Algorithms for optimal guidance of users in road networks

Farida Manseur

UNIVERSITÉ PARIS-EST
ÉCOLE DOCTORALE MSTIC
MATHÉMATIQUES ET SCIENCES ET TECHNOLOGIES DE
L'INFORMATION ET DE LA COMMUNICATION

THÈSE

Présentée pour l'obtention du titre de

Docteur de l'université Paris-Est
Spécialité : Mathématiques appliquées

Par

Farida MANSEUR

Algorithmes de guidage optimal des usagers dans les réseaux de transport

Thèse préparée à l'IFSTTAR,
au sein du laboratoire GRETTIA

Soutenue publiquement le 16/10/2017 devant un jury composé de :

Said MAMMAR	Université d'Evry Val d'Essonne	Rapporteur
Anne BOUILLARD	Nokia Bell-Labs	Rapporteur
Marie-Christine COSTA	ENSTA Paris-Tech	Examinatrice
Frédéric ROUPIN	Université Paris Nord	Examinateur
Nadir FARHI	IFSTTAR (GRETTIA)	Encadrant
Jean-Patrick LEBACQUE	IFSTTAR (GRETTIA)	Co-encadrant
Habib HAJ-SALEM	IFSTTAR (GRETTIA)	Directeur de thèse

A mes parents, qui ont toujours tout fait pour me permettre de réussir

A mon mari et notre fille Sara, qui ont toujours cru en moi et n'ont jamais cessé de m'encourager

A la mémoire de mon frère

Résumé

Nous nous intéressons dans ce travail au guidage optimal des usagers dans un réseau routier. Plus précisément, nous nous focalisons sur les stratégies adaptatives de guidage avec des garanties en termes de fiabilité des temps de parcours, et en termes de robustesse de ces stratégies. Nous nous basons sur une approche stochastique où des distributions de probabilités sont associées aux temps de parcours sur les liens du réseau. Le guidage est adaptatif et individuel. L'objectif de ce travail de recherche est le développement de stratégies « robustes » de guidage des usagers dans un réseau de transport routier. Une stratégie de guidage d'un nœud origine vers un nœud destination est dite robuste, ici, si elle minimise la détérioration de sa valeur maximale calculée au départ de l'origine, contre d'éventuelles reconfigurations du réseau dues à des coupures de liens (accidents, travaux, etc.) La valeur de la stratégie de guidage est maximisée par rapport à la moyenne et à la fiabilité des temps de parcours associées à la stratégie. Deux principales parties sont distinguées dans ce travail. Nous commençons par l'aspect statique du guidage, où la dynamique du trafic n'est pas prise en compte. Nous proposons une extension d'une approche existante de guidage, pour tenir compte de la robustesse des itinéraires calculés. Dans une deuxième étape, nous combinons notre nouvel algorithme avec un modèle microscopique du trafic pour avoir l'effet de la dynamique du trafic sur le calcul d'itinéraires robustes.

Abstract

In this work, we are interested in the optimal guidance of users on road networks. More precisely, we are focused on the adaptive strategies of guidance with guarantees in terms of the travel time reliability and in terms of the robustness of the strategies. We base here on a stochastic approach, where probability distributions are associated to travel times on the links of the network. The guidance is adaptive and user-based. The objective of this work is the development of "robust" strategies for user guidance in a road network. A guidance strategy is said to be robust, here, if it minimizes the deterioration of its maximum value calculated at the origin, against eventual reconfigurations of the network due to link failures (accidents, works, etc.) The value of a guidance strategy is maximized with respect to the mean travel time and its reliability. Two main parts are distinguished in this work. We start with the static aspect of the guidance, where the traffic dynamics are not taken into account. We propose an extension of an existing guidance approach, to take into account the robustness of the calculated itineraries. In a second step, we combine our new guidance algorithm with a microscopic traffic model in order to have the effect of the traffic dynamics on the robust route calculation.

Remerciements

C'est avec un grand soin que je réserve quelques mots pour remercier les personnes qui m'ont aidé à réaliser ce petit bout de recherche. Que ce soit les personnes qui ont eu un rapport direct avec mes travaux, qui m'ont aidé à évoluer et avancer, ou celles qui ont tout simplement été présentes pendant les moments durs et qui m'ont aidé à les surpasser grâce à leurs petits mots d'encouragement. D'avance pardon à ceux que j'aurais malgré tout oublier.

Je tiens tout d'abord à exprimer mes plus profonds remerciements à mes encadrants, M. Nadir Farhi, M. Habib Haj-Salem et M. Jean-Patrick Lebacque pour leur disponibilité, leurs précieux conseils et leur implication durant ces deux ans et demi de thèse.

Je tiens bien évidement à remercie les membres de mon jury de thèse qui ont accepté d'évaluer mon travail. Je remercie M. Saïd Mammar et Anne Bouillard qui ont accepté la lourde tâche de rapporter ce mémoire, mais aussi Mme Marie-Christine Costa et Frédéric Roupin qui nous ont fait l'honneur de faire partie du jury d'évaluation. Je les remercie pour le temps et l'énergie consacrés à l'évaluation de ma contribution.

Un grand merci à mes collègues de travail qui ont rendu mon quotidien agréable durant ces trois dernières années. Pour leurs aides diverses, je tiens à remercier Joêlle Guillot, Annie Thuilot et Marie Josephe Duboc du secrétariat du GRETTIA, Méranh Karounna et Nathalie Galéa du secrétariat général de l'IFSTTAR, Sylvie Cach du secrétariat de l'Ecole Doctorale MSTIC et également Mustapha Tendjaoui (sans qui rien ne fonctionnerait au GRETTIA), Régine Seidowsky directrice adjointe du GRETTIA, Cyril Nguyen Van Phu ingénieur de recherche au GRETTIA. Je remercie l'ensemble des membres du GRETTIA. Merci à tous les doctorants du GRETTIA.

Merci a mes amis qui m'ont supporté pendant cette période stressante. Merci à vous Jennie Lioris, Asma Khelifi, Mouna Amir, Naima Manseur, Rachid Manseur, Lynda Laarbi-Cherif, Taous Chabani, Zina boudjemaa, Moh Tinicha, Chafea Haddad.

Enfin mes derniers mots et mes plus tendres pensées vont à ma famille et à mes proches, à mes chers parents qui m'ont tout donné, à mes adorables frères et sœurs, à mon époux et notre fille Sara. Merci pour votre soutien constant, vos encouragements, et l'affection indéfectible que vous me prodiguez.

Introduction générale

Contexte et Problématique

De nos jours, le besoin de se déplacer est devenu une question primordiale pour l'homme. Ce besoin est bien fondamental pour la vie professionnelle et la vie sociale de l'être humain. Avec l'augmentation de la population, l'amélioration du niveau de vie des citoyens et l'extension des zones urbaines, le nombre de véhicules particuliers ne cesse d'augmenter. Selon l'Association des Constructeurs Européens d'Automobiles (ACEA), les taux de motorisation en 2014 sont de 661 pour 1000 habitants pour l'Amérique du Nord et 569 pour 1000 habitants pour l'Union européenne, contre 44 pour 1000 habitants en Afrique. En France, le taux de motorisation a atteint un pic avec 82,5 % des ménages français disposant d'un véhicule particulier en 2010 et 82,9% en 2015 [source statista.com].

Chaque année, TomTom publie le TomTom Traffic Index qui permet de déterminer le taux d'embouteillage dans les plus grandes métropoles mondiales. En 2016, c'est la ville de Mexico qui est la plus touchée par les embouteillages à l'échelle mondiale, avec un taux de 66%, en hausse de 7% par rapport à 2015. Cela représente 227 heures de plus passées dans sa voiture sur l'année pour le conducteur mexicain. En France, Marseille et Paris arrivent en tête du classement avec un taux de 40% et 38% respectivement. En effet, en 2016, le trafic en région parisienne a augmenté de 2% et les automobilistes franciliens ont perdu en moyenne 40 minutes par jour dans les embouteillages, soit 154 heures dans l'année.

Ce niveau élevé des taux de motorisation a des conséquences négatives considérables en terme économique, social, et surtout environnemental. En effet, cela indique une hausse du niveau de demande de déplacements sur les réseaux routiers (qui peut être due à une hausse du niveau de la demande de déplacement sur tous les modes, comme ça peut être un basculement de la demande de déplacement vers le mode routier). Les deux cas, sont générateurs de congestions récurrentes sur les réseaux routiers qui traduisent un déséquilibre notable entre l'offre de transport et la demande de déplacement. Ces congestions induisent de nombreuses conséquences économiques, sociales, sanitaires et écologiques telles que des émissions inutiles de gaz à effet de serre, une perte de temps importante, et une augmentation de la consommation du carburant. Par conséquent,

l'optimisation de la gestion des déplacements et du transport s'impose comme un sujet primordial à traiter.

Pour combattre la congestion routière, on agit ou bien sur la demande ou sur l'offre de déplacement (en référence à la définition de ce phénomène qui apparait à chaque fois que la demande de déplacement dépasse l'offre). L'action sur l'offre consiste à améliorer la capacité du réseau routier, en augmentant sa capacité (construction de nouvelles routes ou autoroutes, augmentation des nombres de voies etc.) ou en optimisant sa gestion (gestion des intersections, limitations de vitesses, contrôle d'accès, etc.). L'action sur la demande consiste en l'orientation des usagers dans leurs choix de mode de déplacement, dans leurs choix d'itinéraire, ou encore dans leurs choix d'heure de départ.

Nous nous intéressons dans cette thèse à l'action sur la demande de déplacement, et plus particulièrement au guidage des usagers dans leurs déplacements, qui consiste donc à les guider dans leurs choix d'itinéraires. Nous nous basons sur une approche stochastique et adaptative du guidage, qui permet de garantir des itinéraires à temps de parcours fiables, et des stratégies adaptatives de routages aux niveaux des jonctions du réseau. Nous proposons ici une extension à cette approche dans l'objectif d'inclure une sorte de robustesse pour ces stratégies de guidage.

Objectifs et Contribution

Les systèmes de transport sont des systèmes complexes. L'optimisation de leur gestion nécessite une bonne compréhension du fonctionnement de ces systèmes, et un développement de stratégies efficaces de gestion et de régulation du trafic. Dans les réseaux routiers, les usagers ont le choix entre différents moyens de transport pour effectuer leurs trajets, à savoir le véhicule particulier, les transports collectifs (bus, train, etc), la marche à pieds, etc. En termes de confort, les usagers choisissent en général le véhicule particulier même si en termes du coût financier et du temps de trajet, les transports en commun sont plus favorables. Une optimisation du choix d'itinéraire pourrait être bénéfique aussi bien pour l'usager que pour la collectivité.

Nous nous intéressons ici au mode routier, et au choix d'itinéraire. Ce dernier réalise une affectation des usagers sur les itinéraires du réseau. Plusieurs formulations de l'affectation existent : guidage, affectation statique (équilibre usager et optimum social), et affectation

dynamique. Nous nous intéressons dans ce travail au guidage optimal des usagers dans un réseau routier. Plus précisément, aux stratégies adaptatives de guidage avec des garanties en termes de fiabilité des temps de parcours et en termes de robustesse de ces stratégies. On entend par robustesse d'une stratégie de guidage, une stratégie qui proposerait un ou des chemins dont les temps de parcours et leurs fiabilités respectives peuvent « résister » à des événements exceptionnels qui changeraient la reconfiguration du réseau : ruptures de liens suite à des travaux, des accidents, ou autre. L'approche adoptée ici est stochastique, où des distributions de probabilités sont associées aux temps de parcours sur les liens du réseau. Le guidage est adaptatif, en ce sens que la stratégie peut adapter le chemin proposé avant d'atteindre la destination. Cette adaptation se fait en général aux niveaux des nœuds, et tient compte de nouvelles informations sur l'état du trafic dans le réseau, au moment d'arrivée au nœud concerné. Le guidage est individuel, en ce sens que chaque usager optimise son cheminement. Il est dynamique dans le sens où l'optimisation est adaptative et se fait dans le temps. L'optimisation ne prend pas en compte la dynamique du trafic, ni l'élasticité de la demande, comme dans un modèle d'affectation dynamique. Cependant, l'adaptation du guidage en temps réel à l'état du trafic récompense cet aspect. La modélisation stochastique que nous avons choisie ici permet d'avoir non seulement des chemins à temps de parcours minimaux en moyenne, mais elle permet surtout de maximiser la probabilité d'arriver à destination dans un temps donné. Ceci permet une meilleure gestion du temps par les usagers, et garantie la fiabilité des temps de parcours des chemins obtenus. De plus, cette approche étant adaptative, ceci nous donne la possibilité de relancer le calcul au niveau de chaque nœud du réseau et donc d'utiliser l'information temps-réel sur l'état du trafic.

L'objectif de cette thèse est le développement de stratégies « robustes » de guidages des usagers dans un réseau de transport routier. Deux aspects sont distingués dans ce travail. Nous commençons par l'aspect statique du guidage, où la dynamique du trafic n'est pas prise en compte. Nous proposons une extension d'une approche existante de guidage, pour tenir compte de la robustesse des itinéraires calculés. Dans une deuxième étape, nous combinons notre nouvel algorithme avec un modèle microscopique de trafic pour avoir l'effet de la dynamique du trafic sur le calcul d'itinéraires robustes.

Nous citons dans ce qui suit les différentes approches existantes sur lesquelles nous nous baserons dans cette thèse, ainsi que nos contributions principales.

Comme nous l'avons précisé précédemment, l'une des approches efficaces pour minimiser la congestion du trafic dans les réseaux routiers est de fournir aux usagers des informations fiables et opportunes sur l'état du trafic. Cela inclut à la fois des informations prédictives basées sur des indicateurs historiques de la demande, des informations sur les événements tels que des fermetures de routes, des conditions météorologiques, des accidents, etc. La mise à disposition de ces informations permet aux voyageurs de prendre des décisions éclairées et d'accroître l'efficacité du réseau routier. La prolifération des téléphones portables intelligents et des dispositifs de navigation GPS a révolutionné la capacité de fournir de telles informations. Par conséquent, les voyageurs peuvent facilement obtenir des informations routières via les applications fournies par Google, Apple, TomTom, Garmin, Waze, etc., et par l'intermédiaire d'organisations telles que 511.org, autoroute.fr.

L'une des principales caractéristiques de ces applications est la possibilité de planifier des chemins et de guider les véhicules vers leur destination selon le principe du plus court chemin. Les systèmes actuels sont capables de fournir des informations routières en temps réel et ils ont la capacité de guider dynamiquement les usagers en tenant compte de l'état du trafic sur le réseau. Cependant, la sélection d'itinéraire dans de nombreux contextes pratiques peut nécessiter à la fois une route courte et fiable. Les systèmes de navigation actuels n'ont pas la capacité de fournir des informations de fiabilité sur l'itinéraire recommandé.

Lorsque les temps de parcours des liens sont stochastiques, l'approche simple consiste à trouver l'itinéraire le plus court en moyenne appelé le chemin LET (Least Expected Travel-time) proposé par Louis [121]. Le problème LET a été bien étudié et de nombreux algorithmes efficaces existent avec des différentes variantes du problème comme celles proposées par Fu et al. [114], ou encore par Miller-Hooks et al. [63]. Lorsque les temps de parcours sur les liens sont indépendants et lorsque leurs distributions sont invariables dans le temps, le problème LET peut être réduit au problème du plus court chemin déterministe, en associant le poids de chaque lien à son temps de parcours moyen. Ce problème a été largement étudié depuis l'algorithme original de Dijkstra [111]. Abraham et al dans [117] ont montré que les solutions actuelles pour résoudre le problème du plus court chemin déterministe peuvent être exécutées en temps logarithmique avec un prétraitement polynomial. Cependant, Hall dans [61] à montré que l'algorithme de Dijkstra et ses

variantes ne fournissent pas une solution optimale lorsque les poids des liens varient en fonction du temps. Si le réseau satisfait la condition FIFO (First In First Out) étudiée par Astrarita [164], le problème peut être résolu à l'aide d'une programmation dynamique utilisant des poids dépendant du temps avec le graphe original proposé par Dean [144]. Cependant, les solutions du problème du plus court chemin à temps variant sont considérablement plus lentes que leurs équivalentes à temps invariant.

Bien que des algorithmes efficaces existent pour résoudre le problème du plus court chemin déterministe, le problème du plus court chemin stochastique (tenant compte de la fiabilité de la solution) se révèle considérablement plus difficile à résoudre. Selon Fan et al. [123], Nikolava [128], il existe de nombreuses définitions possibles pour le plus court chemin stochastique en fonction du compromis entre le temps de parcours moyen et la variance. Par exemple, on peut minimiser la variance sous réserve d'une contrainte sur le temps de parcours moyen; ou minimiser les temps de parcours moyens sous réserve d'une contrainte sur la variance, ou encore minimiser une somme pondérée du temps de parcours et de la variance.

Une autre définition du plus court chemin stochastique est proposée par Frank [45]. Elle consiste à maximiser la probabilité d'atteindre la destination sous réserve d'un budget de temps. Cette formulation élimine la nécessité de considérer plusieurs objectifs. Elle est connue sous le nom de *problème stochastique d'arrivée à l'heure* (SOTA : Stochastic On Time Arrival). Le problème SOTA peut être résolu en tant que problème de contrôle optimal stochastique étudié par Bertsekas [136]. La stratégie de guidage se traduit dans ce cas comme une politique optimale adaptative, par opposition à un chemin optimal. Une politique optimale génère une règle de décision qui s'adapte aux niveaux des nœuds, et qui définit le chemin optimal d'un nœud donné vers la destination, conditionné sur le temps de déplacement réalisé. Il est clair qu'une telle politique adaptative est meilleure qu'une solution statique apriori. Avec des usagers qui obtiennent des instructions de navigation en temps réel durant leurs trajets, par opposition à une direction imprimée avant le départ, une solution adaptative est également une approche pratique. Pour les situations où un chemin fixe est nécessaire, une route statique peut être obtenue en résolvant le problème SOTA basé sur le chemin associé. Flajolet et al. [125] ont étendu cette formulation SOTA pour résoudre des objectifs plus généraux en modifiant simplement la fonction du coût à la destination.

Nous nous basons ici sur la famille d'algorithmes SOTA (Stochastic on Time Arrival). Une extension de cet algorithme existant est proposée afin d'inclure le critère de robustesse dans le calcul de stratégies de routage. Notre définition de la robustesse est la suivante. Une stratégie de guidage est dite robuste si elle minimise la détérioration des valeurs maximales (en termes de temps de parcours et de sa fiabilité) des chemins qu'elle génère. Nous rappelons que l'approche SOTA permet déjà l'obtention d'un chemin de valeur maximale, c.-à-d. de probabilité maximale d'atteindre la destination avec un budget de temps donné. Notre extension consiste donc à prendre en compte les ruptures de liens, et de favoriser les chemins permettant des alternatives intéressantes en termes de temps de parcours et de sa fiabilité, dans ces cas de rupture.

Organisation de la thèse

Le mémoire de la thèse est présenté en trois chapitres. Le premier chapitre, écrit en langue française, est dédié à l'état de l'art bibliographique. Les chapitres 2 et 3 sont issus d'articles, soumis ou publiés. Ils correspondent donc aux contributions principales de ce travail de recherche. Ces deux parties sont écrites en langue anglaise. Ce mémoire est organisé comme suit.

Chapitre 1: Etat de l'art

L'état de l'art est présenté en trois parties.

Partie 1 : Variables et modèles du trafic routier.

Cette première partie est une introduction à la modélisation et à la simulation du trafic routier. Nous rappelons les modèles existants (statiques et dynamiques) sur les deux principales échelles de modélisations microscopique et macroscopique. Le but est de préciser le cadre de cette étude et de décrire les outils mathématiques existants. L'accent est mis sur les modèles microscopiques car nous nous intéressons dans ce travail par des stratégies de guidage des usagers individuellement.

Partie 2 : Problèmes et algorithmes courant de plus court chemin

Cette deuxième partie porte sur les fondements théoriques de la problématique traitée. Nous commençons par présenter la théorie des graphes utilisée pour la modélisation des réseaux, et les différentes méthodes existantes pour la présentation d'un graphe. Nous nous intéressons également au problème de plus court chemin, sa définition et les algorithmes de recherche d'itinéraire dans les différents graphes, à savoir les graphes statiques, dynamiques, stochastiques, ..., etc.

Partie 3 : Optimisation robuste, et critère de prise de décision dans un environnement incertain.

Cette dernière partie de ce premier chapitre s'articule autour de l'optimisation en définissant l'optimisation robuste, et les différents critères existants qui nous permettent de prendre une décision face à l'incertitude qui fait partie intégrante de notre vie.

Chapitre 2 : Robust guidance

Ce chapitre traite le problème de guidage robuste qui constitue le cœur de notre sujet de recherche. Ce chapitre est organisé en 7 sections comme suit.

