur le terrain pouvant accomplir les actions ainsi que les utilisateurs du système. Le prototype peut répondre à de multiples menaces et prendre en charge jusqu'à trois ressources et un utilisateur. Une future version du module devrait être en mesure d'utiliser plus encore de ressources et répondre à de multiples utilisateurs. Pour ce faire, l'architecture de l'algorithme servant à fusionner les plans (UP) sera, dans l'avenir, à améliorer afin d'augmenter son efficacité. L'implémentation en parallèle de cette partie devrait être très sérieusement envisagée pour y arriver. De plus, un protocole de négociation entre les différents utilisateurs se partageant les ressources devrait également être mis au point puisque pour l'instant le prototype ne peut répondre qu'à un utilisateur. Le temps d'accomplissement d'une action devra aussi être implémenté afin qu'une unité puisse accomplir une seconde action après en avoir complété une première. Une fois ces dernières améliorations apportées au système, un premier utilisateur sera en mesure d'utiliser une unité pour accomplir une action avant qu'un second utilisateur l'utilise quelques instants plus tard dans un autre plan. Finalement, il serait très intéressant de calculer la valeur (pointage) des plans en fonction d'autre chose que des résultats simulés, par exemple le risque, la complexité ou encore le déplacement. Ainsi, une action simple, sans danger et effectué sans un grand déplacement sera préférée à une seconde action plus complexe.

Bien qu'il ne s'agisse que d'une première esquisse du module *OptiEvents*, le prototype produit dans ce travail est révélateur. Il montre la contribution que pourrait apporter un tel système aux militaires. Rapidement, il simule certaines actions et construit des plans dans un environnement complexe et dynamique. De telles opérations prendraient un temps énorme, et combien crucial dans certaines situations si elles étaient réalisées par

un être humain. Bien qu'il ne se veuille pas le remplaçant des commandants des armées de demain, un tel système pourrait leur offrir une assistance incomparable. Pour cette raison, il est certainement voué à un avenir prometteur et trouvera certes sa place au sein des futures armées.

ANNEXE 1

Implémentation JAVA de la génération de trafic automobile

La génération du trafic automobile pour une ville entière est une tâche extrêmement laborieuse, d'où la nécessité d'avoir recours à une implémentation multitâche. On distingue deux groupes de tâches répétitives dans l'algorithme décrit à la figure 3 : le calcul du chemin le plus court pour une paire OD et l'assignation des véhicules dans les segments formant ces chemins. Dans notre cas, il y a 1849 paires OD et donc autant de chemins à trouver et de vagues de véhicules à assigner à chacune des itérations. La figure 19 illustre les différentes classes intervenant dans le processus d'assignation du trafic automobile.

La classe *RealCity* est le point central de l'assignation. Son rôle consiste à appeler les différentes tâches (encadrées dans la figure 4), soit la recherche du chemin le plus court et la modification des données de trafic. La classe *RealCity* est elle-même appelée par *MyMain*, qui sert à lire les données contenues dans le graphe de la ville. Cette étape est effectuée par la classe *JGraphTReader*. *MyMain* sert aussi à construire les différentes classes utilisées par l'algorithme : *lookUpTableOfTravelTime*, *traficMatrix* et *ODMatrix*. Finalement, *MyMain* effectue, en appelant la classe *SecteurAssignation*, le classement des segments en 43 secteurs correspondant aux données de la matrice OD.