La section 1 a pour but d'introduire ce chapitre.

La section 2, fait l'objet d'un bref rappel sur le problème SOTA, que nous divisons en trois sous-sections

- La sous-section 2.1 introduit le modèle de l'approche SOTA existant dans le cas où les distributions des temps de parcours sur les liens du réseau sont indépendantes.
- La sous section 2.2 s'articule autour d'un modèle SOTA qui prend en considération les corrélations des distributions des temps de parcours sur les liens du réseau.
- Dans la sous-section 2.3, nous illustrons cette approche SOTA avec un petit exemple académique.

La section 3 propose un algorithme pour le guidage robuste des usagers.

- La sous section 3.1 s'intéresse au cas où les temps de parcours sur les liens du réseau ne sont pas corrélés. Nous commençons cette sous-section par étendre l'algorithme SOTA proposé par Samaranayake et al. en 2012, en incorporant les choix de routage alternatifs au niveau des nœuds en considérant d'éventuelles défaillances des liens sur les routes menant à la destination. L'algorithme développé diffère de la famille d'algorithmes SOTA. En effet, dans le calcul de la stratégie de guidage, nous calculons une moyenne pondérée des probabilités cumulatives pour arriver à la destination dans un budget de temps donné, au lieu du maximum de ces probabilités cumulatives. Nous fournissons aussi une définition de "l'Optimisation Robuste" dans la sous-section 3.1.1, et nous expliquons comment un usager peut mesurer la robustesse et étudier la qualité de la solution dans la sous-section 3.3.4. Nous donnons également la forme discrète du model robuste et nous analysons sa complexité dans la sous-section 3.1.2.
- La sous-section 3.2 expose le modèle de guidage robuste dans le cas ou les distributions des temps de parcours sont corrélées. Comme dans Samaranayake et al. (2012), nous présentons une extension simple à notre formulation qui considère la corrélation entre un lien et ses voisins en amont (c.-à-d. deux liens successifs). Nous ne nous intéressons pas dans ce travail, aux autres corrélations (c.-à-d. les corrélations entre les liens non successifs), car cela rend le problème plus complexe. Ce point sera une perspective pour nos futurs travaux.

Dans la section 4 nous proposons une optimisation des paramètres de l'algorithme de guidage robuste, et nous montrons comment choisir le paramètre de robustesse en fonction de la fiabilité et du budget de temps souhaités.

La section 5 s'intéresse à une extension de notre algorithme au cas temps-variant, et fait quelques propositions relatives au problème de guidage sous certaines conditions.

La section 6 s'articule autour d'une expérience numérique dans laquelle nous comparons plusieurs scénarios sur un réseau routier bien connu (réseau Sioux Falls) dans le

cas statique, afin de considérer la sensibilité de l'approche proposée aux changements par rapport aux paramètres clés.

La section 7 fait l'objet d'une conclusion pour le chapitre.

Chapitre 3: Combination of the robust routing algorithm with a dynamic traffic model

Dans ce chapitre, le problème de routage est testé dans le contexte de l'affectation du trafic grâce au logiciel de simulation microscopique SUMO (Simulation of Urban MObility). Ce chapitre est organisé comme suit.

La section 1 fait l'objet d'une introduction de ce dernier chapitre.

La section 2 s'articule autour de la procédure à suivre pour combiner notre algorithme de guidage avec le simulateur microscopique SUMO.

La section 3 s'intéresse à introduire le simulateur microscopique SUMO et à présenter les différentes étapes à suivre pour l'initialiser. Le but est de décrire les différents packages que nous utilisons dans cette étude pour générer le réseau et le trafic routier, et définir les différents fichiers et données pour notre fichier de configuration.

La section 4 rappelle le modèle du trafic utilisé dans SUMO, qui est le modèle de Krauss par défaut.

La section 5 s'intéresse aux différentes étapes à suivre pour récupérer les temps de parcours, leurs moyennes et leurs variances.

La section 6 expose la procédure de simulation.

La section 7 s'articule autour d'une expérience numérique, et fournit les différents résultats de simulations obtenus.

La section 8 fait l'objet d'une conclusion pour ce chapitre.

Publications

F. Manseur, N. Farhi, H. Haj-Salem, J-P. Lebacque. An algorithm for robust routing strategies in networks. Journal of Traffic and Transportation Engineering 5 (2017) 8-20. doi : 10.17265/2328-2142/2017.01.002. (presented in the second international conference on intelligent decision science (IDS 2016) in DUBAI).

F. Manseur, N. Farhi, H. Haj-Salem, J-P. Lebacque. Robust adaptive strategies for the guidance of users in road networks. Transportation Research Procedia. 22 (2017) 645-654. doi : 10.1016/j.trpro.2017.03.060. (Presented in the 19th EURO Working Group on Transportation Meeting (EWGT2016) in Istanbul, Turkey).

F. Manseur, N. Farhi, H. Haj-Salem, J-P. Lebacque. Maximizing the Probability of Arriving on Time: Robust Guidance. Poster presented in seventh international conference on industrial engineering and operations management (IEOM2017) in Rabat, Morocco.

S.Berkani, F.Manseur, A. Maidi . Optimal control based on the variational iteration method. Comput Math Appl 64 (2012) 604–610. doi : 10.1016/j.camwa.2011.12.066

En cours de rédaction :

F. Manseur, N. Farhi, C. Nguyen Van Phu, H. Haj-Salem, J-P. Lebacque. An algorithm for robust routing with an application to the optimal user-guidance in road networks. A soumettre dans le journal " European Journal of Operational Research".

Chapitre 1

Etat de l'art

Sommaire

Introduction

Pour traiter les problèmes présentés dans les chapitres suivants, des notions issues de la modélisation des transports, de la théorie des graphes, et de l'optimisation sont utilisées. L'objectif de ce chapitre que nous avons organisé en trois parties, est d'introduire ces différentes notions.

Tout d'abord, il est important d'exposer quelques notions fréquemment utilisées dans la modélisation et la simulation du trafic, et d'attirer l'attention du lecteur sur les différences de dénominations des approches de modélisation. En effet, il apparait que les appellations semblent varier selon l'objectif visé c.à.d. la modélisation peut être effectuée sur la base de gestion de temps (statique, dynamique), ou sur la base des niveaux d'agrégation (microscopique, macroscopique, mesoscopique et sous-microscopique) ou encore sur la base d'autre critères basés sur le modèle et ce qu'il prend en compte (monomodale, multimodale, urbaine et interurbaine). Dans le cadre de ce travail, nous nous intéressons beaucoup plus aux définitions du "microscopique", "monomodale", "statique" et "dynamique".

En suite, nous nous intéressons au problème et algorithmes du plus court chemin qui constituent le vif du sujet de la thèse. Nous donnons un bref historique et quelques notions liés à la théorie des graphes qui est un outil primordial pour les problèmes d'acheminement. Puis nous résumons les différentes méthodes de représentation d'un graphe. Par la suite, nous définissons les différentes classes de graphes. Pour chaque classe de graphes, le problème du plus court chemin est donné et un des algorithmes de sa résolution est présenté.

Enfin, nous considérons l'optimisation robuste et les problèmes d'optimisation dans un environnement incertain. Nous exposons une vaste collection de concepts de robustesse dynamiques, offrant chacun leurs avantages et inconvénient.

Partie 1

Variables et modèles du trafic routier

Sommaire

1. Introduction

Les modèles de transport sont un outil indispensable de compréhension et de planification des déplacements. Il est en effet essentiel de disposer d'instruments permettant de simuler l'évolution des trafics et donc de prévoir la fréquentation des futures infrastructures de transport, de comparer l'intérêt de plusieurs variantes ou bien de tester des scénarios prospectifs à l'échelle d'un territoire. La modélisation est cependant un exercice difficile qui nécessite une constante prise de recul par rapport aux hypothèses que l'on est amené à prendre.

Cette première partie de ce premier chapitre a donc pour objet d'apporter des éléments essentiels liés à la modélisation et à la simulation du trafic. Nous commençons par examiner quelques notions et définitions relative à la demande de déplacement, à l'affectation et à l'écoulement du trafic, avant de nous intéresser aux différentes approches pour la modélisation du trafic et d'exposer le modèle classique à quatre étapes. Nous présenterons ensuite l'approche microscopique en commençant par présenter les différentes variables relatives à cette approche, pour exposer par la suite quelques modèles qui décrivent et évaluent les phénomènes d'un trafic à l'échelle microscopique .Nous terminons par une conclusion à cette première partie.

2. Quelques définitions et notions liées au trafic

Ces éléments classiques et généraux, on peut les trouver aussi d'une manière détaillée dans [1].

- **La demande individuelle de déplacement :** est le déplacement qu'un usager souhaite à réaliser. Cette demande est supposée réalisable au moins par rapport aux possibilités offertes par le réseau de relier son origine i à sa destination j.
- **La demande globale de déplacement :** est la somme des demandes individuelles. On la note généralement sous la forme d'une matrice de demande, appelée matrice Origine-Destination (OD).
- **Le débit :** on le note généralement $D_{ij}(t)$, il définit la demande de l'origine i vers la destination j à l'instant t.
- **L'émission :** notée $E_i(t)$ c'est la demande totale à l'origine i, avec $E_i(t) = \sum_j D_{ij}(t)$.

- **L'attraction :** notée $A_j(t)$ désigne la demande totale vers la destination j, avec $A_j(t) = \sum_i D_{ij}(t)$.
- **L'option :** notée p indique toute possibilité de déplacement entre une origine i et une destination j. Les options peuvent être décrites sous la forme d'itinéraires, de variantes ou autre.
- **Le point de choix :** est un lieu ou les usagers peuvent choisir entre différentes options pour atteindre leur destination. Les points de choix correspondent à des nœuds de réseau (les interactions sur un réseau réel).
- **L'affectation du trafic :** est le processus qui associe une demande de déplacement à une répartition de cette demande entre plusieurs options.
- **Le réseau d'affectation :** est le réseau sur lequel les usagers conçoivent leur déplacement c.à.d. celui sur lequel ils appréhendent les différentes options et effectuent leurs choix d'option.
- **Le coefficient d'affectation :** la proportion de la demande utilisant une option donnée est appelée coefficient d'affectation relatif a cette option. On note donc $\gamma_p^{ij}(t)$ le coefficient d'affectation définissant la proportion de la demande de l'origine i vers la destination j qui empreinte l'option p à l'instant t. D'après la conservation des véhicules appliquée en un point de choix , ces coefficients vérifient la relation $\sum_{p \in E_n^{ij}} \gamma_p^{ij}(t) = 1, \forall\ i, j, n, t.$
- **Le Modèle :** un modèle est une représentation simplifiée d'une partie du monde réel qui se concentre sur certains éléments considérés comme importants d'un point de vue particulier. Les modèles sont, par conséquent, spécifiques au problème et au point de vue.
- **Le guidage :** se réfère à la proposition d'un ensemble de chemins ou de stratégies de cheminement entre un nœud origine et un nœud destination d'un réseau, en optimisant une fonction objective soumise à la demande de trafic et aux contraintes de capacité.

3. Modélisation du trafic

La modélisation est la conception d'un modèle qui sert à prédire le comportement d'un système. Elle peut être classée en différentes catégories selon son objectif et les moyens utilisés.

3.1. La modélisation sur la base de gestion du temps

Selon l'objectif de cette catégorie qui est la gestion de temps, on peut distinguer deux grandes classes à savoir la modélisation statique et la modélisation dynamique.

3.1.1. La modélisation statique : la modélisation statique consiste à représenter de manière simplifiée les déplacements sur un secteur défini et une période donnée. Ce modèle ne prend pas en compte les fluctuations et interactions de la demande de déplacements par rapport au temps et à la distance sur une période donnée, car il s'appuie sur des hypothèses de fonctionnement des déplacements. On parle de modèle statique car les flux sont modélisés de manière stable tout au long d'une période de temps (heures de pointes ou journée). La modélisation statique de représentation des trafics est utilisée pour des réseaux maillés complexes (échelles de territoire variables) dans ce cas la modélisation est macroscopique, ou encore pour des études de carrefours pour lesquels il n'y a pas de choix d'itinéraires possibles dans ce cas la modélisation est microscopique et l'affectation est statique. Un modèle statique peut être monomodal ou multimodal. De même, la méthodologie de modélisation peut être différente selon les données d'entrée et de sortie utilisées pour reproduire le modèle. En effet, on peut distinguer les modèles déterministes et les modèles stochastiques. Les modèles de l'affectation déterministe introduite dans [2], étudient non pas directement les interactions entre les usagers mais l'état du réseau résultant de l'affectation de la demande sur le réseau. Dans cet état, qualifié d'équilibre usager du réseau, les usagers minimisent effectivement et individuellement le coût de leur déplacement et seules les options les plus performantes sont empruntées. Les modèles stochastiques d'affectation proposent de décrire les choix individuels comme les résultats d'une perception stochastique des coûts des options. Les interactions entre les usagers sont alors intégrées a posteriori à travers l'utilisation de l'équilibre stochastique qui régit l'état final résultant de tous les choix individuels.

Les modèles statiques ont pour objet de répartir la demande de déplacement sur le réseau en fonction du plus court chemin, au moyen d'un algorithme faisant intervenir la notion de coût de déplacement minimum. Le coût de trajet étant paramétré par le temps de parcours, le trajet parcouru et le budget en cas de péage. Le choix du modèle

est fonction de la problématique. Les modèles statiques servent à tester des projets ou des politiques de transport sur une zone d'étude donnée.

3.1.2. La modélisation dynamique : La précédente modélisation fait abstraction de tout aspect dynamique et temporel. Or le trafic routier est par nature un phénomène dynamique du fait de la variabilité de la demande et de sa propagation sur le réseau routier. Donc par opposition à la modélisation statique, la modélisation dynamique s'appuie sur une demande de transport qui évolue dans le temps et qui tient compte des interactions entre les résultats des différents pas de temps et sections du réseau. Un modèle dynamique va en effet générer chaque véhicule de la matrice correspondante (avec une origine et une destination) et lui faire parcourir son trajet toujours selon le plus court chemin. Cependant, il tient aussi compte des autres véhicules déjà introduits dans le modèle et son chemin initial peut varier en fonction de l'état du trafic en temps réel. Il peut aussi être bloqué par un encombrement et ne pas avoir le temps d'arriver à destination. La matrice de résultat est donc différente de la matrice de demande de déplacement et les écarts les plus importants sont significatifs d'un dysfonctionnement du réseau. Un autre aspect dynamique du problème est lié au choix de temps de départ. Les usagers peuvent, en effet, choisir l'heure de départ en vue de rencontrer de meilleurs conditions de circulation qui leurs garantissent un instant d'arrivée plus proche de leurs instant d'arrivée souhaité. [3] pose les bases du modèle de choix de l'heure de départ, dans le cas élémentaire d'un couple OD relié par un seul itinéraire, en introduisant un coût généralisé qui intègre le coût des éventuels retard ou avance. Par la suite, de nombreux chercheurs ont enrichi cette formulation initiale au niveau de la modélisation du comportement des usagers d'une part (c.à.d. par rapport à la contrainte de l'heure d'arrivée par exemple en introduisant différentes classes de conducteurs non homogènes [4] ou des tolérances vis-à-vis du retard [6]), et de la complexité de la situation de trafic considérée d'autre part (avec l'utilisation de modèles de trafic plus sophistiqués sur des réseaux à plusieurs couples OD, par exemple [7]ou [5]). Désormais, La plupart des modèles d'affectation dynamique intègre une composante de choix de l'heure de départ, et ce généralement sous la forme de modules annexes.

3.2. La modélisation sur la base des niveaux d'agrégation

Dans cette catégorie, la modélisation est effectuée sur la base de comportement des véhicules, c'est à dire soit on prend chaque véhicule individuellement, soit on considère l'ensemble des véhicules. Ces modèles sont généralement classés en deux principales familles : modèles microscopiques et modèles macroscopiques, que nous décrivons ci-dessous.

3.2.1. La modélisation microscopique : elle permet de représenter l'évolution individuelle des véhicules sur le réseau. Ce type de modélisation est utilisée lorsque les problèmes à résoudre nécessitent une description détaillée de la route et des comportements des conducteurs. Beaucoup de paramètres sont généralement pris en compte, et le comportement d'un véhicule est perçu de façon individuelle par ses voisins. Dans cette approche, le trafic est vu comme un système de véhicules en interaction dont chacun est géré par un modèle. Deux types de modèles sont proposés dans la littérature, le modèle longitudinal et le modèle latéral. Les modèles microscopiques longitudinaux sont utilisés pour décrire le mouvement du véhicule seul ou bien pour représenter son comportement de poursuite d'un autre véhicule. Les modèles microscopiques latéraux sont utilisés pour contrôler le changement de voies des véhicules dans un trafic à deux ou plusieurs voies. Ces modèles latéraux comportent en général deux étapes; l'étape de prise de décision et l'étape de changement de voie. Deux approches ont été proposées dans la littérature pour l'étape de changement de voie à savoir l'approche discrète c.à.d. le conducteur fait des sauts entre les voies [19], ou encore l'approche continue. Dans ce dernier cas le véhicule suit une trajectoire bien déterminée pour changer la voie.

3.2.2. La modélisation macroscopique : dans la modélisation macroscopique, le trafic est considéré comme un groupe de véhicules. Les paramètres mis en jeu dans cette approche sont principalement la concentration k (densité), la vitesse moyenne U et le débit Q . Ces paramètres caractérisent l'état d'un trafic donné ainsi que son diagramme fondamental.

Définition d'un diagramme fondamental : Le diagramme fondamental du trafic donne une relation entre le débit et la densité. Il peut être utilisé pour prédire le comportement d'un tronçon routier. Le diagramme fondamental peut s'écrire sur trois plans différents. La figure ci-dessous montre la forme générale du diagramme fondamental dans les trois plans.

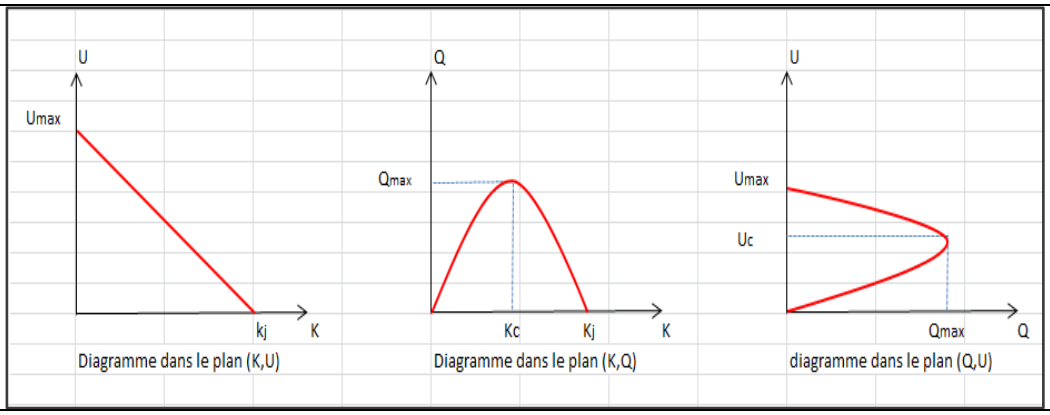


Figure1 : diagramme fondamental dans différents plans

Prenant le diagramme dans le plan (k, Q). La plupart des travaux dans la littérature définissent deux zones du trafic; la zone libre et la zone congestionnée. Le trafic libre est qualifié par sa faible densité (zone où $k < k_c$). Dans cette zone, une augmentation de la densité k est accompagnée par une augmentation du débit Q. Le seuil k_c est appelé densité critique. Un trafic congestionné est caractérisé par une forte densité (zone où $k_c < k$). Dans cette zone une augmentation de la densité k est accompagnée par une diminution du débit Q. Cependant, [18] a identifié une troisième zone appelée zone critique qui est caractérisée par une densité moyenne k_c et un débit maximal Q_{max}, il la qualifié cette zone par un trafic synchronisé. Plusieurs travaux ont été réalisés dans le cadre de cette approche macroscopique. Pour plus de détails le lecteur peut se référer à [20-21], ou encore [23]. Parmi les modèles du trafic macroscopiques les plus reconnus aujourd'hui, on cite les trois modèles suivants : le modèle d'équilibre LWR [11-12], le modèle de non-équilibre ARZ [13] et le modèle multi-classes d'origine-destination MOD [10] et [14]. Dans ces modèles, l'évolution temporelle des grandeurs macroscopiques comme densité, vitesse et flux est représentée par des systèmes d'équations aux dérivées partielles non-linéaires.

Il existe deux autres classes de modélisation, moins courantes que les classes des modèles microscopiques et macroscopiques.