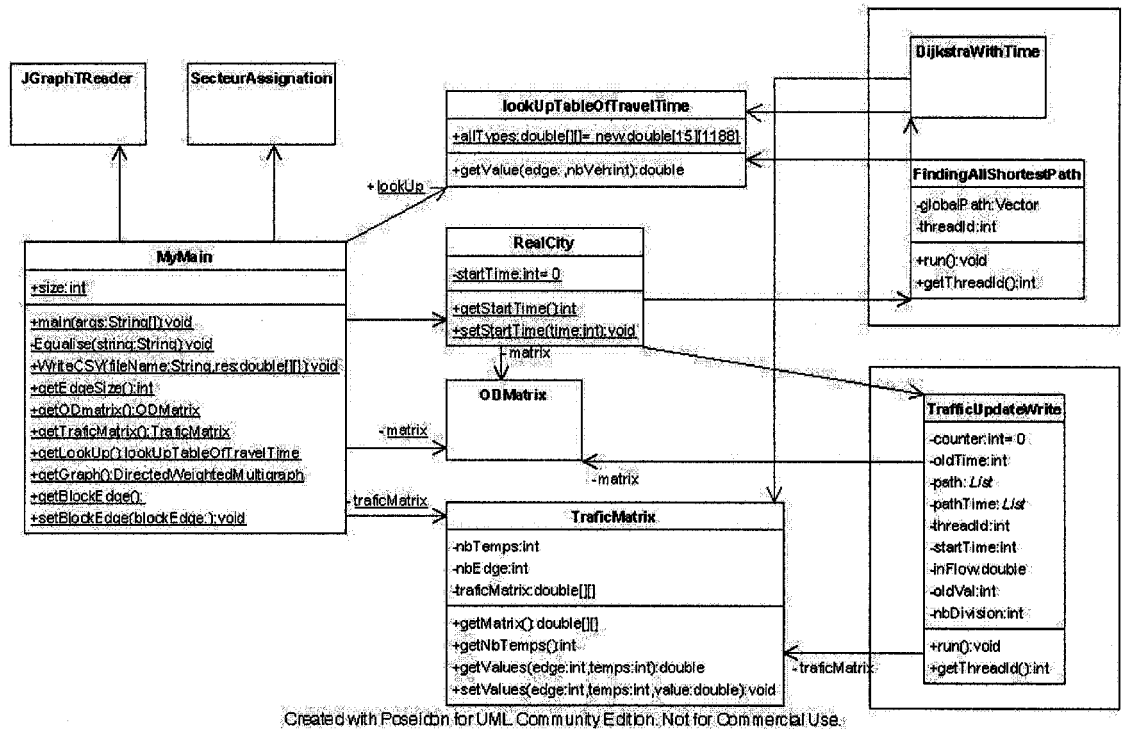


Figure 19 Schéma des classes utilisées dans l'implémentation de l'assignation du trafic automobile.

Lorsque la tâche *FindingAllShortestPath* est lancée, elle fait en parallèle 1849 appels à la classe de recherche du chemin le plus court : *Dijkstra*. Cette classe est une version modifiée de l'algorithme de *Dijkstra [43]* afin d'y incorporer les données de trafic de telle sorte qu'elle puisse pondérer correctement chacun des tronçons. En effet, le temps pour parcourir un segment dépend directement du nombre de voitures dans celui-ci. C'est pourquoi la classe *Dijkstra* doit faire appel à *traficMatrix* pour obtenir le nombre de véhicule dans le segment et à *lookUpTableOfTravelTime* pour trouver le temps de parcours du segment. En effet, une table de conversion (*lookUpTableOfTravelTime*) a été implémentée afin d'augmenter la vitesse de traitement. À chacun des tronçons évalués par l'algorithme de *Dijstra*, celui-ci doit convertir le nombre de voitures dans le tronçon (provenant de la classe *traficMatrix*) en un temps de parcours. Pour ce faire, on a généralement recours à l'équation 2.1. Cette équation étant très fastidieuse à calculer,

la table de conversion améliore considérablement la rapidité d'exécution en remplaçant la formule mathématique.

Lorsque tous les chemins ont été trouvés, ils sont inscrits dans un vecteur qui est passé en argument à la classe *TrafficUpdateWrite*. Cette classe a pour but de mettre à jour en parallèle le nombre de véhicules dans notre réseau (*TraficMatrix)*. Une fois cette étape complétée, *RealCity* vérifie s'il y a d'autres itérations à exécuter et si c'est le cas, relance *FindingAllShortestPath.*

ANNEXE 2

Implémentation JAVA de la simulation d'un barrage routier

L'implémentation en langage JAVA du simulateur de barrage routier ne comporte que deux classes, *MyMain* et *RoadBlock2*. La figure 20 les montre en format UML.