3.2.3. La modélisation mesoscopique : le point de vue mésoscopique peut être qualifié d'intermédiaire entre les descriptions microscopique et macroscopique en termes de quantité d'informations pour la modélisation des systèmes. Ce type de modélisation permet de modéliser les véhicules par paquets (pelotons) pouvant s'échanger des véhicules. [25] a proposé deux stratégies pour cette approche. La première stratégie permet de prendre en considération l'hétérogénéité des différents conducteurs en combinant le comportement microscopique d'un véhicule avec les règles macroscopiques comportementales. La deuxième stratégie est basée sur la spécification des comportements d'un groupe hétérogène de véhicules. Plusieurs travaux ont été menés dans le cadre de cette approche à savoir ceux développés par [22], [24] ou encore [31].

3.2.4. La modélisation sous-microscopique : Ce type de modélisation est à un niveau en dessous de la modélisation microscopique. On considère que chaque véhicule peut se diviser en plusieurs sous-structures (par exemple modélisation de la vitesse de rotation du moteur par rapport à la vitesse du véhicule).

Ces quatre approches de la modélisation du trafic sont illustrées par la figure ci dessous

			
a-Approche macroscopique	b-Approche mesoscopique	c-Approche microscopique	d-Approche sous-microscopiques

Plusieurs simulateurs de trafic ont été développés pour étudier les modèles issus de ces approches. On cite à titre d'exemple, AIMSUN [9] pour la modélisation microscopique,

METACOR [16] pour la simulation macroscopique, et SUMO [147] pour la simulation mesoscopique et microscopique.

Dans la suite de cette thèse, les algorithmes que nous définissions ont tendance à privilégier les modèles microscopiques. Pour cette raison, nous allons aborder d'une manière un peu plus détaillée, dans la section 5 de ce chapitre, ces modèles microscopiques. Le principal simulateur utilisé dans cette thèse est SUMO qui possède une vision microscopique du trafic routier, afin de modéliser les comportements individuels des automobilistes et de se rapprocher d'un cas réel.

3.3. Autres classifications

Si on s'intéresse de plus près au modèle et de ce qu'il prend en compte, on peut avoir d'autres approches de modélisation à savoir:

3.3.1. Modélisation monomodale : On s'intéresse seulement à un seul mode de transport pendant tout le trajet. Même s'il s'agit d'un seul type spécifique de véhicule, le transfert peut avoir lieu avec ce moyen de transport, par exemple bus-bus, métro-métro,.....etc.

3.3.2. Modélisation multimodale : On considère plusieurs modes de transport et aussi les interactions entre eux. Par exemple métro et bus, vélo, et marche à pied,....etc.

3.3.3. Modélisation urbaine ou interurbaine : Dans la modélisation interurbaine, les flux Origines Destinations sont reconstitués en estimant directement les flux routiers sur la zone d'étude, avant de les affecter sur le réseau modélisé. Cette méthode est utilisée pour les modèles d'affectation des Transports en Commun sur les réseaux urbains. Dans ces cas, la demande de transport est une donnée d'entrée.

Le modèle classique le plus connu dans la littérature est le modèle à quatre étapes "MODUS" qui est un modèle statique, multimodal et macroscopique qu'on résume dans la sous section suivante.

3.4. Modèle à quatre étapes

Le « modèle à quatre étapes » ne constitue pas un modèle au sens propre, mais une approche générale de modélisation. Il s'agit de modéliser (prévoir) les déplacements à l'intérieur d'une agglomération urbaine donnée, découpée en plusieurs zones pour les besoins de la modélisation. La formulation classique qui procède en quartes étapes a été proposé initialement par Wilson 1970. Ces quartes étapes répondent généralement aux questions suivantes:

1)-Quelles sont mes pratiques de mobilité ? (Étape de génération)
2)-Comment est déterminée ma destination ? (Étape de distribution)
3)-Quels sont les modes de transport à ma disposition ? (Étape de choix modal)
4)-Quel chemin j'emprunte pour réaliser ce déplacement ? (Étape d'affectation)

Les deux premières étapes visent à déterminer la demande de déplacement, les deux dernières correspondent à la répartition du trafic en fonction de l'offre modale et routière. Dans la pratique, on ne suit pas forcément l'enchainement de ces quartes étapes, même si, implicitement, elles figurent toujours d'une manière ou d'une autre dans la modélisation, en étant parfois conjointes (génération-distribution à l'aide d'un modèle gravitaire, choix modal-distribution à l'aide d'un modèle Logit).

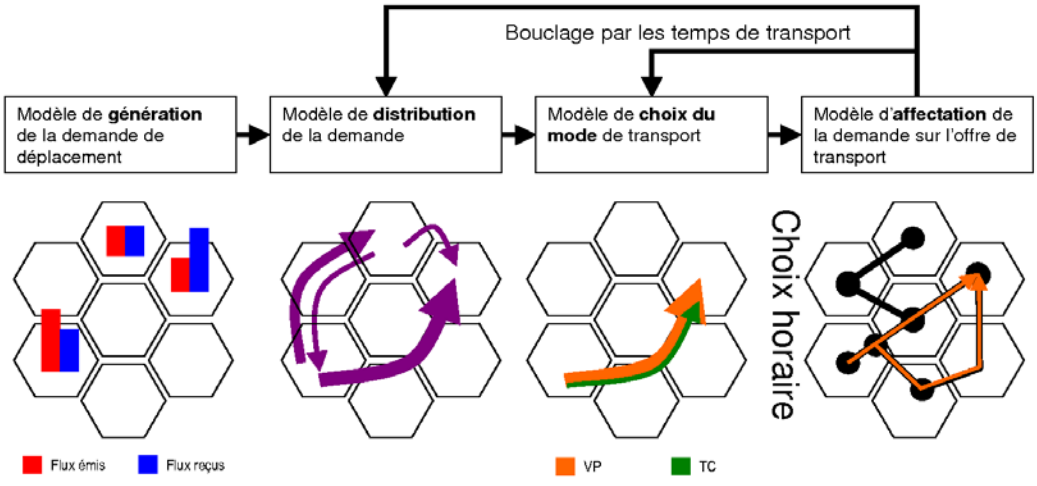


Figure2 : Schématisation des séquences de modélisation pour le modèle classique

4. Modèles microscopiques de l'écoulement du trafic

Les techniques de modélisation du trafic routier permettent aux gestionnaires des réseaux de transport de mieux exploiter leurs infrastructures et représentent ainsi des outils d'aide à la décision. En effet, les modèles permettent la prédiction de l'état du trafic. En prévenant les congestions et en détectant les incidents et accidents, ils offrent la possibilité de traiter et intervenir dans des délais de temps réduits.

Comme nous l'avons précisé dans la section précédente il existe plusieurs types de modèles à différentes échelles qu'il convient de choisir en fonction du phénomène physique que l'on cherche à comprendre. Selon qu'on s'intéresse à l'écoulement global du trafic sur un réseau routier ou à des interactions locales entre quelques véhicules lors d'un changement de direction ou à l'approche d'une intersection, la question de spécification du niveau de détail est primordiale. Pour une raison liée à l'objectif des travaux de notre thèse, nous n'allons aborder dans cette section que les modèles microscopiques qui reflètent le comportement individuel des conducteurs interagissant avec les véhicules environnants. Avant d'exposer les différents modèles de cette famille, nous commençons d'abord par introduire les variables de base au niveau microscopique (le niveau du véhicule).

4.1. Variables de base des modèles microscopiques

On considère un véhicule n à l'instant t , à la position $x_n(t)$, de vitesse instantanée $v_n(t) = \dot{x}_n(t)$. L'accélération est notée $a_n(t) = \ddot{x}_n(t)$. Le véhicule leader $n+1$ précède le véhicule suiveur n.

Les variables microscopiques sont alors :

- la distance inter-véhiculaire $s_n(t) = x_{n+1}(t) - x_n(t)$, définie comme la distance séparant deux points identiques (par exemple, les deux pare-chocs avant) des deux véhicules, comme le montre la figure 3 ci dessous
- l'inter-distance $d_n(t)$ représente l'espace disponible entre deux véhicules (la distance entre la pare-chocs avant du véhicule suiveur, et le pare-chocs arrière du véhicule leader). L'inter-distance est liée à la distance inter-véhiculaire par la relation suivante.

$$s_n(t) = d_n(t) + l_{n+1}$$

où l_{n+1} est la longueur du véhicule $n+1$;

- le temps inter-véhiculaire $h_n(t)$ est le rapport de la distance inter-véhiculaire par la vitesse du véhicule suiveur n ;
- la vitesse relative du véhicule n par rapport à son véhicule leader $n+1$ est $\dot{s}_n(t) = \dot{x}_{n+1}(t) - \dot{x}_n(t)$;
- le temps à collision $t_c(t)$ défini comme le rapport de l'inter-distance à la vitesse relative ;

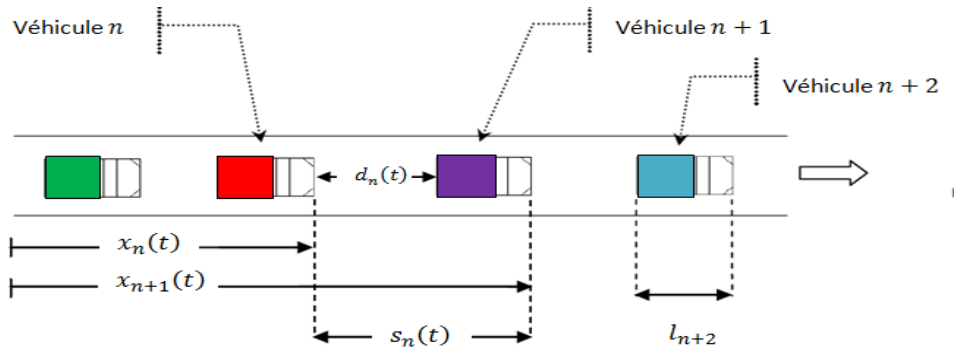


Figure 3 : variables pour les modèles microscopiques

4.2. Les principaux modèles microscopiques

Un modèle microscopique fournit une description des mouvements individuels de véhicules. Il s'agit de mouvements considérés comme étant attribuables aux caractéristiques des conducteurs et des véhicules, aux interactions entre les éléments conducteur-véhicule, aux caractéristiques routières, aux conditions extérieurs, et aux règles de circulation et de contrôle. La plupart des modèles de simulation microscopiques supposent que le conducteur répondra uniquement à l'un des véhicules qui roule dans la même voie devant lui (c.à.d. le leader). Lorsque le nombre d'unités de conduite-véhicule sur la route est très faible, le conducteur peut choisir librement sa vitesse compte tenu de ses préférences et capacités, des conditions de la chaussée, de la courbure, des limites de vitesse en vigueur, etc. En tout état de cause, le conducteur aura peu de raison d'adapter sa vitesse aux autres usagers de la route. La vitesse cible du pilote est la soi-disant vitesse libre. Dans la vie réelle,

la vitesse libre varie d'un pilote à l'autre, mais la vitesse libre d'un seul conducteur change aussi dans le temps. La plupart des modèles microscopiques supposent, cependant, que les vitesses libres ont une valeur constante qui est spécifiée par le conducteur. Lorsque les conditions de circulation se dégradent, les conducteurs ne pourront plus choisir leurs vitesses librement, car ils ne pourront pas toujours dépasser ou passer un véhicule plus lent. Le conducteur devra adapter sa vitesse aux conditions de circulation en vigueur, c'est-à-dire que le conducteur suit.

Dans cette section nous présentons un état de l'art des modèles microscopiques qu'on divise en deux grandes classes à savoir les modèles latéraux et les modèles longitudinaux. Nous nous intéressons beaucoup plus aux modèles latéraux, particulièrement le modèle de KRAUSS[17] qui a été choisi par défaut dans le simulateur SUMO[147].

4.3. Les modèles longitudinaux : Ces modèles visent à gérer la poursuite d'un véhicule sur un autre. Ils décrivent, analysent et évaluent plusieurs caractéristiques du trafic comme la congestion. On peut distinguer deux catégories de modèles longitudinaux en se basant sur le style de conduite, à savoir la conduite "manuelle" et la conduite "automatisée". Parmi les modèles longitudinaux les plus connus on cite:

- Le modèle de "conducteurs" : c'est un modèle qui tient compte de plusieurs paramètres liés d'une part au comportement du conducteur avec le véhicule et d'autre part à sa réaction par rapport à l'environnement. Parmi les premiers modèles "conducteurs", on cite le modèle GRH (Gazis, Herman, Rothery, 1958). L'idée de ce modèle est basée sur l'application d'un retard sur la commande du véhicule suivi dans un contexte de conduite en fille. Ce modèle à été étudié et amélioré dans plusieurs travaux, par exemple [30]. En 2000, Treiber a développé un autre modèle appelé IDM (Intelligent Driver Model), c'est un modèle qui privilégie une définition de l'accélération mais n'integre pas de temps de retard.

- Le modèle de véhicules automatisés : ce type de modélisation « automatisée » est relativement récent. On trouve plusieurs modèles qui ont été publiés par les constructeurs automobiles à savoir le modèle décrit dans [29], qui est basé sur la modélisation de la dynamique du véhicule. Ce modèle est actuellement

utilisé par le constructeur américain Ford. On trouve aussi, le modèle de développé dans [28] qui est un modèle basé sur les réseaux de neurones et la logique floue. Plus récemment, un autre modèle basé sur la théorie de l'élasticité et de la mécanique des contacts à été développé par [27].

4.4. Les modèles latéraux : Ces modèles représentent le comportement latéral du véhicule en situation de changement de voie. La modélisation du trafic à plusieurs voies a commencé par les travaux de Gipps [33]. La plupart de ces modèles sont basés sur le principe que les conducteurs évaluent la voie occupée et la voie cible, puis ils choisissent leurs directions de voie (changer ou ne pas changer de voie) en comparant ces deux voies (occupée et ciblée) sur la base de critères bien définis. Parmi les modèles latéraux les plus connus on cite :

- Modèle de Gipps [33] : ce modèle est basé sur la condition de sécurité qui peut être considérée par les distances de freinage de voitures individuelles. Ainsi, le comportement du conducteur est déterminé par deux critères principaux, à savoir le maintien de la vitesse désirée et le bon choix de la voie pour une manœuvre de virage. Si le virage est loin, le conducteur se base seulement sur sa vitesse désirée (atteinte ou maintenue). Lorsque le conducteur ne dispose plus d'une voie acceptable pour changer, alors le conflit est résolu d'une manière déterministe à l'aide d'un système de priorité en tenant en compte l'emplacement des obstacles, la présence de véhicules lourds et le gain de vitesse. Krauss [17] a présenté un modèle basé sur le modèle de Gipps qui a été intégré dans le simulateur de trafic SUMO, et que nous allons utilisé dans la suite de cette thèse. Pour cette raison, une présentation détaillée de ce modèle de Krauss est donnée dans le chapitre 3 de cette thèse.
- Modèle de Hidas [26] : dans ce modèle, le critère de faisabilité est indispensable, c.à.d. on doit disposer d'un minimum d'inter-distance entre le véhicule leader et le véhicule suiveur. C'est un modèle qui considère l'état de trafic sous ses trois forme à savoir trafic libre, trafic congestionné avec un véhicule leader coopératif (un véhicule leader coopératif est celui qui adapte et évalue sa vitesse et la distance de sécurité pour laisser au véhicule suiveur la possibilité de changer de voie), et un trafic congestionné avec un véhicule leader non coopératif.

- Modèle de Kesting [19]: récemment, un modèle de changement de voie dans un trafic à deux ou plusieurs voies à été proposé dans [19]. Contrairement au modèle de Hidas, ce modèle considère trois véhicules pour la prise de décision de changement de voie, c.à.d. il prend en compte le véhicule courant, le véhicule suiveur dans la voie occupée (actuelle), et encore le véhicule suiveur dans la voie ciblée. D'après Kesting, pour changer la voie, on doit satisfaire deux conditions; la sécurité (fixation d'un seuil de décélération pour le prochain véhicule suiveur dans la voie ciblée), et la motivation (caractéristiques des conducteurs, et caractéristiques de l'autoroute).

5. Conclusion

Dans cette première partie, nous avons fait l'état de l'art sur la modélisation de trafic. Nous avons présenté dans un premier temps les notions essentielles à la modélisation du trafic. Nous avons résumé dans une seconde phase les différentes approches existantes, de façon à éclairer d'une part le point de vue temporel et d'autre part le point de vue comportemental, pour effectuer une modélisation. La dernière phase qui concerne spécifiquement la description de l'approche microscopique présente les modèles significatifs correspondant à la conduite longitudinale. Ces modèles permettent d'une part, d'analyser et d'évaluer plusieurs caractéristiques du trafic comme la congestion, et d'autre part, de gérer la poursuite d'un véhicule par un autre. On a également cité les modèles de la conduite latérale microscopiques, qui sont destinés à gérer le changement de voie dans un trafic à deux ou plusieurs voies. Nous reportons à la deuxième partie la description du problème du plus court chemin et des algorithmes pour sa résolution.

Partie 2

Problèmes et algorithmes courants de plus court chemin

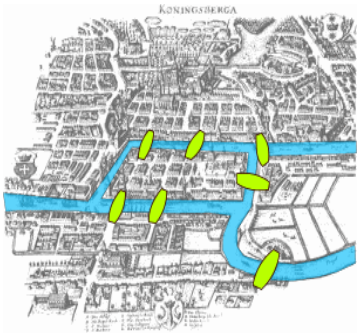
Sommaire

1. Introduction

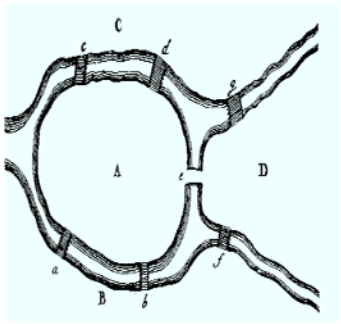
Sur la base des éléments fournis dans la partie 1 de ce premier chapitre, nous abordons dans cette deuxième partie le problème du plus court chemin, et exposons les outils essentiels à sa résolution. Nous commençons par donner quelques notions liées à la définition de graphe, puis exposer le problème du plus court chemin tel que défini dans la littérature. Nous terminons par quelques algorithmes essentiels pour trouver le plus court chemin dans un réseau de transport.

2. Un bref historique sur la théorie des graphes

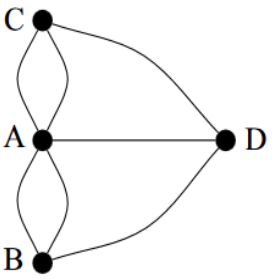
On accorde à Euler l'origine de la théorie des graphes parce qu'il fut le premier à proposer un traitement mathématique de la question, suivi par Vandermonde. La théorie des graphes est la discipline mathématique et informatique qui étudie les graphes. Elle constitue un outil primordial pour résoudre les problèmes de cheminement. Un article du mathématicien suisse Leonhard Euler, présenté à l'Académie de Saint-Pétersbourg en 1735 puis publié en 1741, traitait du problème des sept ponts de Königsberg, ainsi que schématisé par la figure 4 .



a. Image réel des sept ponts de Königsberg



b. Dessin simplifié des sept ponts de Königsberg



c. Graphe associé au problème sept ponts de Königsberg

Figure 4. Problème des sept ponts de Königsberg

Le problème posé était le suivant. Deux îles A et D sur la rivière Pregel à Konigsberg (alors capitale de la Prusse de l'Est, aujourd'hui rebaptisée Kaliningrad) étaient reliées entre elles ainsi qu'aux rivages B et C à l'aide de sept ponts (désignés par des lettres minuscules), comme le montre la figure b. Le problème consistait à trouver une promenade à partir d'un point donné (A, B, C ou D) qui fasse revenir à ce point en passant une fois et une seule par chacun des sept ponts de la ville de Königsberg. Euler représenta cette situation à l'aide d'un "dessin" où les sommets représentent les terres et les arrêtes représentent les ponts, comme le montre la figure c. Grace au développement de la théorie des graphes et avec l'introduction d'une nouvelle structure de données appelée plus tard *graphes,* il démontre l'impossibilité d'emprunter tous ces sept ponts une et une seule fois. Cette structure de donnée très simple a donné naissance à de multiples applications dans tous les domaines liés à la notion de réseau de transport, de télécommunication ou encore aux réseaux sociaux.

3. Notions et définitions générales

Un graphe représenté mathématiquement par $G = (N, A)$ constitue d'un ensemble de nœuds (ou sommets) noté N et d'un ensemble d'arcs (ou arrêtes) noté A. La liaison entre un nœud $i \in N$ et un nœud $j \in N$ est appelée arc (arrête) noté $(i, j) \in A$.