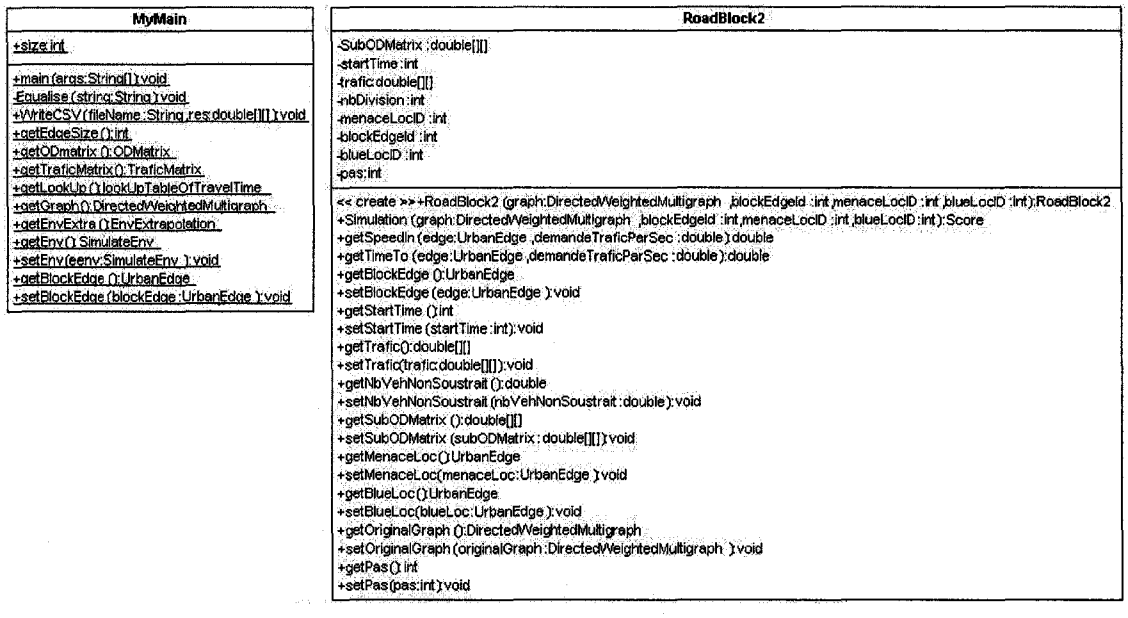


Figure 20 Schéma des classes utilisées dans l'implémentation du simulateur de barrage routier.

La classe *MyMain* sert à charger différentes ressources en mémoire : graphe, matrice de trafic, position du barrage routier. Elle ne fait pas véritablement partie du simulateur, mais commande la simulation du barrage routier en construisant la classe *RoadBlock2*. La simulation a proprement dite est effectuée par cette classe d'une manière séquentielle. Cette façon de faire affecte grandement la rapidité de simulation. Il est impossible d'effectuer ces tâches en parallèle puisque chacune est dépendante du résultat de la précédente.

ANNEXE 3

Implémentation JAVA du module OptiEvents

L'implémentation JAVA du module *OptiEvents* peut être représentée par le diagramme de la figure 21. La classe *HighLevelPlanner* correspond au PHN. Cette classe comporte l'attribut *cheminBleu* qui définit le chemin emprunté par l'utilisateur. Pour sa part, *menaceVertexTable* représente les différentes menaces affectant les tronçons de ce chemin. C'est à partir de cet attribut que le PHN formule son *annonce de tâche* pour les PBN.

Les PBN sont construits à partir de la classe *LowLevelPlanner*. Son code est plus séquentiel que le PHN puisqu'il correspond à une action en particulier. Lorsqu'un agent de type *ressource* reçoit l'*annonce de tâche,* il construit différents PBN en leur donnant les TP répertoriés dans la liste *pointProblématique*.

Quant à *PlanMerger* la classe construisant l'UP, elle a comme attribut la liste *listePlanPartielle* qui servira à contenir les plans partiels qui lui seront envoyés par les PBN. À partir du moment où cette liste comporte plus d'un plan, l'UP s'active et commence à chercher des plans à fusionner.

Notons aussi que les classes *EnvExBlue* (utilisateur) et *EnvExRess* (*ressource*) héritent toutes deux de la classe *EnvExAgent*. C'est dans cette dernière classe que le comportement général des agents est défini. Par exemple, l'envoi et la réception d'un message y est décrit. Ceux-ci se font via la liste *inbox* qui sert à emmagasiner les messages reçus. Lorsque cette liste comporte des éléments, l'agent les ouvre et effectue les opérations en conséquence. Par exemple, si un agent de type *ressource* reçoit un message d'*annonce de tâche*, il démarrera des PBN. Les messages lus sont par la suite déplacés vers la liste *inboxOpened*. Le reste des attributs des agents sont définis dans les classes *EnvExRess* et *EnvExBlue*.