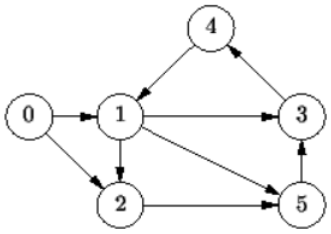
- Un graphe orienté est un graphe dont les arêtes sont orientées, à savoir qu'il est possible de distinguer l'extrémité initiale d'une arête de son extrémité finale. Dans ce cas, on parlera d'arc plutôt que d'arête. Les graphes orientés son souvent nécessaires pour modéliser les routes
- Un graphe est dit non orienté si les arêtes sont non-orientées.
- Chaque arc (ou arrête) dans un graphe peut être associé à une valeur appelée poids notée $c(i,j)$. Un poids peut représenter un coût, un temps de parcours *en encor*e une longueur d'un voyage d'un nœud à un autre nœud voisin.
- Un graphe G possédant des poids associés à chacun de ses arcs, est appelé évalué ou pondéré. La longueur d'un chemin dans ce type de graphes est calculée par la somme de tous les poids des arcs traversés.
- Un graphe G est sans boucles si A ne contient pas d'arcs de la forme (i,i), c.à.d. joignant un nœud à lui-même. Le nombre de nœuds est appelé ordre du graphe.
- Un graphe simple ou encore un $1-\text{graphe}$ est celui qui ne possède pas de boucles ni d'arcs parallèles (deux arcs distincts joignant la même paire de nœuds). En revanche un $p-\text{graphe}$ ou graphe généralisé est un graphe dans lequel il n'existe pas de p arcs de la forme (i,i).
- Un graphe G est réflexif s'il possède une boucle sur chaque sommet (nœud).
- Un graphe est symétrique si, pour tout arc $a_1 = (i,j) \in A$, l'arc $a_2 = (j,i) \in A$. En revanche, un graphe est antisymétrique si, pour tout arc $a_1 = (i,j) \in A$, l'arc $a_2 = (j,i) \notin A$
- Un graphe G est transitif si quels que soient deux arcs adjacents $a_1 = (i,j) \in A$ et $a_2 = (j,k) \in A$, alors l'arc $a_3 = (i,k) \in A$.
- Un graphe G est dit complet si, pour toute paire de sommets (i,j), il existe au moins un arc de la forme (i,j) ou (j,i). Pour plus des détails veuillez se référer par exemple à [35], [36] ou encore [38].

Un chemin dans un graphe peut être défini comme une suite ordonnée de nœuds tels que chaque deux nœuds successifs soient reliés par un arc.

4. Méthodes de représentation d'un graphe

Pour représenter un graphe, il existe plusieurs modes. On peut les classer en trois principales catégories selon la nature des traitements que l'on souhaite appliquer au graphe considéré.

4.1. Liste des successeurs : on peut représenter un graphe à l'aide d'un dictionnaire c.à.d. sous forme d'une table qui contient des entrées. Chaque ligne dans la table correspond à un sommet et comporte la liste des successeurs ou des prédécesseurs de ce sommet. Prenant à titre illustratif, le graphe de la figure 5 ci-dessous.



0	1, 2
1	3, 5, 2
2	5
3	4
4	1
5	3

0	-
1	0, 4
2	0, 1
3	1, 5
4	3
5	1, 2

Graphe illustratif | Table des successeurs | Tables des prédécesseurs

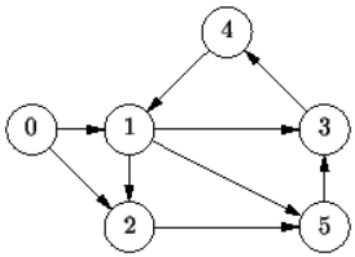
Figure 5. Exemple d'une présentation d'un graphe par une liste des successeurs

Les outils classiques d'algèbre linéaire peuvent également être utilisés pour coder les graphes. La première façon consiste à utiliser la matrice d'adjacence. Une seconde façon est d'utiliser la matrice d'incidence.

4.2. Matrice d'adjacence : avec ce mode, on considère chaque arc comme un lien entre deux sommets. Considérant un graphe $G = (N, A)$ comportant n sommets $(1,2, \dots n)$. La matrice d'adjacence de G est la matrice booléenne $M_{adj} = \left(m_{adj}(i,j)\right)_{1 \le i,j \le n}$ telle que :

$$m_{adj}(i,j) = \begin{cases} 1 \text{ si } (i,j) \in A \text{ c'est à dire que } (i,j) \text{ est un arc} \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{si non} \end{cases}$$

Un graphe orienté quelconque a une matrice d'adjacence quelconque, alors qu'un graphe non orienté possède une matrice d'adjacence symétrique. L'absence de boucle se traduit par une diagonale nulle. La matrice d'adjacence du graphe de la figure précédente peut s'écrit comme suit (Figure 6).

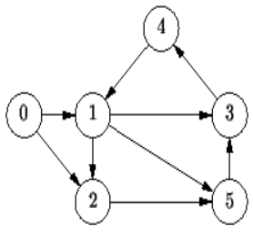


	0	1	2	3	4	5
0	0	1	1	0	0	0
1	0	0	1	1	0	1
2	0	0	0	0	0	1
3	0	0	0	0	1	0
4	0	1	0	0	0	0
5	0	0	0	1	0	0

Graphe illustratif | Matrice d'adjacence

Figure 6. Exemple d'une présentation d'un graphe par une matrice d'adjacence

4.3. Matrice d'incidence : dans ce mode de représentation, la relation entre arcs et sommets est exploitée. Considérant un graphe $G = (N, A)$ comportant n sommets $(1,2, \ldots . n)$ et m arcs $(a_1, a_2, \ldots . a_m)$, alors la matrice d'incidence de G est la matrice $M_{ind} = (m_{inc\,(i,j)})$ de dimension $n \times m$ telle que :



	(0,1)	(0,2)	(1,3)	(1,5)	(1,2)	(2,5)	(3,4)	(4,1)	(5,3)
0	1	1	0	0	0	0	0	0	0
1	-1	0	1	1	1	0	0	-1	0
2	0	-1	0	0	-1	1	0	0	0
3	0	0	-1	0	0	0	1	0	-1
4	0	0	0	0	0	0	-1	1	0
5	0	0	0	-1	0	-1	0	0	1

Graphe illustratif | Matrice d'incidence

Figure 7. Exemple d'une présentation d'un graphe par une matrice d'incidence

$$m_{inc}(i,k) = \begin{cases} 1 \text{ si i est l}'\text{extrémité initiale de } a_k \\ -1 \text{ si i est l}'\text{extrémité terminale de } a_k \\ 0 \text{ si si i n}'\text{estpas une extrémité de } a_k \end{cases}$$

La matrice d'adjacence du graphe de la Figure 6 peut s'écrit comme sur la Figure 7.

5. Le problème du plus court chemin

Le problème du plus court chemin constitue la base de tous les problèmes de planification d'itinéraire. C'est un problème qui se rencontre dans de nombreuses applications telles que :

- les problèmes de tournées,
- certains problèmes d'investissement et de gestion de stocks,
- les problèmes de programmation dynamique à états discrets et temps discret,
- les problèmes d'optimisation dans les réseaux (routiers, télécommunications),
- certaines méthodes de traitement numérique du signal, de codage et de décodage de l'information.

Le problème du plus court chemin dans un graphe se formule comme suit. Etant donné un graphe orienté $G = (N, A)$, on associé à chaque arc $a = (i, j) \in A$ un nombre $l(a) \in \mathbb{R}$ appelé longueur de l'arc. On dit que le graphe G est valué par les longueurs $l(a)$. Le problème de plus court chemin entre deux nœuds i et j consiste alors à trouver un chemin $u(i, j)$ allant de i à j et dont la longueur totale $l(u(i, j)) = \sum_{a \in u(i,j)} l(a)$ soit minimum.

Ce problème a de nombreuses applications pratiques car la "longueur" $l(a)$ peut s'interpréter aussi bien comme un coût de transport sur l'arc, comme les dépenses de construction de l'arc a, comme le temps nécessaire pour parcourir l'arc a, etc. Selon les propriétés du graphe traité (les longueurs sont quelconques, positives ou toutes égales, le graphe est quelconque ou sans circuit) et selon le problème considéré (recherche du plus court chemin d'un sommet à un autre, ou d'un sommet à tous les autres, ou entre tous les couples de sommets), il existe de nombreux algorithmes qui ont été mis au point et qui permettent l'obtention d'une solution. Nous nous contenterons simplement dans la suite de ce chapitre à la définition du problème du plus court chemin selon chaque type de

graphes, et à la description de quelques algorithmes existants, sans explorer toutes les situations.

6. Graphes statiques

6.1. Graphes statiques déterministes

6.1.1. Définition : un graphe $G = (N, A)$ orienté et pondéré est dit statique déterministe si les poids sur tous les arcs sont des valeurs constantes. N représente l'ensemble des nœuds, et A constitue l'ensemble des arcs. Un chemin entre deux nœuds $(i, j) \in N^2$ du graphe est une suite d'arcs liant i à j, ou bien une suite de nœuds qu'on note $Ch(i, j) = (u_0, u_1, \dots, u_n)$ avec $u_0 = i$ et $u_n = j$.

6.1.2. Problème du plus court chemin : Le problème du plus court chemin dans les graphes statiques déterministes est considéré comme un problème classique de la théorie des graphes. Il a attiré l'attention de plusieurs chercheurs et a fait l'objet de nombreuses études. L'objectif consiste à déterminer le plus court chemin entre deux nœuds d'un graphe statique déterministe en définissant la fonction de coût. Dans ce genre de graphe, on définit la fonction de coût notée C d'un chemin $Ch(i, j)$ dans G de la manière suivante :

$$C(u_0, u_1, \dots, u_n) = \begin{cases} \sum_{k=0}^{n-1} FC(u_k, u_{k+1}) & \text{Si } n > 0 \; et \; \forall k, (u_k, u_{k+1}) \in A \\ 0 & \text{Si } n = 0 \\ \infty & \text{Sinon} \end{cases}$$

La fonction $FC(u_k, u_{k+1})$ est la fonction de poids pour les arcs.

Alors le plus court chemin entre i et j qu'on note par $PCCh(i, j)$ est donné par la relation suivante:

$PCCh(i, j) = Ch(i, j)$ avec $C\big(Ch(i, j)\big) = \min_{P \in ECh\,(i,j)}\{C(P)\}$

Si plusieurs chemins existent avec un coût minimal, alors on choisira un aléatoirement.

6.1.3. Algorithme de résolution : de nombreux algorithmes ont été proposés dans la littérature pour résoudre le problème du plus court chemin statique déterministe

à savoir l'algorithme de Bellman-Ford, l'algorithme de Johnson, et le plus connu l'algorithme proposé par Dijkstra [111], que nous présentons dans ce qui suit.
L'algorithme de Dijkstra consiste à déterminer le chemin le plus court entre un nœud origine et tous les sommets d'un graphe pondéré dont le poids est non-négatif. L'algorithme de Dijkstra utilise deux ensembles notés Q et F. L'ensemble des nœuds candidats Q contient tous les nœuds dont le plus court chemin n'a pas encore été calculé. Par contre l'ensemble F des nœuds fermés contient tous les nœuds dont le plus court chemin à été déterminé définitivement. L'algorithme peut aussi être utilisé pour calculer le plus court chemin entre un seul nœud origine et un seul nœud destination. Plusieurs améliorations ont été proposées dans la littérature pour améliorer l'algorithme initial et diminuer sa complexité [37], [39], [40].

Table 1. Algorithme de Dijkstra (G, FC, s)

Step 0 : Initialisation

$\mathrm{PCCh}(s,s) = (s)$ #initialiser le plus court chemin
$\mathrm{PCCh}(s,i) = \emptyset \quad \forall i \in N, i \neq s$
$F = \emptyset$
$Q = \{s\}$ # initialiser Q qui contient au départ que le nœud origine s

Step 1
Tant que $(Q \neq \emptyset)$ faire
    Choisir i dans Q tel que $C(\mathrm{PCCh}(s,i))$ soit minimal # on choisit le sommet avec la
    plus petite valeur de C
    $Q = Q - \{i\}$
    $F = F \cup \{i\}$
    Développer les successeurs j de i
    Pour tout sommet j faire # relâchement de l'arc (i,j)
        Si $C(\mathrm{PCCh}(s,j)) > C(\mathrm{PCCh}(s,i)) + FC(i,j)$ alors #test de plus court ou non?
            $\mathrm{PCCh}(s,j) = \mathrm{PCCh}(s,i) \cup \{j\}$ #mise à jour la distance temporaire
            $Q = Q \cup \{j\}$ #mise à jour la file de priorité
        Fin Si
    Fin Pour
Fin Tant que

La distance temporaire est mise à jour après chaque détente et comparaison d'un arc, ce qui permet de dire que cet algorithme est un algorithme à fixation d'étiquettes "label-setting". La mise à jour peut être une opération de type "insérer" si j n'est pas marqué ou bien une opération de type "diminuer clé". L'exécution de cet algorithme sur un graphe avec n nœuds et m arcs consiste alors à effectuer n opérations d "insertion" dans la file de priorité, n opérations de "suppression", et m opérations de "diminution clé" dans le cas le plus défavorable, ce qui donne une complexité de l'ordre $O(m + n \log n)$ [64]

6.2. Graphes statiques stochastiques

6.2.1. Définition : un graphe $G = (N, A)$ orienté et pondéré est considéré comme un graphe statique stochastique si les poids sur tous les arcs sont définis par des distributions de probabilités discrètes. Alors la fonction de poids FC, dans ce cas, pour un arc (i, j) est définit sous la forme suivante :

$$FC(i,j) = \begin{cases} c_1, & p_1 \\ c_2, & p_2 \\ \vdots & \\ c_m, & p_m \end{cases} \qquad \text{avec} \quad \sum_{k=1}^{m} p_k = 1$$

Avec $c_1, c_{2,\ldots,} c_m$ sont les poids possibles pour l'arc (i, j) et $p_1, p_{2,\ldots,} p_m$ indiquent les probabilités associés aux poids.

6.2.2. Problème du plus court chemin : le problème du plus court chemin dans les graphes statiques stochastiques a été abordé de différentes manières dans littérature, selon la définition de la fonction d'utilité utilisée. La définition la plus courante est celle qui définit comme chemin optimal, le chemin qui maximise la valeur espérée d'une fonction d'utilité [41-42-43-44]. Une autre définition d'un chemin optimal a été donnée dans [45-46] et qui considère qu'un chemin optimal est celui qui maximise la probabilité que la longueur totale du chemin ne dépasse pas une valeur fixe. Les auteurs dans [47-48-49] ont considéré un chemin optimal comme étant celui qui a la plus grande probabilité d'être le chemin le plus court. La plus simple définition utilisée dans la littérature et celle qui décrit l'optimalité d'un chemin par rapport au coût espéré du chemin [50].

6.2.3. Algorithme de résolution : comme nous l'avons précisé précédemment, il existe plusieurs algorithmes pour résoudre le problème du plus court chemin dans le cas statique stochastique. Dans cette sous-section, nous exposons uniquement la version la plus simple qui est l'algorithme proposé par Jamali en 2006. Le principe de cet algorithme est très simple. L'algorithme résout le problème du plus court chemin en calculant les estimations des coûts sur chaque arc (par apprentissage). Ce calcul est fait d'une manière itérative. Chaque nœud du graphe construit un arc avec son nœud voisin, sauf pour le nœud de destination. Sur chaque nœud on a une estimation de la longueur du chemin le plus court qui devrait être mise à jour, et une autre information sur les coûts estimés des liens voisins. Dans cet algorithme, on commence à parcourir le graphe du nœud source au nœud destination. A chaque nœud i, on sélectionne l'arc (i, s) qui conduit au plus petit coût estimé du nœud i vers le noeud destination en fonction de la connaissance du nœud sur l'estimation des coûts dans cette étape :

$$s = \arg \min_{i \in N, (i,j) \in A} \left(c_{ij}^{*} + est_{j} \right)$$

c_{ij}^{*} représente le coût estimé sur l'arc (i, j), et est_{j} est l'estimation apprise sur le coût le plus faible pour atteindre le nœud destination à partir du nœud j. En parcourant le graphe, le coût estimatif de chaque arc sera mis à jour. Lors du passage d'un arc, nous aurions un coût avec une distribution de probabilité qui lui serait affecté. Les mises à jour des estimations sont faites par la relation suivante:

$$c_{ij}^{*}(k+1) = \frac{k \times c_{ij}^{*}(k) + c_{ij}}{k+1}$$

$c_{ij}^{*}(k)$ est l'estimation du coût sur le lien (i, j) à $k^{ième}$ étape. c_{ij} est le coût payé après avoir traversé le lien (i, j).

Par la suite, Jamali a remplacé le coût estimé par la moyenne pondérée du coût estimatif précédent et du coût nouvellement payé. Lorsque l'usager arrive au nœud de destination, on met à jour le coût estimé pour chaque nœud pour atteindre le nœud de destination par la relation suivante :

$$est_i = \min_{(i,j)\in A} c_{ij}^* + est_j$$

Dans cet algorithme, on continue les itérations jusqu'à ce qu'il n'y ait pas de changement ou que la quantité moyenne de changement soit inférieure à une constante donnée.

Cet algorithme est en quelque sorte comme l'algorithme de Dijkstra [111], mais il utilise simplement les connaissances locales sur les nœuds.

7. Graphes dynamiques

7.1. Graphes dynamiques déterministes

7.1.1. Définition : un graphe $G = (N, A)$ orienté et pondéré est dit dynamique déterministe si les poids sur tous les arcs sont des valeurs constantes mais dépendantes du temps. Par conséquent, la fonction de coût pour un lien (i, j) est définit dans ce cas par $FC(i, j, t)$, où t désigne le temps de départ du nœud i. Le temps, pour ce genre de graphe, peut être considéré comme continu ou discret. Mais dans la plus part des cas, et pour simplifier le problème, on considère un temps discret en le divisant en périodes égales de longueurs Δt. La fonction de poids est supposée constante durant chaque période Δt, et le temps n'est pas réinitialisé au niveau de chaque nœud.

Avant de décrire le problème du plus court chemin dans le cas d'un graphe dynamique déterministe, nous abordons d'abord une propriété très importante et utile dans la modélisation d'un réseau. Cette propriété s'appelle la condition FIFO (First In, First Out) qui modélise l'attente à un nœud.

Condition FIFO. Formellement la définition de FIFO pour un graphe est la suivante [35] : un graphe $G = (N, A)$ est un graphe FIFO si tous ses arcs sont des arcs FIFO. Un arc (i, j) est dit FIFO s'il vérifie la propriété suivante :

$$\forall t, t' \geq 0 \Longrightarrow t + FC(i, j, t) \leq t' + FC(i, j, t')$$

Cela signifie qu'en démarrant du nœud i pour aller vers le nœud j à $t' \geq t$ implique nécessairement une arrivée à j plus tard que $t + FC(i, j, t)$. Pour un

exemple pratique, considérant une route à une seule voie. La voiture qui emprunte cette route la première et certaine d'arriver la première car aucun véhicule ne peut la dépasser.

7.1.2. Problème du plus court chemin : le problème du plus court chemin dans le cas des graphes dynamiques a été étudié pour la première fois dans [51] où les auteurs ont proposé un algorithme basé sur le principe d'optimalité de Bellman. Depuis, différentes variantes de ce problème ont été proposées par d'autres chercheurs, comme le problème du plus court chemin dynamique dans le cas discret développé par [52-53-54]. La recherche dans le cas continu est effectuée dans [55-56-57].

Dans le cadre d'un problème du plus court chemin dynamique déterministe, on peut destinguer deux principales problématiques. La première problématique consiste à determiner le plus court chemin d'un noeud origine vers tous les autres noeuds du graphe, pour une heure de départ du noeud origine fixée. La seconde problèmatique est de calculer le plus court chemin d'un noeud origine vers tous les noeuds du graphe mais cette fois pour toutes les heures de départ du noeud origine.