La classe *message* comporte les différents attributs nécessaires à la communication entre deux agents. La liste *to* est l'ensemble des destinataires du message, alors que l'attribut *contrat* est le numéro de contrat relatif au message. Il comporte aussi la table de hachage

pointProblématiqueTable qui est utilisée par le PHN dans son *annonce de tâche* pour définir les points à améliorer. Cette classe contient aussi l'opération *getThisPlan* qui réfère à un plan. Lorsqu'un agent communique un plan à un autre agent, cette opération réfère un objet informatique construit à partir de la classe *Plan*. Cet objet comporte les attributs relatifs au pointage du plan (*score*), aux ressources utilisées (*ressourceUsed*), aux sous-plans (*subPlan*) ainsi qu'aux TP améliorés par ce plan (*resolvedVertex*).

Finalement, la classe *EnvExtrapolation* correspond à l'OE. Ses attributs permettent de trouver quels agents sont sur le terrain puisqu'ils sont répertoriés dans la liste *unitNameList*. Il comporte aussi les données de trafic automobile (*trafic*) ainsi que la position des menaces (*menaceVertexTable*).

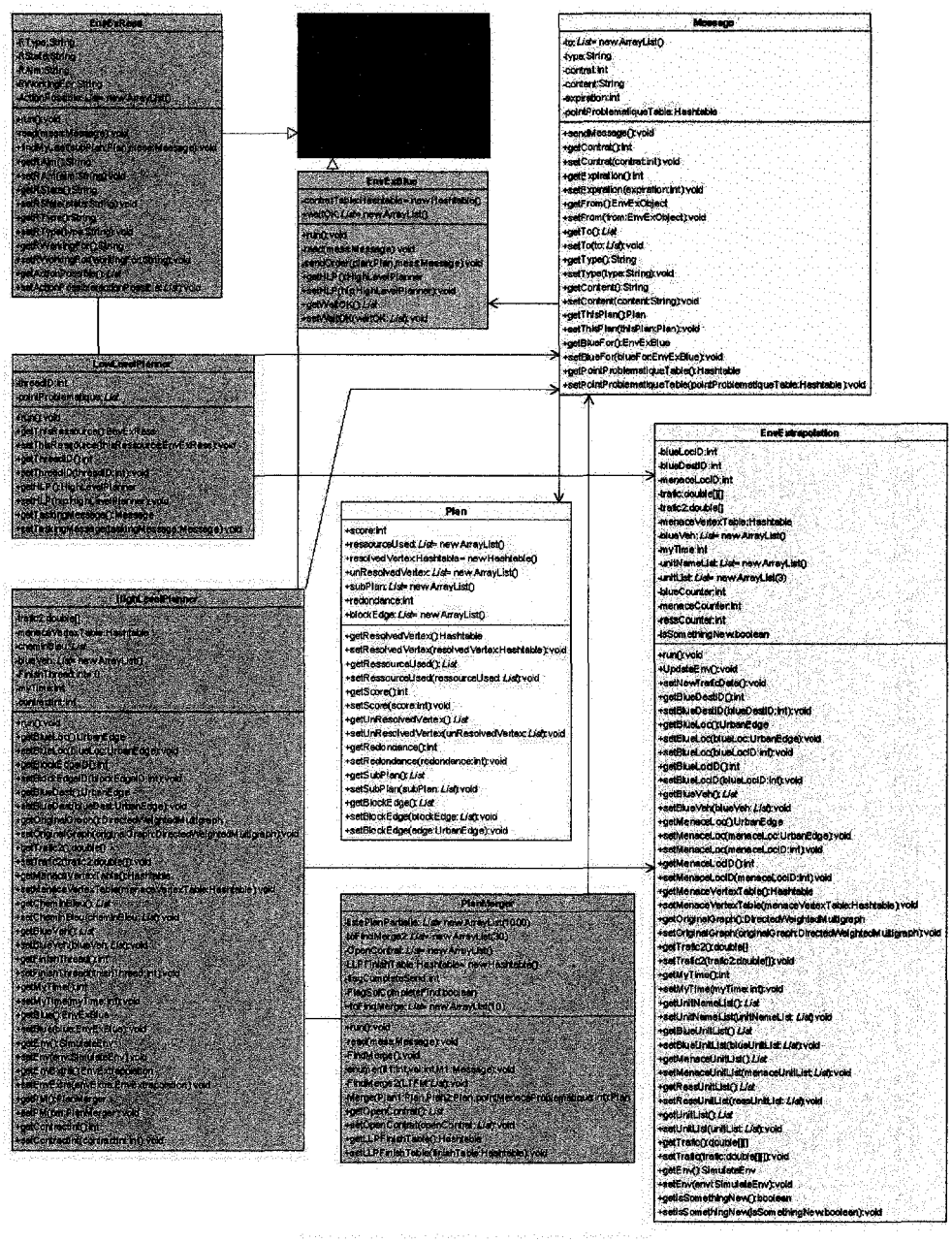


Figure 21 Schéma des classes utilisées dans le module *OptiEvents*.

En bleu sont représentées les trois classes servant à définir les agents. Les algorithmes de planification sont caractérisés dans les classes représentées en vert alors que l'OE est représenté en rouge. Les classes jaunes servent à la communication entre les agents.

ANNEXE 4

Algorithme de visualisation

Un problème rencontré lors de la validation des résultats est la lecture des données obtenues. Dans la simulation d'un barrage routier, il faut vérifier la modification des données de trafic, c'est-à-dire le nombre de véhicules dans des segments de route précis, avant et après l'application du barrage. Notre ville comptant 38 000 segments, les résultats sont sauvegardés dans un tableau ayant 38 000 lignes. Dans ce format, il devient très fastidieux de comparer les données. Pour éviter ce travail, un algorithme pouvant représenter graphiquement ces données a été mis au point.

Cet algorithme prend en entrée deux points qui constitueront les deux coins opposés du secteur à afficher ainsi qu'un tableau segment-valeur. Le tableau peut-être, par exemple, celui contenant le nombre de véhicules pour les 38 000 segments discuté précédemment, ou encore une TTDM. L'algorithme d'affichage parcourt l'ensemble des segments du graphe et affiche ceux contenus dans les secteurs à afficher, tel que précisé à l'appel de la fonction. Le graphe ne contenant que les données géographiques des deux extrémités de chacun des segments, l'algorithme trace une ligne droite entre ces points. Le résultat est une approximation du réseau puisqu'aucune courbe n'est représentée. On peut constater cette approximation en comparant la figure 12 et la figure 13. La première est la représentation la plus fidèle d'un cartier de la ville de Québec alors que la seconde est son approximation à l'aide de cette méthode. Par la suite, l'algorithme utilise le tableau transmis pour choisir la couleur qu'auront les segments. Par exemple, dans le cas de la vérification de congestion, la couleur est fonction du nombre de véhicules dans le réseau (tel que défini dans la table) divisé par la capacité de ce même segment.

À l'aide de cet algorithme simple, il devient plus facile d'analyser et d'exposer les résultats de nos expérimentations.

BIBLIOGRAPHIE

[1] Hill, J. M. D., Miller, M. S., Yen, J., & Pooch, U. W. (2000). *Tactical event resolution using software agents, crisp rules, and a genetic algorithm.* Paper presented at the Proceedings of the Military, Government and Aerospace Simulation, 16-20 April 2000, Washington, DC, USA.

[2] Hill, J. M. D., Surdu, J. R., & Pooch, U. W. (2000). Implementation of the Anticipatory Planning Support System. *Proceedings Advanced Simulation Technologies Conference (ASTC'2001): Military, Government, and Aerospace Simulation Symposium*, 76-81.

[3] Lee, J. J., & Fishwick, P. A. (1994). Real-time simulation-based planning for computer generated force simulation. *Simulation, 63*(5), 299.

[4] Mason, C. R., & Moffat, J. (2000). *Representing the C2 process in simulations: modelling the human decision-maker.* Paper presented at the Proceedings of WSC 2000, Winter Simulation Conference, 10-13 Dec. 2000, Orlando, FL, USA.