Les fonction de coût d'un chemin et du plus court chemin dans un graphe dynamique deterministe sont définies comme suit :

Etant donné un graphe dynamique deterministe $G = (N, A)$, on note par FC la fonction de coût d'un chemin dans G, et par $FC^{t_0}(u_0, u_1, \dots, u_m)$ le coût d'un chemin $(u_0, u_1, \dots, u_m)$ en partant du noeud u_0 à t_0. La valeur de $FC^{t_0}(u_0, u_1, \dots, u_m)$ est définie par la relation suivante :

$$FC^{t_0}(u_0, u_1, \dots, u_m) = \begin{cases} FC^{t_0}\big((u_0, u_1, \dots, u_{m-1}) + FC(u_{m-1}, u_m, FC^{t_0}(u_0, u_1, \dots, u_{m-1}))\big) & \text{si } m > 0 \\ t_0 & \text{sinon} \end{cases}$$

Le plus court chemin $PCCh^{t_0}(i, j)$ entre i et j en partant du neoud i à l'instant t_0 est alors défini comme suit:

$$PCCh^{t_0}(i, j) = Ch(i, j) \text{ avec } FC^{t_0}\big(Ch(i, j)\big) = \min_{P \in ECh(i,j)}\{FC^{t_0}(P)\}$$

Comme dans le cas statique déterministe, si plusieurs chemins avec un coût minimal existent, alors on choisira arbitrairement un parmi eux. Dans les travaux de [55] il a été montré que déterminer le plus court chemin dynamique pour une heure de départ fixée est *NP-Difficile* , et qu'il est polynomial pour les graphes de type FIFO. [59] et par la suite [58] ont proposé une version dynamique de Dijkstra qui offre une solution optimale dans le cadre des graphes FIFO. Sept ans plus tard, [60] ont proposé un paradigme algorithmique appelé *Chrono-SPT*, pour les graphes non-FIFO. L'idée de ce paradigme consiste à visiter les nœuds dans un ordre chronologique, et il est valable que pour le cas où le temps est discret.

7.1.3. Algorithme de résolution : comme nous l'avons mentionné précédemment, une version de l'algorithme Dijkstra adaptée au cas dynamique existe. Nous présentons dans cette sous section cette version dynamique de l'algorithme de Dijkstra. Considérant un graphe $G(N, A)$ FIFO, avec un nœud origine $s \in N$. Le pseudo-code de cet algorithme qui détermine le plus court chemin en partant du nœud origine s à l'instant t_0 vers tous les nœuds du graphe est présenté dans la table ci-dessous.

Table 2. Version dynamique de l'algorithme de Dijkstra

Step 0 : Initialisation

$PCCh^{t_0}(s, s) = (s)$ #initialiser le plus court chemin
$PCCh^{t_0}(s, i) = \emptyset \quad \forall i \in N, i \neq s$
$F = \emptyset$
$Q = \{s\}$ # initialiser Q qui contient au départ que le nœud origine s

Step 1
Tant que $(Q \neq \emptyset)$ faire
Choisir i de Q tel que $FC^{t_0}\left(PCCh^{t_0}(s, i)\right)$ soit minimal # on choisit le sommet avec la
plus petite valeur de C
$Q = Q - \{i\}$
$F = F \cup \{i\}$
Développer les successeurs j de i
Pour tout sommet j faire # relâchement de l'arc (i, j)
Si

$FC^{t_0}\left(PCCh^{t_0}(s,j)\right) >$
$FC^{t_0}\left(PCCh^{t_0}(s,i)\right) + FC(i,j,FC^{t_0}\left(PCCh^{t_0}(s,i)\right))$ alors #test de plus court ou non?
$PCCh^{t_0}(s,j) = PCCh^{t_0}(s,i) \cup \{j\}$ #mise à jour la distance temporaire
$Q = Q \cup \{j\}$ #mise à jour la file de priorité
Fin Si
Fin Pour
Fin Tant que

7.2. graphes dynamiques stochastiques

7.2.1. Définition : un graphe $G(N,A)$ est dit graphe dynamique si tous les poids sur les arcs sont représentés par des distributions de probabilité discrètes dépendantes du temps. Le poids d'un arc (i,j) dans un graphe dynamique stochastique est présenté par une fonction de coût $FC(i,j,t)$, où t désigne l'instant de départ du nœud i. Cette fonction de coût est déterminée à chaque instant t par une distribution de probabilité de la forme:

$$FC(i,j,t) = \begin{cases} c_1^t, & p_1^t \\ c_2^t, & p_1^t \\ \vdots & \\ c_m^t, & p_m^t \end{cases} \qquad \text{avec} \quad \sum_{k=1}^{m} p_k^t = 1$$

où $c_1^t, c_2^t, \dots, c_m^t$ représentent les poids possibles d'un lien (i,j) à l'instant t et $p_1^t, p_2^t, \dots, p_m^t$ sont les probabilités associées aux poids à l'instant t.

7.2.2. Problème du plus court chemin : le problème du plus court chemin dynamique stochastique a reçu une grande attention dans différents domaine. Le premier qui a abordé ce problème est Hall [61]. Il a évoqué le problème de déterminer le plus court chemin entre deux nœuds avec des temps de déplacement qui sont à la fois aléatoires et dépendants du temps. Il a montré que les algorithmes standards appliqués dans le cas statique stochastique ne sont pas efficaces sur des réseaux dynamiques stochastiques. Il a montré que le «choix de route» optimal n'est pas un calcul à priori d'un chemin mais une règle de décision adaptative, et que le meilleur itinéraire à partir d'un nœud donné à la destination

finale dépend de l'heure d'arrivée à ce nœud. [62] a montré que le calcul à priori d'un chemin avec un coût espèré minimal est un problème NP-Difficile. Une solution à ce problème à été porté dans [63] en proposant un algorithme à correction de label appelé LET (Least Expected Time) à priori. Le problème de trouver une strategie adaptative a reçu plus d'attention que celui de trouver un chemin à priori. Dans la littérature, plusieurs algorithmes de routage adaptatif ont été proposés sous différentes approches pour résoudre ce problème. La plupart des algorithmes adaptatifs pour un chemin optimal sont basés sur la minimisation de coût espéré comme la version adaptative de LET proposé dans [118]. Cependant, il existe plusieurs cas dans lesquels la solution LET n'est pas adéquate, car elle ne tient pas compte de la variance des distributions des temps de parcours sur les arcs et n'offre aucune garantie de fiabilité. Le chemin optimal défini par la solution LET peut être peu fiable et peut entraîner des réalisations très variables du temps de parcours. Dans de nombreux cas, les voyageurs ont des délais difficiles. Par exemple dans le routage commercial, il existe des garanties de livraison qui doivent être respectées et des produits périssables qui doivent être livrés dans un délai fixe. [45] présente une définition très naturelle d'un chemin optimal fiable, comme le chemin qui maximise la probabilité de réaliser un temps de déplacement inférieur à une constante donnée. Cependant, la formulation donnée par Frank exige l'énumération de tous les chemins possibles et n'est donc pas traitable pour des problèmes pratiques. [123] considèrent la formulation de Frank également connu sous le nom du problème SOTA (Stochastic On Time Arrival) et le formulent comme un problème de programmation dynamique stochastique. Le programme dynamique proposé est résolu à l'aide d'un algorithme d'approximation successive standard (SA) [124].

Le travail effectué dans notre thèse est basé sur cette dernière approche (Problème SOTA) que nous allons presenter d'une manière détailleé dans le chapitre qui suit.

7.2.3. Algorithme de résolution : comme nous venons de le voir, plusieurs algorithmes ont été developpés dans la litterature pour résoudre le plroblème du plus court chemin dans le cas dynamique stochastique et deux approches ont été proposées, à savoir l'approche qui vise à calculer un chemin à priori, et

l'approche qui détermine une strategie de routage en ligne permettant de réagir aux poids effectifs des noeuds intermédiaires pour choisir efficacement les noeuds suivants. Dans cette partie nous presentons la formulation du problème SOTA comme elle a été présentée pour la première fois par les auteurs de [123].

Etant donné un graphe $G(N, A,)$, on note par $u_i(t)$ la probabilité maximale d'arriver au noeud destination d à partir du noeud i avec un budget inferieur ou égal à t. Si un usager qui est au noeud i choisit de visiter le noeud successeur j, alors la probabilité de temps de parcous sur le lien (i, j) est donnée par $p(\omega)d\omega$. Le temps restant pour atteindre la destination d à partir du noeud j est alors $t - \omega$. Sur la base du principe de Bellman, peu importe le nœud que le voyageur choisit de visiter, le voyageur doit atteindre la meilleure valeur de la fonction $u_j(t - \omega)$ en partant du nœud j vers la destination d avec le temps restant $t - \omega$. Il peut y avoir plusieurs nœuds j qui peuvent être visités à partir du nœud actuel i. Le voyageur devrait choisir le nœud j qui fournit la probabilité maximale d'arrivér à la destination à partir du nœud i. L'application du principe de Bellman dans le contexte du problème SOTA fournit le système d'équations non linéaires suivant :

$$u_i(t) = \max_{i \neq j} \int_0^t p_{ij}(\omega) u_j(t - \omega) d\omega \ , \qquad 0 \leq t < \infty$$

$$u_d(t) = 1$$

$p_{ij}(\omega)d\omega$ est la distribution des temps de parcours sur le lien (i, j) . $u_i(t)$ c'est la probabilité qu'à partir du nœud i, un voyageur arrive au nœud destination d avec un budget t. La stratégie $s_i(t)$ optimale est donnée comme suit.

$$s_i(t) = \arg\max_{i \neq j} \int_0^t p_{ij}(\omega) u_j(t - \omega) d\omega \ , \qquad 0 \leq t < \infty$$

L'une des approches possibles pour résoudre ce système d'équations est la méthode de Picard basée sur les approximations itératives appelée SA (Successive Approximation). Pour plus de détails sur cette approche, les lecteurs peuvent se référer à [66] et [127].

8. Conclusion

Cette deuxième partie de ce chapitre introductif a fait l'objet d'un état de l'art sur le problème et les algorithmes du plus court chemin qui constituent la problématique principale traitée dans notre sujet de thèse. Nous avons débuté cette partie par une brève introduction sur la théorie des graphes qui est considérée comme l'un des outils essentiels pour les problèmes d'acheminement. Puis nous avons résumé les différentes méthodes de représentation d'un graphe. Par la suite, nous avons défini les différentes classes de graphes. Pour chaque classe de graphes, le problème du plus court chemin a été donné, et quelques algorithmes de résolution ont été présentés à la fin de cette partie. Nous reportons à la troisième partie la description des problèmes d'optimisation robuste et les différents concepts de robustesse existant dans la littérature.

Partie 3

Optimisation robuste et critères de prise de décision dans un environnement incertain

Sommaire

1. Introduction

L'optimisation robuste est un domaine de recherche jeune et émergent qui a reçu une attention considérable au cours de la dernière décennie. De la même façon que l'approche de l'optimisation stochastique, l'optimisation robuste concerne des modèles dans lesquels les données exactes sont inconnues, mais limitées par un ensemble de réalisations possibles (ou de scénarios). Les premières étapes de l'optimisation robuste remontent au travail de Soyster [67]. L'optimisation robuste n'est apparue comme un domaine de recherche à part entière que vers la fin des années 90 avec les travaux centraux de [68-69]. Dans cette troisième partie de ce premier chapitre, nous décrivons plus en détail le contexte général de l'optimisation robuste, puis nous donnerons un résumé des différents concepts de robustesse existants.

2. Les problèmes d'optimisation incertains

Presque chaque problème d'optimisation souffre d'incertitude avec un certain degré, même si cela ne semble pas être le cas à première vue. D'une manière générale, on peut distinguer deux types d'incertitude. L'incertitude microscopique, comme les erreurs numériques et les erreurs de mesure; et l'incertitude macroscopique, telle que les erreurs de prévision, les perturbations, ou d'autres conditions modifiant l'environnement où une solution est mise en œuvre. Dans l'optimisation "classique", on définit ce qu'on appelle "scénario nominal", qui décrit le comportement attendu ou "le plus typique" des données incertaines. Selon le type d'incertitude, ce scénario peut être, par exemple, le coefficient de la précision donnée pour les erreurs numériques, la valeur mesurée pour les erreurs de mesure, la prévision la plus probable pour les erreurs de prévision ou tout simplement un environnement pour les solutions à long terme. Selon l'application, calculer un tel scénario peut être un processus non trivial, voir par exemple [71].

Nous considérons des problèmes d'optimisation décrits sous la forme :

$$(P) \qquad \begin{aligned} &\min f(x) \\ &\text{s.t. } F(x) \leq 0 \\ &x \in \chi \end{aligned}$$

Où $F : \mathbb{R}^n \to \mathbb{R}^m$ décrit le problème à m contraintes. $f : \mathbb{R}^n \to \mathbb{R}$ est la fonction objectif. $\chi \subseteq \mathbb{R}^n$ est l'espace variable. Dans des applications réelles, les contraintes et la fonction objectif peuvent dépendre des paramètres incertains. Pour tenir compte de ces incertitudes, au lieu de (P), on considère la famille de problèmes paramétrés suivante :

$$(P(\xi)) \qquad \begin{aligned} &\min f(x,\xi) \\ &\text{s.t. } F(x,\xi) \leq 0 \\ &x \in \chi \end{aligned}$$

$F(x,\xi) : \mathbb{R}^n \to \mathbb{R}^m$ et $f(x,\xi) : \mathbb{R}^n \to \mathbb{R}$ pour n'importe quelle valeur fixe de $\xi \in \mathbb{R}^M$. Chaque ξ décrit un scenario qui peut se produire. Souvent, en pratique, on ne peut pas savoir exactement quelle valeur qu'un tel scenario ξ peut prendre pour le problème d'optimisation $P(\xi)$. On suppose qu'il est connu que ξ réside dans un ensemble d'incertitude donné $\mathcal{U} \subseteq \mathbb{R}^M$. Un tel ensemble d'incertitude représente les scénarios qui sont susceptibles d'être considérés [73]. Le problème d'optimisation incertain qui correspond à $P(\xi)$ est alors donné par $(P(\xi), \xi \in \mathcal{U})$

Le problème d'optimisation incertaine consiste en fait en tout un ensemble de problèmes paramétrés, qui sont souvent infiniment grand. Le but des concepts d'optimisation robustes est de transformer cette famille de problèmes en un seul problème, qui s'appelle la contrepartie robuste. Le choix de l'ensemble de l'incertitude est d'un impact majeur non seulement pour l'application, mais aussi pour la complexité informatique de la contrepartie robuste qui en résulte. Il doit donc être choisi avec soin par le modéliste [72].

Pour un problème d'optimisation incertain donné $(P(\xi), \xi \in \mathcal{U})$, on note l'ensemble réalisables de scenarios $\xi \in \mathcal{U}$ par :

$$\mathcal{F}(\xi) = \{x \in \chi : F(x,\xi) \leq 0\}$$

En outre, s'il existe un scénario nominal, il est noté par $\hat{\xi} \in \mathcal{U}$. La valeur optimale de la fonction objectif pour le scenario $\xi \in \mathcal{U}$ est notée par $f^*(\xi)$.

On dit qu'un problème d'optimisation incertain $(P(\xi), \xi \in \mathcal{U})$ est convexe (quasi-convexe, affine, linéaire) lorsque les deux fonctions $F(x,.) : \mathcal{U} \to \mathbb{R}^m$ et $f(x,.) : \mathcal{U} \to \mathbb{R}$ sont convexes (quasi-convexes, affines, linéaires).

Les différentes classes d'incertitude. Il existe certains types d'ensembles d'incertitude qui sont fréquemment utilisés dans la littérature. Ceux-ci inclus:

1)- Incertitude finie : $\mathcal{U} = \{\xi^1, \xi^2, \ldots., \xi^N\}$.

2)- Incertitude bornée par intervalle : $\mathcal{U} = \left[\underline{\xi}_1, \bar{\xi}_1\right] \times \ldots.\times \left[\underline{\xi}_M, \bar{\xi}_M\right]$.

3)- Incertitude polytopique : $\mathcal{U} = \text{conv}\,\{\xi^1, \xi^2, \ldots., \xi^N\}$.

4)- Incertitude bornée par norme : $\mathcal{U} = \left\{\xi \in \mathbb{R}^m : \left\|\xi - \hat{\xi}\right\| \leq \alpha\right\}$ pour $\alpha \geq 0$.

5)- Incertitude ellipsoïdale : $\mathcal{U} = \left\{\xi \in \mathbb{R}^m : \sqrt{\sum_{i=1}^{M} \xi_i^2/\sigma_i^2} \leq \Omega\right\}$ pour $\Omega \geq 0$.

3. Concepts et critères de robustesse

Une optimisation robuste a commencé par des concepts assez conservateurs couvrant tout ce qui est considéré comme susceptible de se produire. Ces concepts ont été développés grâce aux différentes situations et applications appelant à des solutions «robustes». Dans cette section, nous donnons un aperçu des critères anciens les plus importants et des critères récents.

3.1. Strict Robustness

Cette approche est connue aussi parfois sous le nom d'optimisation classique robuste, optimisation min-max, ou simplement optimisation robuste. Une solution $x \in \chi$ au problème d'optimisation incertain $(P(\xi), \xi \in \mathcal{U})$ est dite robuste absolue si elle est réalisable pour tous les scenarios dans $\mathcal{U}$ c.à.d. $F(x, \xi) \leq 0$ pour tout $\xi \in \mathcal{U}$. L'objectif suit habituellement la vue pessimiste de minimiser le pire des cas sur tous les scénarios. On note par $SR(\mathcal{U})$ l'ensemble des solutions robustes absolues par rapport à l'ensemble incertain $\mathcal{U}$.

$$SR(\mathcal{U}) = \bigcap\nolimits_{\xi \in \mathcal{U}} \mathcal{F}(\xi)$$

Le problème d'optimisation équivalent est donné comme suit.

$$(SR) \qquad \begin{array}{l} \min \ \sup_{\xi \in \mathcal{U}} \ f(x,\xi) \\ \text{s.t. } x \in \mathrm{SR}(\mathcal{U}) \\ x \in \chi \end{array}$$

Les décisions prises dans ce cas sont de nature conservatrice, car elles sont basées sur l'anticipation que le pire pourrait arriver. Le premier qui a étudié la robustesse absolue à partir d'une perspective d'un programme linéaire généralisé est Soyster en 1973. Il a considéré un ensemble incertain $\mathcal{U}$ du type $\mathrm{K}_{1,}\mathrm{K}_{2},\ldots\ldots,\mathrm{K}_{\mathrm{n}}$, où K_{i} contient tous les vecteurs colonnes possibles A_{i} de la matrice des coefficients A. Par la suite, plusieurs travaux ont été menés par des chercheurs dans le cadre de cette approche, voir par exemple [74-75]. Cependant, la formulation et la construction de cette approche dans un cadre théorique solide est due à une série de travaux de [65], [76], [68] et [70]. Leurs travaux ont été résumés dans le livre "*Robust optimization*" [77].

L'idée sous-jacente de base est de se prémunir contre tous les scénarios qui peuvent survenir. Comme ils le font valoir, une telle approche est logique dans de nombreux contextes, par exemple pour les avions ou pour les centrales nucléaires. Cependant, ce degré élevé de conservatisme de robustesse absolue ne s'applique pas à toutes les situations nécessitant des solutions robustes. Un exemple pour cela est l'horodatage dans les transports publics : être strictement robuste pour un calendrier signifie que tous les horaires d'arrivée et de départ annoncés doivent être respectés, peu importe ce qui se passe. Cela peut signifier ajouter des temps de tampon élevés, en fonction de l'ensemble d'incertitude utilisé, et donc n'entraînerait pas un calendrier pratiquement applicable. De telles applications ont déclenché des recherches en optimisation robuste sur les façons d'étendre le concept. Nous décrivons certaines de ces approches dans ce qui suit.

3.2. Cardinality Constrained Robustness

L'une des possibilités de résoudre le problème du conservatisme de la robustesse absolue est de réduire l'ensemble des incertitudes $\mathcal{U}$. Cela a été introduit dans [78] pour des problèmes de programmation linéaire. Pour cette raison, ce concept est parfois aussi connu sous le nom de "*l'approche de Bertsimas et Sim*" ou encore Γ-Robustesse. En analysant la structure de l'incertitude dans les applications typiques, Bertsimas et Sim ont observé qu'il est peu probable que tous les coefficients d'une contrainte changent simultanément à leurs valeurs les plus défavorables. Au lieu de cela, ils proposent de se protéger uniquement contre

des scénarios pour lesquels les paramètres les plus incertains par contrainte changent à leurs valeurs les plus défavorables. En faisant ça, on limite le nombre de coefficients qui sont autorisés à changer, en conduisant à la notion de robustesse à cardinalité limitée. Considérant une contrainte de la forme

$$a_1x_1 + \cdots + a_nx_n \leq b$$

Avec une incertitude $\mathcal{U} = \{a \in \mathbb{R}^n : a_i \in [\hat{a}_i - d_i, \hat{a}_i + d_i], i = 1, \ldots . n\}$. Le concept de robustesse à cardinalité limitée nécessite une solution x pour satisfaire:

$$\sum_{i=1}^{n} \hat{a}_i x_i + \max_{S \subseteq \{1,\ldots.n\}, |S|=\Gamma} \left\{ \sum_{i \in S} d_i |x_i| \right\} \leq b$$

Pour un paramètre $\Gamma \in \{1, \ldots . n\}$ donné. Toute solution x à ce modèle se protège donc de tous les scénarios dans lesquels un plus grand nombre Γ de coefficients incertains peuvent s'écarter de leurs valeurs nominales en même temps. Il a été montré que la robustesse de cardinalité limitée peut également être considérée comme une robustesse absolue en utilisant l'enveloppe convexe de l'ensemble d'incertitude à cardinalité limitée :

$$\mathcal{U}(\Gamma) = \{a \in \mathcal{U} : a_i \neq \hat{a}_i, \text{pour } \Gamma \text{ d}' \text{indices i au plus}\} \subseteq \mathcal{U}$$

Comme $conv(\mathcal{U}(\Gamma))$ est un ensemble polyédrique, les résultats sur une robustesse absolue par rapport à l'incertitude polyédrique peuvent également être appliqués à la robustesse à cardinalité limitée. Le concept de cette approche s'étend également aux valeurs fractionnaires de Γ. Leur concept a été étendu aux ensembles d'incertitude selon les normes générales dans [78]. L'approche des problèmes d'optimisation combinatoire a été généralisée dans [79-80].

3.3. Adjustable Robustness

Dans [81], une observation complètement différente des cas se produisant dans des problèmes réels avec des données incertaines est utilisée. Souvent, les variables peuvent être décomposées en deux ensembles. Les valeurs des variables *here-and-now* doivent être trouvées à l'avance par l'algorithme d'optimisation robuste, alors que la décision concernant

les variables *wait-and-see* peut attendre que le scénario actuel $\xi \in \mathcal{U}$ devienne connu. Cela est similaire au principe de la programmation en deux étapes en optimisation stochastique.

Pour cette approche, on suppose que les variables x sont divisées en $u \in \chi^1 \subseteq \mathbb{R}^{n_1}$ et en $v \in \chi^2 \subseteq \mathbb{R}^{n_2}$, tels que $n_1 + n_2 = n$, où les variables u doivent être déterminées avant que le scenario $\xi \in \mathcal{U}$ devienne connu, tandis que les variables v peuvent être déterminées après la réalisation du scenario ξ . Ainsi, on peut également écrire $x(\xi)$ pour souligner la dépendance de v sur les scenarios. Le problème d'optimisation incertain $(P(\xi), \xi \in \mathcal{U})$ peut être alors réécrit de la façon suivante.

$$(P(\xi)) \qquad \begin{array}{l} \min f(u, v, \xi) \\ \text{s.t. } F(u, v, \xi) \leq 0 \\ (u, v) \in \chi^1 \times \chi^2 \end{array}$$

Lorsque on fixe les variables *here*-and-*now*, on doit s'assurer que pour tout scenario possible $\xi \in \mathcal{U}$, il existe $v \in \chi^2$ tel que (u, v) est réalisable pour ξ. L'ensemble des solutions robustes réglables est donc donné par :

$$\begin{aligned} aSR &= \{u \in \chi^1 : \forall \xi \in \mathcal{U} \ \exists v \in \chi^2 \ \text{ s.t. } (u, v) \in \mathcal{F}(\xi)\} \\ &= \bigcap\nolimits_{\xi \in \mathcal{U}} \mathrm{Pr}_{\chi^1} \mathcal{F}(\xi) \end{aligned}$$

où $\mathrm{Pr}_{\chi^1} \mathcal{F}(\xi) = \{u \in \chi^1 : \exists v \in \chi^2 \text{ s.t. } (u, v) \in \mathcal{F}(\xi)\}$ représente la projection de $\mathcal{F}(\xi)$ sur χ^1. La fonction objectif dans le pire cas pour certains $u \in aSR$ est donnée par

$$z^{aSR}(u) = \sup_{\xi \in \mathcal{U}} \inf_{v:\,(u,v) \in \mathcal{F}(\xi)} f(u, v, \xi)$$

La solution robuste réglable est donnée par

$$\min\{z^{aSR}(u) : u \in aSR\}$$

Il existe plusieurs variantes du concept de robustesse réglable. Au lieu de deux étapes, plusieurs étapes sont possibles. Par exemple, les auteurs dans [82] ont calculé un ensemble de solutions statiques possibles au lieu de calculer une nouvelle solution pour chaque scénario. Cet ensemble a été calculé de sorte qu'au moins une des solutions de cet ensemble est réalisable à chaque étape. En outre, le développement de la robustesse réglable a été

précédé par l'approche similaire de [83], où les auteurs ont considéré un problème d'optimisation linéaire incertain de la forme

$$\big(\mathrm{P}(\mathrm{B},\mathrm{C},\mathrm{e})\big) \qquad \begin{aligned} &\min \mathrm{c}^{\mathrm{t}}\mathrm{u} + \mathrm{d}^{\mathrm{t}}\mathrm{v} \\ &\mathrm{s.t.}\ \ \mathrm{Au} = \mathrm{b} \\ &\quad\ \mathrm{Bu} + \mathrm{Cv} = \mathrm{e} \\ &\quad\ \mathrm{u} \in \mathbb{R}_{+}^{n_1}, \mathrm{v} \in \mathbb{R}_{+}^{n_2} \end{aligned}$$

Où u représente un vecteur de variables de *conception* qui ne peut pas être ajusté, et v est un vecteur de variables de *contrôle* qui peut être ajusté lorsque le scénario réalisé devient connu. Pour un ensemble incertain fini $\mathcal{U} = \{(\mathrm{B}^1, \mathrm{C}^1, \mathrm{e}^1), \dots, (\mathrm{B}^{\mathrm{N}}, \mathrm{C}^{\mathrm{N}}, \mathrm{e}^{\mathrm{N}})\}$, sa solution robuste est donnée par

$$\text{(Mulvey)} \qquad \begin{aligned} \min\ \ & \sigma\big(\mathrm{u}, \mathrm{v}^1, \dots., \mathrm{v}^{\mathrm{N}}\big) + \omega\ \rho\big(\mathrm{z}^1, \dots., \mathrm{z}^{\mathrm{N}}\big) \\ \mathrm{s.t.}\ \ & \mathrm{Au} = \mathrm{b} \\ & \mathrm{B}^{\mathrm{i}}\mathrm{u} + \mathrm{C}^{\mathrm{i}}\mathrm{v}^{\mathrm{i}} + \mathrm{z}^{\mathrm{i}} = \mathrm{e}^{\mathrm{i}}, \forall \mathrm{i} = 1, \dots., \mathrm{N} \\ & \mathrm{u} \in \mathbb{R}_{+}^{n_1}, \mathrm{v}^{\mathrm{i}} \in \mathbb{R}_{+}^{n_2}, \mathrm{z}^{\mathrm{i}} \in \mathbb{R}^{\mathrm{m}} \end{aligned}$$

z^{i} est un ensemble de vecteurs d'erreur qui mesureront l'*infaillibilité* autorisée dans les contraintes de contrôle sous scénario i. La fonction σ représente la *robustesse de la solution*. Elle peut être modélisée comme la fonction la plus défavorable (le pire-cas) de la fonction objectif nominale.

$$\sigma\big(\mathrm{u}, \mathrm{v}^1, \dots., \mathrm{v}^{\mathrm{N}}\big) = \mathrm{c}^{\mathrm{t}}\mathrm{u} + \max_{\mathrm{i}=1,\dots.,\mathrm{N}} \mathrm{d}^{\mathrm{t}}\mathrm{v}^{\mathrm{i}}$$

La fonction ρ représente la *robustesse du modèle* et dépend de l'infaillibilité des contraintes incertaines. Les fonctions de pénalité possibles sont

$$\rho\big(\mathrm{z}^1, \dots., \mathrm{z}^{\mathrm{N}}\big) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{p}_{\mathrm{i}} \sum_{\mathrm{j}=1}^{\mathrm{m}} \max\{0, \mathrm{z}_{\mathrm{j}}^{\mathrm{i}}\}$$

ou bien $\rho\big(\mathrm{z}^1, \dots., \mathrm{z}^{\mathrm{N}}\big) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{p}_{\mathrm{i}} \left(\mathrm{z}^{\mathrm{i}}\right)^{\mathrm{t}} \mathrm{z}^{\mathrm{i}}$

Comme (Mulvey) est un modèle bi-critère, alors ω est utilisé comme facteur de scalarisation pour combiner les deux objectifs, et p_{i} représente la probabilité du scénario i.

3.4. Light Robustness

Une autre façon de détendre le conservatisme est donnée dans le concept de *Light Robustness*. Ce concept couple une optimisation robuste avec une approche simplifiée de programmation stochastique en deux étapes, et présente un certain nombre d'avantages importants en termes de flexibilité et de facilité d'utilisation. Cette approche à été proposée dans [85]. Ces auteurs ne sont intéressés que par des solutions qui ne sont pas trop conservatrices et, par conséquent, ajoutent une limite à la valeur nominale de la solution. Dans toutes les solutions qui satisfont cette contrainte, on choisit celle qui dévie moins les contraintes dans le pire des cas. Le concept a été appliqué à des problèmes de programmation linéaire et à des ensembles d'incertitude basés sur des intervalles. Dans une étude de cas, les auteurs de [85] montrent que leur approche convient aux problèmes de chronométrage. Depuis, l'idée de *Light Robustness* a été appliquée à un chronométrage rigoureux des chemins de fer dans [86], à l'information du calendrier dans [87]. Cette approche a été comparée à d'autres concepts de robustesse dans une étude sur le calendrier apériodique dans [89]. *Light Robustness* a également été mentionnée dans [102], et étudiée dans le cadre d'une approche unifiée pour la robustesse dans [90]. Dans [84], les auteurs ont étendu l'idée de Fischetti et Monaci et ont développé une notion générale de *Light Robustness* appelée *Generalized Light Robustness* qui s'applique à tous les types de problèmes d'optimisation et à des ensembles d'incertitude arbitraires. Le problème d'optimisation pour Light Robustness décrit par Schobel [84], a la forme suivante.

$$
\text{(LR)} \qquad
\begin{aligned}
&\min \sum_{i=1}^{m} \omega_i \gamma_i \\
&\text{s.t. } f(x, \hat{\xi}) \leq f^*(\xi) + \rho \\
&\quad F(\xi) \leq \gamma, \forall\, \xi \in \mathcal{U} \\
&\quad x \in \chi, \gamma \in \mathbb{R}^m
\end{aligned}
$$

Où ω_i modélise un poids de pénalité pour la violation des contraintes i et ρ détermine la qualité nominale requise. On désigne par $\hat{\xi}$ le scénario nominal. Cette approche a été, dans sa première application dans [86], utilisée comme un développement ultérieur de la notion de *Cardinality Constrained Robustness*. Les variables γ_i agissent comme des variables de recours en deuxième étape utilisées pour se remettre d'une éventuelle infaillibilité, dont la

somme pondérée est minimisée par la fonction objectif. Chaque variable γ_i définit le niveau de robustesse de la solution par rapport à l'incertitude des paramètres.

3.5. Recoverable Robustness

Similaire à l'approche *Adjustable Robustness*, *Revovrable Robustness* est aussi un concept à deux étapes. Ce concept a été développé dans [91-92-93-94], et utilisé dans [95]. L'idée de base de cette approche consiste à autoriser une classe d'algorithmes de récupération $\mathcal{A}$ qui peuvent être utilisés en cas de perturbation. Une solution x est dite *Recovery Robust* par rapport à $\mathcal{A}$ si pour tout scénario possible $\xi \in \mathcal{U}$, il existe un algorithme $A \in \mathcal{A}$ de sorte que A appliqué à la solution x et au scénario ξ construit une solution $A(x, \xi) \in \mathcal{F}(\xi)$, c.à.d. une solution qui est réalisable pour le scénario actuel. Le programme d'optimisation selon [93] est le suivant :

$$(RR) \qquad \min_{(x,A)\in\mathcal{F}(\xi)\times\mathcal{A}} f(x) \quad \text{s.t. } A(x,\xi) \in \mathcal{F}(\xi), \forall\, \xi \in \mathcal{U}$$

Dans [87-88] et [96-97], la notion de robustesse récupérable a été considérée sous usage de métriques pour mesurer les coûts de récupération. L'objectif est de minimiser les coûts lors de la récupération, où ils diffèrent entre la récupération vers une solution faisable ("*recovery-to-feasibility*") et la récupération vers une solution optimale ("*recovery-to-optimality*") dans le scénario réalisé.

3.6. Regret Robustness

Le concept de *Regret Robustness* diffère des autres concepts de robustesse présentés, dans la mesure où il ne considère généralement que l'incertitude dans la fonction objectif. Au lieu de minimiser les performances les plus défavorables d'une solution, il minimise la différence entre la fonction objectif et la meilleure solution qui aurait été possible dans un scénario. Dans certaines publications, on parle aussi de déviation robuste.

Soit $f^*(\xi)$ la meilleure valeur (optimale) pour le scenario $\xi \in \mathcal{U}$, alors la solution de problème d'optimisation avec la déviation robuste est donnée par

$$\text{(Regret)} \qquad \begin{array}{c} \min \ \sup_{\xi \in \mathcal{U}} \ \left(f(x,\xi) - f^*(\xi)\right) \\ \text{s.t. } F(x) \leq 0 \\ x \in \chi \end{array}$$

Regret robustness est un concept avec une grande quantité d'applications, par exemple, dans la théorie de l'emplacement ou dans la planification. Pour plus de détails veuillez consulter [98-99]. Dans un esprit similaire, le concept de α-robustesse lexicographique a récemment été proposé [100]. Son idée de base est d'évaluer une solution fixe en réordonnant l'ensemble des scénarios en fonction de la performance de la solution. Cette courbe de performance est ensuite comparée à une courbe idéale, où le problème d'optimisation est résolu séparément pour chaque scénario.

3.7. Quelques concepts de robustesse supplémentaires

3.7.1. Reliability. Une autre approche de l'optimisation robuste est de relâcher les contraintes de robustesse absolue. Cela conduit à la notion de *fiabilité* de [101] dans laquelle les contraintes $F(x,\xi) \leq 0$ sont remplacées par $F(x,\xi) \leq \gamma$, pour certaines valeurs de $\gamma \in \mathbb{R}^m_{\geq 0}$. Une solution x qui satisfait $F(x,\xi) \leq \gamma$ pour tous les scenarios $\xi \in \mathcal{U}$ est dite solution fiable par rapport à γ. L'objectif est de trouver une solution fiable qui minimise la fonction objectif initiale dans le pire cas. Le programme d'optimisation fiable est donné alors sous la forme :

$$\text{(Reliability)} \qquad \begin{array}{c} \min \ \sup_{\xi \in \mathcal{U}} \ f(x,\xi) \\ \text{s.t. } F(x,\xi) \leq \gamma, \xi \in \mathcal{U} \\ x \in \chi \end{array}$$

De même que pour l'approche de *Light Robustness*, il faut veiller à ce que la représentation des contraintes n'affecte pas la fiabilité de la solution

3.7.2. Soft Robustness. L'idée de base de cette approche telle que introduite dans [102] est de gérer le conservatisme de l'approche Strict Robustness en considérant une famille imbriquée d'ensembles d'incertitude, et autorisant une plus grande déviation dans les contraintes pour des incertitudes plus importantes. Plus précisément, au lieu d'un ensemble d'incertitude $\mathcal{U} \subseteq \mathbb{R}^M$, on utilise une famille d'incertitudes $\{\mathcal{U}(\varepsilon) \subseteq \mathcal{U}\}_{\varepsilon > 0}$ avec $\mathcal{U}(\varepsilon_1) \subseteq \mathcal{U}(\varepsilon_2)$ pour tous les

$\varepsilon_2 \geq \varepsilon_1$. L'ensemble des solutions qualifiées de Soft Robustness est alors donné par :

$$\text{sofR} = \{x \in \chi : F(x, \xi) \leq \varepsilon, \forall \varepsilon \in \mathcal{U}(\varepsilon), \varepsilon > 0\}$$

On note que l'approche de Strict Robustness est un cas spécial avec $\{\mathcal{U}(\varepsilon) = \mathcal{U}\}_{\varepsilon>0}$.

3.7.3. Comprehensive Robustness. Bien que l'approche *Adjustable Robustness* détend l'hypothèse que toutes les décisions doivent être prises avant que le scénario réalisé ne soit connu, l'approche de *Comprehensive Robustness* supprime également l'hypothèse selon laquelle seuls les scénarios définis dans l'ensemble d'incertitude $\mathcal{U}$ doivent être considérés [103]. Au lieu de cela, en utilisant une mesure de distance $dist$ dans l'espace des scénarios, et une mesure de distance $\overline{\text{dist}}$ dans l'espace de la solution, les auteurs supposent que plus un scénario est loin de l'ensemble d'incertitude, plus la solution correspondante est autorisée à être loin de l'ensemble de solutions réalisables. Comme dans l'approche de *Adjustable Robustness*, la dépendance entre la solution x et le scenario ξ est autorisée, et on peut écrire $x(\xi)$. Le programme d'optimisation dans le cadre de l'approche *Comprehensive Robustness* est de la forme:

$$\text{(CR)} \qquad \begin{aligned} &\min z \\ &\text{s.t. } f(x(\xi), \xi) \leq z + \alpha_0 \, \text{dist}(\xi, \mathcal{U}), \forall \xi \\ &\quad \overline{\text{dist}}\big(x(\xi), \mathcal{F}(\xi)\big) \leq \alpha \, \text{dist}(\xi, \mathcal{U}) \, , \forall \xi \end{aligned}$$

Où α_0 et α représentent les paramètres de sensibilité. Cette formule nécessite d'autres spécifications formelles qui sont données dans [103].

3.7.4. Uncertainty Feature Optimization. Au lieu de supposer qu'un ensemble d'incertitude explicite est donné, ce qui peut être difficile à modéliser pour les problèmes réels, l'approche de l'optimisation des fonctionnalités d'incertitude (UFO) suppose plutôt que la robustesse d'une solution est donnée par une fonction explicite [104]. Pour un problème d'optimisation incertain $\big(P(\xi)\big)$, soit $\mu : \mathbb{R}^n \to \mathbb{R}^p$ une mesure de p performances robustes. Le programme

d'optimisation dans le cadre de cette approche (UFO) est donné sous la forme suivante :

$$(\text{UFO}) \qquad \begin{aligned} &\text{vecmax } \mu(x) \\ &\text{s.t. } F(x) \leq 0 \\ &\quad f(x) \leq (1+\rho) f^*(\hat{\xi}) \\ &\quad x \in \chi \end{aligned}$$

où $f^*(\hat{\xi})$ présente la meilleure valeur au problème nominal. Les auteurs ont montré que cette approche généralise à la fois l'optimisation stochastique et le concept de *Cardinality Constrained Robustness* de Bertsimas et Sim.

4. Conclusion

Dans cette dernière partie de ce premier chapitre, nous avons donné une vaste collection de concepts de robustesse dynamique, offrant chacun leurs avantages et inconvénients. [67] a introduit le concept de *Strict Robstness*. Ce concept a été illustré dans plusieurs exemples (par exemple, pour la programmation linéaire dans [71], ou pour le bras d'une porte à feu dans [68]), et analysé pour ces exemples de manière mathématique. L'analyse dans ces articles a montré que la complexité du problème augmente en introduisant la robustesse. Le critère *Strict Robstness* tend à conduire à des décisions qui sont très conservatrices dans la nature et la principale préoccupation est de savoir comment se prémunir contre le pire événement possible. Pour faire face au caractère de conservatisme, d'autres concepts moins conservateurs ont été proposés dans la littérature. Des idées ont été prises en compte dans [78] en introduisant leur nouvelle notion de *Cardinality Constrained Robustness*, qui est moins conservatrice et plus abordable sur le plan informatique, mais qui peut être appliquée uniquement à des ensembles d'incertitude plus faciles à manipuler. L'application de ce concept pour la planification des trains a été le point de départ de [86] qui ont assoupli les contraintes et ont développé le concept de *Light Robustness*, qui a ensuite été généralisé à des ensembles d'incertitude arbitraires par [84]. En reconnaissant que le concept de *Strict Robstness* est trop conservateur, [81], ont proposé la première approche de robustesse en deux étapes en introduisant leur concept de *Adjustable Robustness*. Lors de l'application de ce concept à plusieurs applications de la planification ferroviaire dans le cadre du projet ARRIVAL (voir [105], [93]), les actions autorisées à ajuster un calendrier ne correspondent pas aux besoins pratiques. Cela les a motivés à intégrer les actions de

récupération dans une planification robuste, ce qui a donné naissance à un autre concept appelé *Recoverable Robustness*. Malheureusement, les solutions robustes avec cette approche sont difficiles à obtenir. La recherche sur l'élaboration d'algorithmes pratiques est toujours en cours. Les exemples récents sont une approche basée sur la génération de colonnes pour les problèmes de sacs à dos et les problèmes de chemins les plus courts avec une demande incertaine [Bouman, 2011], une approche utilisant la décomposition de Bender pour la planification du matériel roulant ferroviaire [107], et l'idée de remplacer l'algorithme de récupération par une métrique [88] et [96-97].