[5] Schwalm, N. D., Samet, M. G., Silver, J., & Novick, Y. (1994). *Scenario generation for planning and execution monitoring.* Paper presented at the Military Communications Conference, 1994. MILCOM '94. Conference Record, 1994 IEEE.

[6] Hayes, C. C., Schlabach, J. L., & Fiebig, C. B. (1998). *FOX-GA: an intelligent planning and decision support tool.* Paper presented at the 1998 IEEE International Conference on Systems, Man, and Cybernetics.

[7] Orvosh, D., & Davis, L. (1994). *Using a genetic algorithm to optimize problems with feasibility constraints.* Paper presented at the IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on Evolutionary Computation. .

[8] Suantak, L., Hillis, D., Schlabach, J., Rozenblit, J. W., & Barnes, M. (2003). *A coevolutionary approach to course of action generation and visualization in multi-sided conflicts.* Paper presented at the 2003 IEEE International Conference on Systems, Man, and Cybernetics.

[9] Kewley, R. H., & Embrechts, M. J. (2002). Computational military tactical planning system. *Systems, Man and Cybernetics, Part C, IEEE Transactions on, 32*(2), 161-171.

[10] Cox, J. S., & Durfee, E. H. (2003). *Discovering and Exploiting Synergy between Hierarchical Planning Agents.* Paper presented at the Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 03, Jul 14-18 2003, Melbourne, Vic., Australia.

[11] Pappachan, P. M., & Durfee, E. H. (2000). *Interleaved plan coordination and execution in dynamic multi-agent domains.* Paper presented at the Proceedings Fourth International Conference on MultiAgent Systems, 10-12 July 2000, Boston, MA, USA.

[12] Wilkins, D. E., & Myers, K. L. (1998). *A multiagent planning architecture.* Paper presented at the Proceedings of Fourth International Conference on Artifical Intelligence Planning Systems, 1998, Pittsburgh, PA, USA.

[13] Clement, B. J., & Durfee, E. H. (1999). Top-down search for coordinating the hierarchical plans of multiple agents. *Proceedings of the 1999 3rd International Conference on Autonomous Agents, May 1-May 5 1999*, 252-259.

[14] Cox, J. S., & Durfee, E. H. (2004). *Efficient mechanisms for multiagent plan merging.* Paper presented at the Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2004, Jul 19-23 2004, New York, NY, United States.

[15] Russell, S. J., & Norvig, P. (1995). *Artificial intelligence : a modern approach* (Upper Saddle River, N.J. : Prentice-Hall ed.).

[16] Li, Y., Ma, S., Li, W., & Wang, H. (2003). *Microscopic urban traffic simulation with multi-agent system.* Paper presented at the ICICS-PCM 2003. Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing and Fourth Pacific-Rim Conference on Multimedia, 15-18 Dec. 2003, Singapore.

[17] Yang, Q., & Koutsopoulos, H. N. (1996). Microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies, 4*(3), 113-129.

[18] Ministère des Transports du Québec
http://www1.mtq.gouv.qc.ca/fr/services/documentation/statistiques/index.asp

[19] Binetti, M., & De Mitri, M. (2002). *Traffic assignment model with fuzzy travel cost.* Paper presented at the Proceedings of the 13th Mini-EURO Conference, Polytechnic University of Bari.

[20] Kikuchi, S., Nanda, R., & Perincherry, V. (1993). Method to estimate trip o-d patterns using a neural network approach. *Transportation Planning and Technology, 17*(1), 51.

[21] Kikuchi, S., & Tanaka, M. (2000). Estimating an origin-destination table under repeated counts of in-out volumes at highway ramps: Use of artificial neural networks. *Transportation Research Record*(1739), 59-66.

[22] Robillard, P. (1975). Estimating the OD Matrix from Observed Link Volumes. *Transportation Research, 9*, 123–128.

[23] Nanda, R., & Kikuchi, S. (1993,). *Estimation of trip O-D matrix when input and output are fuzzy.* Paper presented at the 1993 (2nd) International Symposium on Uncertainty Modeling and Analysis, 25-28 April 1993, College Park, MD, USA.

[24] Govind, S., Ardekani, S. A., & Kazmi, A. (1999). A PC-based decision tool for roadway incident management. *Computer-Aided Civil and Infrastructure Engineering, 14*(4), 299-307.