Conclusion

Dans ce chapitre, nous avons introduit les fondements théoriques de notre travail qui s'appliqueront dans les chapitres suivants. Le chapitre a débuté par un état de l'art sur la modélisation du trafic. Un bref historique sur la théorie des graphes et les algorithmes pour la recherche d'itinéraire a été ensuite exposé. Le chapitre termine par un rappel des problèmes d'optimisation robuste et des différentes approches existantes pour qualifier une solution de robuste.

Chapter 2

Robust guidance

Summary

1. Introduction

In a given network, the most used way to plan a route consists to determine shortest paths for origin/destination pairs. Finding such optimal paths is an important problem in the network theory, and has vast applications in many scientific and engineering fields, particularly in transportation engineering. Shortest path algorithms are widely available in the literature. However, in a stochastic framework, different performance criteria may lead to different optimal routing strategies. In case of having perfect information on the link travel-times, a shortest path problem may be solved for example by Dijkstra algorithm [110], or more generally by value iteration [111]. If we have a known dependence on the link travel-times, an optimal strategy may be a sequence of links, that is, a path, because no additional information can be acquired therein. In case of stochastic shortest path problem, link travel-times are random variables, and available information is given as probability distributions. This kind of problems is widely studied in the literature, and is formulated by means of different objective functions; we can cite for example, -minimizing the expected travel-time explored in [112-113-114-115-116-117-118-119], -maximizing the expected utility, as investigated in [120-121], -maximizing the probability of arriving at the destination on time [122-123-124-45-126-127-128], -minimizing the expected travel-time while ensuring a pre-specified probability of arriving by a given deadline [127-129-130-131-132], or minimizing the sum of expected travel-time [45-127-133-134]. The stochastic shortest path problem solutions are either a priori or adaptive solutions. In a priori case, the path is solved based on the information available at the starting time only. In adaptive solutions, the solution consists in a routing policy that adapts the path according to information obtained during the travel.

Among the several formulations, the simple formulation of the stochastic shortest path problem is the Least Expected Time (LET). This formulation is based on the minimization of the expected travel-time. Determining a LET path when random link travel-times are independently distributed and constant over time, is trivial. Fortunately, LET problems have been studied broadly and have been extended to other cases. Hall [61] studied the LET problem in a stochastic time-dependent network and considered that the travel-times on outgoing links from a node are conditional on the arrival time to the node. The authors [112-114-115] addressed the LET problem by including real-time information about the

travel-time. In [112], Fu has shown that the relationship between route planning and information can be divided into three schemes. The first one is the non-adaptive routing rule which is made at the beginning of the task. In this category, a complete fixed path is identified on the basis of a priori or historical travel-time information. In general, this complete path is computed before a trip starts and no re-route adaptive diversion is taken into account, either because of lack of real-time information, or of an unavailability of route guidance system. The second category is the open-loop adaptive routing. It is similar to the non-adaptive routing in the sense that a complete path must be computed. In this case, the remaining path by non-adaptive routing is replanted every time new data become available. The third class is the closed-loop adaptive routing. Contrary to the open-loop adaptive routing, an optimal adaptive routing system should be one of closed-loop adaptive routing, because this category takes into account the future availability of travel-time information, and thus, specifies at each node how to react to the information obtained.

The LET problem is not affected by the travel time variance. In other words, LET solutions are risk-neutral and they do not depend on the uncertainties of the link travel-time. A possible method to formulate attitude towards risk is to use a utility function [136-137]. More details on utility functions in stochastic shortest path problems can be found in [119-120-121]. There exist heuristic methods to study risk aversion also; see for example [127-133]. However, most of these formulas result in extremely complex problems in relation to the objective function that does not accumulate over the links. In this case, dynamic programming cannot be applied [120].

In the stochastic path planning problem, Frank [45] introduced the objective of maximizing the probability that the destination travel-time is less than a given deadline time. Since the work of Frank, the on-time arrival problems have been explored by many researchers [122-123-45-126-127-128]. In [122], Fan et.al defined optimality as the maximization of the travel-time reliability and proposed an adaptive optimal path algorithm to solve this optimization problem. This is the Stochastic On-Time Arrival (SOTA) problem. In this section, a brief discussion on the concept of the SOTA problem is given including the two existing variants, for the convenience of readers and the continuity of our discussion in the next sections. In the SOTA problem, one seeks to maximize the probability of a time arrival at a given destination, departing from a given origin, with a given travel-time budget. The travel-time across every link is a random variable with some

arbitrary probability distribution. Two primary variants can be distinguished for the SOTA problem. The first one consists in finding the most reliable fixed path to the destination. This first variant is designed as the shortest path problem on-time arrival reliability, or also the path-based SOTA problem as explored in [130]. The second variant is referred as the policy-based SOTA problem, which consisting in calculating a routing policy such as the selection of the next node at each intersection depending on the current state (remaining time budget). The policy-based SOTA problem is solved in discrete-time. In [126], the authors presented a discrete SOTA algorithm that ensures finite convergence and runs very well in polynomial time. Solving the SOTA problem in discrete-time consist in computing product convolution of arbitrary distributions. The computation of the policy requires a subsequent maximization step. Unfortunately, this step mixes distributions and prevents finding an analytical solution in continuous-time. A successive approximation method is proposed in [123] for solving the policy-based SOTA problem. This algorithm is improved in [128-138] to a dynamic programming algorithm, and the speed-up techniques including zero-delay convolution [139] have been explored to solve the problem in pseudo-polynomial-time. The authors of [140] have shown how pre-processing methods can be used to further reduce the computation time of the SOTA problem. Unfortunately, the structure of the SOTA problem formulation limits the types of pre-processing methods that can be used for this problem, and prevents massive running times reductions as possible in the deterministic case.. Recently, [141] presented a novel approach to reduce the immense computational effort of stochastic routing based on existing techniques for alternative routes.

In this chapter an extension of the SOTA algorithm reported in [128] is presented. This extension permits to include robustness in the criteria of the routing optimization. Instead of considering a unique objective of maximizing the travel-time reliability, we also optimize the robustness of the selected paths and routing policies against link failures. The remainder of this chapter is organized as follows.

In section 2, we give a short review of the SOTA problem in cases of independent link travel-times and correlated link travel-times.

Section 3 contains 2 main subsections. In sub-section 3.1, we consider the problem of robust guidance in case of independent link travel-times. A guidance strategy is said to be

robust here, if it minimizes the deterioration of its maximum value (in terms of the mean travel time and its reliability) calculated at the origin, against eventual reconfigurations of the network due to link failures (accidents, works, etc.). We begin this sub-section by extending the stochastic on time arrival (SOTA) algorithm proposed in [128] by incorporating the alternative routing choices at nodes considering the possible failures of links in routes connecting to the destination. The proposed algorithm developed differs from the existing SOTA algorithm family in that, in the objective function we use a weighted average of the cumulative probabilities to arrive at the destination in a given time budget instead of the maximum of probabilities. We provide also, a new definition of robust-optimality and we explain how can the user measure the robustness and investigate the solution quality. In this section, we provide also, the discretization scheme of the model for the case of independent link travel times and we analyze the algorithm complexity of the model. In sub-section 3.2, we provide the robust guidance model in case of correlated link travel-times. As in [128] we present a simple extension to our formulation that considers the correlation between a link and the upstream neighbors (ie successive links) via which the link is reached. We are not interested in other correlations (ie correlations between non-consecutive links) because it makes the problem more complex.

In section 4, we propose a parameter optimization of the robust routing algorithm, and we show how to choose the robustness parameter in function of the desired travel-time reliability and the desired travel-time budget.

In section 5, we present an extension of our model in case of time varying link travel-times.

In section 6, we conduct numerical experiments and we compare several scenarios on a well-known road network (the Sioux Falls network) in a static domain, to consider the sensitivity of the proposed approach to changes in the key parameters.

In section 7, some conclusions are drawn.

2. Stochastic On Time Arrival (SOTA) problem

In this section, we briefly summarize the original SOTA problem for the convenience of readers and for the continuity of our discussion in the next sections. The details of the approach summarized here are available for example in [123].

2.1. Formulation of the SOTA problem with uncorrelated link travel-times

A road network is represented by a graph including arcs and nodes which correspond respectively to the links and junctions of the road network. We denote the graph representing the road network by $G(N, A)$, where N is the set of nodes, with $|N| = n$, and A is the set of arcs, with $|A| = a$. The set of successor and predecessor nodes are denoted by $\Gamma^{+1}(i) = \{j|(i,j) \in A\}$ and $\Gamma^{-1}(i) = \{k|(k,i) \in A\}$ respectively. The travel times through the links of the network are assumed here to be stochastic. Probability distributions of the travel times are then associated to the links of the graph. The SOTA problem consists in finding the best routing strategy from any starting node $i, (i = 1,2, \dots \dots, n)$, that maximises the probability of arriving to a given destination node, denoted d, within a time budget t.

Given a node $i \in N$ and a time budget t, $u_i(t)$ denotes the maximum probability of arriving to the destination node d parting from node i, within a time budget t, and under the optimal policy. The latter, denoted by $s_i(t)$ is given by the optimal subsequent node. $p_{ij}(.)$ denotes the probability distribution function (pdf) of the travel time on link (i,j). It is assumed to be known and can for example be obtained using historical data or real-time traffic information. The maximum probability $u_i(t)$ and the optimal successor node $s_i(t)$ are written as follows.

$$u_i(t) = \max_{j \in \Gamma^{+1}(i)} \int_0^t p_{ij}(w) u_j(t-w) dw, \quad \forall i \in N \backslash \{d\}, j \in \Gamma^{+1}(i), 0 \leq t \leq T \tag{1}$$

$$u_d(t) = 1, \quad 0 \leq t \leq T \tag{2}$$

$$s_i(t) = \arg\max_{j \in \Gamma^{+1}(i)} \int_0^t p_{ij}(w) u_j(t-w) dw, \quad \forall i \in N \backslash \{d\}, j \in \Gamma^{+1}(i), 0 \leq t \leq T \tag{3}$$

Where T is the maximum time budget.

Formula (1) expresses the fact that a traveler being at node i, having a time budget t, and knowing $u_p(t), \forall p \in \{1,2,\dots,n\}, \forall s \in [0,t]$, should go through the link (i,j) that maximizes the probability of arriving within time t to the destination node d, with respect to all the possible successor nodes j of i. Formula (2) tells simply that parting from node d, the maximum probability of arriving to the same node d, within any time budget is 1. Formula (3) tells that the optimal successor node for the traveller being at node i, is given as the argument of the maximum taken in (1).

To solve the system of nonlinear equations (1)-(2), the Picard method of successive approximation is one possible approach proposed in [123]. This fixed point method starts with initial approximations of the solution and refines these approximations by successive iterations. Then, the iterative relationships for successive approximations are given as follows.

$$u_i^{iter+1}(t) = \max_{j \in \Gamma^{+1}(i)} \int_0^t p_{ij}(w) u_j^{iter}(t-w)dw, \tag{4}$$

$$\forall i \in N\backslash\{d\}, j \in \Gamma^{+1}(i), 0 \le t \le T$$

$$u_d^{iter+1}(t) = 1, \quad 0 \le t \le T \tag{5}$$

Where the superscript $iter$ is the iteration index. The function $u_i^{iter}(t)$ represents the probability of reaching the destination node d if optimal choices are made.

Given initial approximations, the pseudo-code for successive approximation algorithm is given by Algorithm 1 below.

Algorithm 1 : Successive approximation algorithm (Fan and Nie, 2006)

Step 0. Initialization

$iter = 0$ (iteration index)

$u_i^{iter}(t) = 0, \quad \forall i \in N\backslash\{d\}, 0 \le t \le T$

$u_d^{iter}(t) = 1, \quad 0 \le t \le T$

Step 1. Update

$$\text{iter} = \text{iter} + 1$$

$$u_d^{iter}(t) = 1, \quad 0 \le t \le T$$

$$u_i^{iter+1}(t) = \max_{j \in \Gamma^{+1}(i)} \int_0^t p_{ij}(w) u_j^{iter}(t-w)dw$$

$$\forall i \in N \backslash \{d\}, j \in \Gamma^{+1}(i), 0 \le t \le T$$

Step 2. Convergence test

If $\forall (i,t) \in N \times [0,T], \max_{i,t} \left| u_i^{iter+1}(t) - u_i^{iter}(t) \right| = 0,$ then stop;

Otherwise go to step 1

In an acyclic network, at each iteration iter, $u_{ki}^{iter}(t)$ gives the probability of reaching the destination node d from origin node i within a time budget $t \in [0,T]$, using a path that contains no more than iter links, and under the optimal policy. The approximation error monotonically decreases with iter and the solution eventually reaches an optimal value when iter is equal to the number of links in the optimal path. However, in a network that contains cycles, as is the case with all road networks, there is no finite bound on the maximum number of iterations required for the algorithm to converge [123]. This is due to the fact that the optimal solution can contain loops. To solve the problem raised by the unbounded convergence of Algorithm 1, A discrete SOTA algorithm is presented in [126] which ensures finite convergence and, more important, runs in polynomial time. The authors of [126] showed that given a finite time budget T, the solution of the SOTA algorithm can be found by scanning a discrete probability expansion network (DPEN), which is constructed from the original stochastic network (see section Finite Convergence Algorithm in [126] for more details).

The authors of [128] show how the SOTA problem can be solved exactly in a finite number of steps, even in cyclic networks when there is a uniform strictly positive minimum link travel-time. As in [123], this algorithm requires computing a continuous-time convolution product, which is one of the main computational challenges of the method. In general, this convolution cannot be solved analytically when routing in general networks, and therefore, a discrete approximation scheme is required. The solution presented in [128] allows for batch computation of the convolution product and thus more efficient

computation methods than the standard (brute force) discrete time approximation algorithm used in [129]. In that formulation, the order in which the nodes of the graph are considered when solving the underlying dynamic program greatly impacts the computation time of the proposed solution. Therefore, an optimal ordering algorithm that determines the best order in which to solve the dynamic program is also proposed in [128].

The model (1)-(3) present the simple case of the SOTA problem where the travel-times on the links of the network are uncorrelated. Different variants of the SOTA problem with correlated link travel times are proposed in the literature, see for example [129-128]. We expose below the model with travel-time correlations proposed in [128].

2.2. Formulation of the SOTA problem with correlated link travel-times

In [128], the authors presented a simple extension of the SOTA model that considers correlation between the travel times of any two consecutive links. Let us denote by $u_{ki}(t,y)$ the maximum probability for a user to arrive to destination node d within a time t, parting from node i, conditioned that the user comes from node k, and that the realized travel time on link (k,i) is y. The maximum probabilities $u_{ki}(t,y)$ satisfy the following equations.

$$u_{ki}(t,y) = \max_{j\in\Gamma^{+1}(i)} \int_0^t p_{ij}(w \mid y)\, u_{ij}(t-w,w)\,dw,$$
$$\forall i \in N\backslash\{d\}, k \in \Gamma^{-1}(i), j \in \Gamma^{+1}(i), 0 \le t \le T, 0 \le y \le T-t \tag{6}$$

$$u_{kd}(t,y) = 1, \forall k \in \Gamma^{-1}(d), 0 \le t \le T, 0 \le y \le T-t. \tag{7}$$

$$s_{ki}(t,y) = \arg\max_{j\in\Gamma^{+1}(i)} \int_0^t p_{ij}(w \mid y)\, u_{ij}(t-w,w)\,dw, \tag{8}$$
$$\forall i \in N\backslash\{d\}, k \in \Gamma^{-1}(i), j \in \Gamma^{+1}(i), 0 \le t \le T, 0 \le y \le T-t$$

where tt_{ij} denotes the travel time on link (i,j), and $p_{ij}(w \mid y)$ is the probability distribution function (pdf) of w, conditioned by y. The pdf $p_{ij}(.)$ is assumed to be known and can be

obtained for example, using historical data or real-time traffic information. $u_{ij}(t-w,w)$ is the maximum probability of arriving to destination node d within time $t-w$, parting from node j, conditioned that the travel time on link (i,j) is w.

2.3. An academic example

Let us consider the network of Figure 8 including 5 nodes and 7 links. We illustrate here the approach routing by applying the SOTA algorithm given by model (1)-(3) to find an optimal path from node 1 to node 5 with a time budget of 25 time units. We derive the probabilities $u_1(t)$ for origin nodes 1 of the network, as well as the associated optimal policies $s_1(t)$.We assume that the travel times on the links of the network are uncorrelated, and follow Gamma probability distributions. The gamma probability distribution is a two-parameter family of continuous probability distributions. There are three different parametizations in common use:

- with a shape parameter k ans a scale parameter θ.
- with a shape parameter $\alpha = k$ and an inverse scale parameter $\beta = \frac{1}{\theta}$ called a rate parameter.
- with a shape parameter k and a mean parameter $\mu = {}^{k}/_{\beta}$.

In each of these three forms, both parameters are positive real numbers.

The corresponding probability density function in the shape α and rate β parameterization that we use in this example is:

$$f(x,\alpha,\beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}$$

where $\Gamma(\alpha)$ is a complete gamma function which is defined by the following integral

$$\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1} e^{-x}\, dx$$

For all the links of the network, we assume an average travel time of 7 time units with a standard deviation of 3 time units, except for link (1,3), for which we assume an average

travel time of 6 time units and a standard deviation of 3 time units. To reach node 5 from node 1, four routes exist: route1 (1-3-5), route2 (1-2-4-5), route3 (1-2-4-3-5) and route 4 (1-2-5). In term of minimum average travel time (i.e. applying the LET routing algorithm), it is easy to check that route 1 is the optimal one.

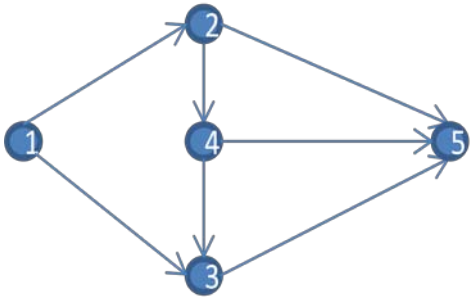


Figure 8. A simple test network

We apply the SOTA algorithm (model (1)-(3)) for this example. We obtain the results given in Table 1. $u_1(t)$ denotes the probability to reach the destination node 5 from origin node 1 within a time budget t, and $s_1(t)$ is the optimal strategy from origin node 1 to destination node 5 within a time budget t.

Note that, we will take the same academic example in the following sections in order to illustrate our robust approach and make a comparison between the two approaches.

Table 3. Optimal solution (probabilities u and policies s) to reach node 5 from node 1

	Time t	15	16	17	18	19	20	21	22	23	24	25
policy	$s_1(t)$	3	3	3	3	3	3	3	3	3	3	3
probability	$u_1(t)$	0,9436	0,9742	0,9909	0,9968	0,9989	0,9990	0,9996	0,9999	1	1	1

We can clearly see in Table 1 that the optimal route is the route passing through node 3 i.e. route 1, for all the considered time budgets. The SOTA algorithm permits the maximum cumulative probability distribution computation of the time arrival toward a given destination in the network. These distributions allow to user selection of the most reliable origin-destination paths under given time cost. However, the question that arises

here is the following. If the link (3.5) fails once the user arrives at node 3, what will be the options for the user? The answer is clear, if link (3,5) fails once the user arrives to node 3, then he will not have any other alternative route to reach the destination.

Then, by this illustrative example, we can say that the optimal strategy given by the SOTA algorithm is not robust because it does not propose any alternative route in case of link failures. To solve this problem, we propose an extension of the SOTA algorithm by introducing robustness against link and path failures for the selection of guidance strategy. The new algorithm takes into account the reliability of itinerary travel times, since it is based on a SOTA algorithm. In addition, it takes into account itinerary robustness, by favoring itineraries with possible and reliable alternative diversions (in case of link failures), with respect to itineraries without or with less reliable alternatives.

In the next sections, based on the SOTA formulation described above, an extension of the routing algorithm is performed including a robustness criterion for the routing strategy, against link and path failures.

3. Robust guidance

In this section we base on the routing model presented in [123] (model (1)-(3) above) where from the probability distributions of travel times through the links of the network, users evaluate their maximum probability to reach their destination in given time budgets, and using different possible routes.

We propose here an extension of this approach in order to take into account the existence and the performance of alternative detours of the selected paths, in the calculus of the guidance strategy. We take into account the fact that one or many links of the selected optimal path may fail during the travel. We then consider that users may be sensitive to path changing. That is to say that they may prefer paths with efficient alternative detours, with respect to paths without, or with less efficient detours, even with a loss in the average travel time, and/or in its reliability. In order to take into account such behaviors, we propose a model that includes the existence as well as the performance of detours for selected paths, in the calculus of the travel time reliability (i.e. the probability of reaching a destination node). This new way of calculating travel time reliability guarantees a kind of

robustness of the guidance strategies. That is to say that the travel time reliability associated to the obtained optimal guidance strategy is not likely to change, however associated adaptive paths change during the travel. The variation of the travel time reliability, with respect to a network structure changing, is thus improved. For that, we propose to calculate for each node i the probability $u_i(t)$ to reach the destination node d, where we take into account the case where the selected path fails before the users who selected it reach the destination node d; for which case, alternative neighboring paths are used. $u_i(t)$ denotes, as above, the probability to reach the destination node d, parting from node i, in a time-budget t. The mathematical definition of $u_i(t)$ is different from the one of the model (1)-(3). It is given in the following sub-sections, in both cases of uncorrelated and correlated link travel-times.