[25] Teodorovic, D., & Kikuchi, S. (1991). *Transportation route choice model using fuzzy inference technique.* Paper presented at the Proceedings. First International Symposium on Uncertainty Modeling and Analysis (Cat. No.90TH0334-3), 3-5 Dec. 1990, College Park, MD, USA.

[26] Wahle, J., Annen, O., Schuster, C., Neubert, L., & Schreckenberg, M. (2001). A dynamic route guidance system based on real traffic data. *European Journal of Operational Research, 131*(2), 302-308.

[27] Damberg, O., Lundgren, J. T., & Patriksson, M. (1996). Algorithm for the stochastic user equilibrium problem. *Transportation Research, Part B: Methodological, 30B*(2), 115-131.

[28] Han, S. (2003). Dynamic traffic modelling and dynamic stochastic user equilibrium assignment for general road networks. *Transportation Research Part B: Methodological, 37*(3), 225-249.

[29] Nielsen, O. A., Frederiksen, R. D., & Simonsen, N. (1998). Stochastic user equilibrium traffic assignment with turn-delays in intersections. *International Transactions in Operational Research, 5*(6), 555-568.

[30] Akiyama, T., & Okushima, M. (2002). Formulations of Traffic Assignment Model with Hybrid Travel Time Description. *Proceedings of SICS & ISIS, 2002.*

[31] Liao, H.-L., & Wang, H.-F. (1996). Application of variational inequality to user equilibrium problem in traffic assignment with fuzzy cost and demand functions. *Journal of Chinese Fuzzy Systems Association, 2*(2), 39-47.

[32] Sunachi, Y., & Imaizumi, M. (1994). *Reflected type optical vehicle sensor.* Paper presented at the Proceedings of the 1994 Vehicle Navigation and Information Systems Conference, Aug 31-Sep 2 1994, Yokohama, Jpn.

[33] Yoshida, M., & Aoyama, K. (1993). *New concepts for UTMS [traffic control].* Paper presented at the Vehicle Navigation and Information Systems Conference, 1993., Proceedings of the IEEE-IEE.

[34] Kim, K.-H., Lee, J.-H., & Lee, B. G. (1997). *Congestion data acquisition using high resolution satellite imagery and frequency analysis techniques.* Paper presented at the Geoscience and Remote Sensing, 1997. IGARSS '97. 'Remote Sensing - A Scientific Vision for Sustainable Development'., 1997 IEEE International.

[35] Schreuder, M., Hoogendoorn, S. P., Van Zulyen, H. J., Gorte, B., & Vosselman, G. (2003). *Traffic data collection from aerial imagery.* Paper presented at the Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems, 12-15 Oct. 2003, Shanghai, China.

[36] Breit, H., Eineder, M., Holzner, J., Runge, H., & Bamler, R. (2003). *Traffic monitoring using SRTM along-track interferometry.* Paper presented at the Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International.

[37] Ruhe, M. H. O., Dalaff, C., & Kuhne, R. D. (2003). *Traffic monitoring and traffic flow measurement by remote sensing systems.* Paper presented at the Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE.

[38] Niver, E., Mouskos, K. C., Batz, T., & Dwyer, P. (2000). Evaluation of the TRANSCOM's system for managing incidents and traffic (TRANSMIT). *Intelligent Transportation Systems, IEEE Transactions on, 1*(1), 15-31.

[39] Lu, J., & Cao, L. (2003). *Congestion evaluation from traffic flow information based on fuzzy logic.* Paper presented at the Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE.

[40] Lu, J., & Cao, L. (2003). *A quantization method of traffic congestion evaluation based on fuzzy logic.* Paper presented at the Systems, Man and Cybernetics, 2003. IEEE International Conference on.

[41] Akcelik, R. (2000). Travel time functions for transport planning purposes: Davidson's functions, its time-dependent form and an alternative travel time function. *Akcelik & Associates Pty Ltd.*

[42] Liu, H. X., Ban, X., Ran, B., & Mirchandani, P. (2003). Formulation and Solution Algorithm for Fuzzy Dynamic Traffic Assignment Model. *Transportation Research Record*(1854), 114-123.

[43] Grenoble, U.P.M.-. http://brassens.upmf-grenoble.fr/IMSS/mamass/graphecomp/dijkstra.htm.