3.1. Robust guidance with uncorrelated link travel-times

In a stochastic framework, robustness is generally defined as the probability that the system of interest has the ability to resist to changes without adapting its initial stable configuration. Recently, path robustness in transport networks has emerged as an important topic. It has attracted many researchers to develop various indicators to assess the path robustness in road networks as explained in chapter 1. In this thesis, we provide a new idea of incorporating robustness with route choice using the case that link failure may occur. We consider here that a routing strategy is robust, if it minimizes the deterioration of its maximum value calculated before the depart at the origin, against eventual reconfigurations of the network that may due to accidents, works, etc. The value of a strategy is maximized with respect to the average travel-time and its reliability associated to the routing strategy. Consequently, a robust routing strategy would resist network reconfiguration due to link failures. We notice here that the connectivity of the network is important for the robustness of routing strategies. Indeed, a path passing through nodes having many successor nodes would be more robust than a path passing through nodes having one or small number of successor nodes; see definition 1 for more details.

In this sub-section we give the robust model which does not take into account correlations between the travel times through the links. First, we begin by presenting the different relations in the continuous time. In a second step, we expose the discrete formula of our robust model.

3.1.1. Continuous time formulation

We consider a directed network $G(N, A)$, where N is the set of nodes, with $|N| = n$, and A is the set of arcs, with $|A| = a$. The set of successor and predecessor nodes are denoted by $\Gamma^{+1}(i) = \{j|(i,j) \in A\}$ and $\Gamma^{-1}(i) = \{k|(k,i) \in A\}$ respectively. The weights of each link $(i,j) \in A$ are a random variables with probability density functions $p_{ij(.)}$ that represent the travel-time on links (i,j). The link travel-time distributions are assumed to be uncorrelated.

The optimal routing policy for the robust guidance problem can be expressed as follows.

3.1.1.1. The probabilities $u_i(t)$

We introduce here a modification on the model (1)-(3) in order to take into account the existence and the performance of alternative paths in the calculated optimal routing strategy. In our model, the word "optimal" refers to the robustness of the strategy. The idea here is to replace the maximum operator in equation (1) by a weighted mean over a chosen number of successor nodes. Instead of calculating $u_i(t)$ basing on the successor node giving the optimum value of $u_i(t)$, we propose here to consider also other successor nodes of i, and we rather calculate $u_i(t)$ basing on a weighted mean over a number of successor nodes of i. Let us consider the following notation.

$$A_{ij}(t) = \int_0^t p_{ij}(w)u_j(t-w)dw, \qquad \forall i \in N\backslash\{d\}, j \in \Gamma^{+1}(i), \forall 0 \le t \le T \tag{9}$$

We denote by $A_i(t)$ the vector $A_i(t) = \left(A_{i1}(t), A_{i2}(t), \dots, A_{in_i}(t)\right)$. Where n_i is the number of successor nodes of node i in the graph. We then define N maps $S_i, i = 1,2, \dots, N$ as follows.

$$\begin{aligned} S_i : \mathbb{R}^{n_i} &\to \mathbb{R}^{n_i} \\ A_i(t) &\mapsto S_i\left(A_i(t)\right) \end{aligned} \qquad \forall i \in \{1,2, \dots, N\} \tag{10}$$

Where $S_i(A_i(t))$ is the vector whose components are the same as those of $A_i(t)$ but sorted in a decreasing order. $S_{ij}(A_i(t))$ denotes here the j^{th} component $\left(S_i(A_i(t))\right)_j$ of vector $S_i(A_i(t))$.

We then rewrite the probability for a user to reach the destination node d from node i in a time budget t, as follows.

$$u_i(t) = \sum_{p=1}^{m} \psi_p S_{ip}(A_i(t)), \quad \forall i \neq d, 0 \leq t \leq T, \tag{11}$$

$$u_d(t) = 1, \quad 0 \leq t \leq T, \tag{12}$$

where m is a parameter giving the number of successor nodes taken into account in the sum of equation (11), ψ_p are non increasing weighting coefficients satisfying

$$\psi_p \geq 0, \forall p \in \{1,2,\dots,m\}, \qquad \sum_{p=1}^{m} \psi_p = 1, \qquad \text{and} \quad \psi_1 \geq \psi_2 \geq \cdots \geq \psi_m.$$

The following two cases are distinguished.

- **Case 1.** $m \leq |\Gamma^{+1}(i)|$, in which case, no more than the number of successors of i are considered in the sum of Eq (11).
- **Case 2.** $m > |\Gamma^{+1}(i)|$, in which case, we have $S_{ip}(A_i(t)) = 0$ for $p > |\Gamma^{+1}(i)|$. and $\sum_{j\in\Gamma^{+1}(i)} \psi_j \leq \sum_{p=1}^{m} \psi_p = 1$.

In case 2 above, nodes i with small numbers of successors are penalized; they get low values $u_i(t)$. Therefore, paths passing through these nodes i.e. paths with small number of alternatives or detours shall have low probabilities to be selected as optimal paths. One way to choose m can be to take the maximum over the cardinals of the sets $\Gamma^{+1}(i)$ of successors of all the nodes of the network.

$$m = \max_{i\in N} |\Gamma^{+1}(i)|$$

where $|.|$ denotes the cardinal of a set. In the examples we give below, we take $m = 2$.

In order that equation (11) will have a meaning, ψ_p have to be chosen such that $\psi_1 \geq \psi_2 \geq \cdots \geq \psi_m$. That is to say that ψ_p decrease as $S_{ip}(A_i(t))$ decrease with respect to p. This dependence of ψ_p on $A_{ip}(t)$ makes the model non-trivial. Indeed, instead of taking the maximum over $A_{ip}(t)$, with respect to successors p of i, as in equation (1), we take a weighted mean in equation (11), where the weights are in the same order as the one of the quantities $S_{ip}(A_i(t))$. Therefore, we need to first sort the quantities $A_{ip}(t)$, before applying the mean operator. So the model (11) needs more operations than the model (1). Finally, let us notice that if $m = 1$, or if $m > 1$ and $\psi_p = 0, \forall p \geq 2$, then equations (1)-(2) coincides with equations (11)-(12). Therefore, the model (11)-(12) extends the model (1)-(2).

3.1.1.2. The successor nodes

In the calculus of $u_i(t)$ by equation (11), instead of maximizing the quantities $A_{ij}(t)$, we proposed to take a weighted mean of theses quantities, with weights $\psi_p, p = 1,2, \ldots . m$. The optimal guidance strategy is then determined by the sequence of successor nodes $s_i(t)$ as given by equation (13) below.

$$s_i(t) = \arg \max_{j \in \Gamma^{+1}(i)} \left(A_{ij}(t)\right), \qquad i \in N \tag{13}$$

$s_i(t)$ denotes here the optimal successor node of node i for a user to reach the destination node d. By taking a mean in equation (11) rather than the maximum (as in equation (1)), we take into account the existence and the performance of alternative deviations at every intermediate node from i to d. We notice here that although formula (13) resembles to formula (3), the resulted successor nodes from the two formulas are not necessarily the same, since the quantities $u_i(t)$ in equation (1) and $u_i(t)$ in equation (11) are calculated differently. Table 4 below gives a continuous time scheme for the model (11)-(13).

Table 4. Continuous time formulation of the robust guidance algorithm

---

Step 0. Initialization

- Fix m (maximum number of successor nodes to take into account). For example, $m = \max_{i \in N} |\Gamma^{+1}(i)|$.
- Fix $\psi_p, p = 1, \dots, m$ such that $\sum_{p=1}^{m} \psi_p = 1$ and $\psi_1 \geq \psi_2 \geq \cdots \geq \psi_m$.
- Fix the time unit $\delta = tt_{min} - \varepsilon$, where tt_{min} is the minimum realizable travel-time across the network, and ε is a short time. We then have $\forall i, j \in N, \forall w \in [0, \delta],\ p_{ij}(w) = 0$.
- Fix $T = L\delta$ The total time budget T is a multiple of δ.
- Fix $iter = 0$ (iteration index).
- $u_i^{iter}(t) = 0, \forall i \in N \backslash \{d\}, 0 \leq t \leq T$
- $u_d^{iter}(t) = 1,\ \ 0 \leq t \leq T$

Step 1. Update

For $iter = 1,2, \dots \dots . L$, with $L = T/\delta$,

- $\tau^{iter} = iter * \delta$
- $u_d^{iter}(t) = 1, \quad 0 \leq t \leq L\delta$.
- $u_i^{iter}(t) = u_i^{iter-1}(t), \forall i \in N, i \neq d, t \in [0, \tau^{iter} - \delta]$.
- Calculate $A_{ip}^{iter-1}(t)$, for every i, p, t as indicated below.
- For every i, t, apply S_i to sort $A_i^{iter-1}(t)$ with in a decreasing order, and obtain $S_i^{iter-1}(t)$
- $u_i^{iter}(t) = \sum_{p=1}^{m} \psi_p S_{ip}^{iter-1}(A_i(t))$, $\forall i \in N \backslash \{d\}, j \in \Gamma^{+1}(i), t \in (\tau^{iter} - \delta, \tau^{iter}]$
- $s_i^{iter}(t) = \arg\max_{j \in \Gamma^{+1}(i)} \left(A_{ij}^{iter-1}(t)\right),\ \forall i \in N \backslash \{d\}, j \in \Gamma^{+1}(i), t \in (\tau^{iter} - \delta, \tau^{iter}]$,

---

Definition 1 : Robust-optimality.

We say that u is *robust-optimal*, with respect to parameters m, $\psi_p, p = 1,2, \dots, m$ and time-budget T, if it is the unique possible routing strategy for the time budget T, or if u is robust-optimal with respect to parameters m, $\psi_p, p = 1,2, \dots, m$ and time-budget $T - \delta$, and u satisfies

$$u_i(t) = \sum_{p=1}^{m} \psi_p S_{ip}\big(A_i(t)\big), \quad \forall i \in N\backslash\{d\}, j \in \Gamma^{+1}(i), t \in (t-\delta, T],$$

Robustness of a routing strategy has a price in term of travel time reliability and a price in term of travel time budget, as explained in section 3.4.1.

Proposition 1. For a total time budget $T = \delta L$, the solution obtained at iteration $iter = \delta L$ of Step 1 is robust-optimal.

Proof : By induction on L.

- For $L = 0$, the total time budget is zero, and thus only Step 0 is performed by the algorithm, which consequently terminates with a u satisfying $u_i(0) = 0, \forall i \in N\backslash\{d\}$ and $u_d(0) = 1$. This solution is robust-optimal since it is the unique possible solution for the time budget zero.

- Assume that for $T = \delta L$, the solution given at iteration $k = \delta L$ of Step 1 of the algorithm of Table 2 is robust-optimal. Then with a time budget $T = (L+1)\delta$, the algorithm keeps the same robust-optimal solution $u_i^{L+1}(t) = u_i^L(t), \forall i \in N$ for every time $t \in [0, \delta L]$. For times $t \in (\delta L, \delta(L+1)]$, $u_i^{L+1}(t)$ is robust-optimal by definition of the robust-optimality.

The meaning of Proposition 1 is that the continuous scheme of Table 4 can be solved in a single update i.e. without resorting to value iteration. It is similar to the one introduced in [21]. The unique difference lies in the calculus of $u_i(t)$, using here a weighted mean instead of the maximum.

3.1.2. Discretization Scheme of the robust guidance algorithm

To numerically approximate $u_i(t)$, we descretize the interval [0, T] into $L = T/\Delta t$ time steps of length Δt. As in [128], the discretization length is supposed to satisfy $\Delta t \leq \delta$. To simplify the notations, we assume that $T = n.\Delta t$. The discretized algorithm is then given in Table 5 below.

The probability distributions of the link travel times are discrete here. Therefore, the calculus of $A_{ij}(x)$ is done as follows (replacing (9)).

$$A_{ij}(x) = \sum_{h=0}^{x} p_{ij}(tt_{ij} = h)u_j(x-h)$$

tt_{ij} : the travel-time on link (i,j).

Table 5. Discrete time formulation of the robust guidance algorithm

---

Step 0. Initialization

- Fix m (maximum number of successor nodes to take into account). For example, $m = \max_{i\in N}|\Gamma^{+1}(i)|$.
- Fix $\psi_p, p = 1, \dots, m$ such that $\sum_{p=1}^{m}\psi_p = 1$ and $\psi_1 \geq \psi_2 \geq \cdots \geq \psi_m$.
- Fix the time unit $\Delta t \leq \delta$. We then have $\forall i, j \in N, \forall w \in [0, \Delta t],\ p_{ij}(tt_{ij} = w) = 0$.
- Fix $T = L\delta$ The total time budget T is a multiple of δ.
- $u_i^{iter}(x) = 0, \forall i \in N\backslash\{d\}, x \in \mathbb{N}, 0 \leq x \leq \frac{T}{\Delta t}$,
- $u_d^{iter}(x) = 1, x \in \mathbb{N}, 0 \leq x \leq \frac{T}{\Delta t}$

Step 1. Update

For $iter = 1,2, \dots \dots L$, with $L = T/\delta$,

- $\tau^{iter} = iter * \delta$
- $u_d^{iter}(x) = 1, x \in \mathbb{N}, 0 \leq x \leq \frac{T}{\Delta t}$.
- $u_i^{iter}(x) = u_i^{iter-1}(x), \forall i \in N, i \neq d, x \in \left[0, \frac{\tau^{iter}-\delta}{\Delta t}\right],\ x \in \mathbb{N}$.
- Calculate $A_{ip}^{iter-1}(x)$, for every i, p, x as indicated in Section 3.1 above.
- For every i, x, apply S_i to sort $A_i^{iter-1}(x)$ with in a decreasing order, and obtain $S_i^{iter-1}(x)$
- $u_i^{iter}(x) = \sum_{p=1}^{m}\psi_p S_{ip}^{iter-1}(A_i(x))$,

  $$\forall i \in N\backslash\{d\}, j \in \Gamma^{+1}(i), x \in \left[\frac{\tau^{iter}-\delta}{\Delta t}+1, \frac{\tau^{iter}}{\Delta t}\right], x \in \mathbb{N}.$$

- $s_i^{iter}(x) = \arg\max_{j\in\Gamma^{+1}(i)}\left(A_{ij}^{iter-1}(x)\right)$,

  $\forall i \in N\backslash\{d\}, j \in \Gamma^{+1}(i), x \in \left[\frac{\tau^{iter}-\delta}{\Delta t}+1, \frac{\tau^{iter}}{\Delta t}\right], x \in \mathbb{N}$.

3.1.2.1. Complexity analysis

Proposition 2. The complexity of the Algorithm of Table 3 is $O(a(T/\Delta t)^2 + (T/\Delta t)a \log a)$.

Proof.

The functions $A_{ij}(.)$ and $p_{ij}(.)$ are vectors of length L. Each link travel-time distribution in the network is of length $T/\Delta t$, and the discretized probability mass function is computed in time $O(T/\Delta t)$ for each link. As there are a links, then the total time is $O(a\,T/\Delta t)$. In step 0, there are $iter$ vectors to initialize for each node, and each vector is of length $T/\Delta t$. Then the initialization is done in time $O(iter\,T/\Delta t)$. In step 1, the algorithm progressively calculates:

- The sum of the convolution product $A_{ij}(x)$ from $x = 0$ to $x = T/\Delta t$. Then the time complexity of the summation for a link is $O(a(T/\Delta t)^2)$.
- The quantities $A_{ij}(x)$ are sorted for each x, then the complexity of this sorting is $O((T/\Delta t)\, a \log a)$.
- The sum on the weighting coefficients from $p = 1$ to $p = m$. Then the time complexity of this summation is $O(a\,T/\Delta t)$.

Therefore, the total complexity to this algorithm is $O(a(T/\Delta t)^2 + (T/\Delta t)\, a \log a)$.

3.1.3. An academic example

Let us consider the same network of Figure 1 of sub-section 2.3. The network includes 5 nodes and 7 links. We illustrate here our approach by applying the algorithm given in Table 2 to find a robust-optimal path from node 1 to node 5, and we compare the results with those obtained in section 2.3. We assume that the travel times on the links of the network are uncorrelated, and follow Gamma probability distributions. For all the links of the network, we assume an average travel time of 7 time units with a standard deviation of

3 time units, except for link (1,3), for which we assume an average travel time of 6 time units and a standard deviation of 3 time units. To reach node 5 from node 1, four routes exist: route1 (1-3-5), route2 (1-2-4-5), route3 (1-2-4-3-5) and route 4 (1-2-5). In term of minimum average travel time (i.e. applying the LET routing algorithm), it is easy to check then route 1 is the optimal one.

We apply our algorithm (model (11)-(13)) for this example. We fix $m = 2$ (the maximum number of successor nodes in the network). We then have two weighting parameters ψ_1 and ψ_2, with $\psi_1 + \psi_2 = 1$. To simplify, we denote $\psi = \psi_1$and then $\psi_2 = 1 - \psi$. We vary ψ in $(1/2, 1]$. The results are given in Table 3 below.

Table 3. Robust-optimal solutions (probabilities u and policies s) to reach node 5 from node 1 for different value of $\boldsymbol{\psi}$.

	Time t	15	16	17	18	19	20	21	22	23	24	25
$\psi = 1$	$s_1(t)$	3	3	3	3	3	3	3	3	3	3	3
	$u_1(t)$	0,9436	0,9742	0,9909	0,9968	0,9989	0,9990	0,9996	0,9999	1	1	1
$\psi = 0.9$	$s_1(t)$	3	2	2	2	2	2	2	2	2	2	2
	$u_1(t)$	0,8442	0,8772	0,9038	0,9283	0,9457	0,9587	0,9683	0,9751	0,9797	0,9828	0,9849
$\psi = 0.8$	$s_1(t)$	2	2	2	2	2	2	2	2	2	2	2
	$u_1(t)$	0,7472	0,7817	0,8208	0,8503	0,8744	0,8941	0,9096	0,9213	0,9300	0,9365	0,9415
$\psi = 0.7$	$s_1(t)$	2	2	2	2	2	2	2	2	2	2	2
	$u_1(t)$	0,6522	0,6921	0,7294	0,7600	0,7865	0,8092	0,8277	0,8423	0,8540	0,8634	0,8711

$\psi = 1$ corresponds to the model (1-3) and $\psi < 1$ corresponds to the model (11-13). We can clearly see in Table 3 that probabilities $u_1(t)$ are decreasing with ψ, for a fixed time-budget. For $\psi = 1$, the optimal route is the route passing through node 3 i.e. route 1, for all the considered time budgets. For $\psi < 1$, the robust-optimal route changes with the time budget. For example, for $\psi = 0.9$, the robust-optimal route is route1 (the same as in the

case $\psi = 1$) for time budgets less than 16 time units, while for time budgets bigger than or equal to 16 time units, the robust-optimal policy changes and node 2 becomes the robust-optimal successor node of node 1. The robust-optimal route in this case is route 4. The selection of route 4 as the robust-optimal route rather than route 1 in case $\psi < 1$ is justified by the fact that, node 2 offers two routing options (go to node 5, or go to node 4). More options offer more alternatives in adaptive routing. Indeed, for a user who has chosen route 4, if link (2,5) fails once the user arrives to node 2, then he will still have two other alternative routes to reach the destination node 5: route 2 and route 3 by link (2,4). However, if the user chooses route 1, then if link (3,5) fails once the user arrives to node 3, then he will not have any other alternative route to reach the destination. Therefore, route 4 passing through node 2 is more robust than route 1 passing through node 3, because node 2 has more successor nodes than node 3. The robust guidance algorithm favors routes passing by nodes with alternative detours. From these results we can conclude that:

- If a user prefers to maximize the travel time reliability of his routing strategy, without taking into account its robustness against link and path failures, then he should select route 1 because it is the one maximizing the probability of reaching the destination node 5 in the considered time budget.
- If the user seeks a guarantee in terms of robustness and if he accepts to lose in terms of travel time budget and/or travel time reliability, then he should select route 4 because it is the one giving more alternative detours in case of link failure.

3.1.4. Price of robust-optimality

Robustness of a routing strategy has a price in term of travel time reliability, in the sense that a user with a fixed time budget can improve the robustness of his routing strategy if he accepts to lose travel time reliability. In the other side, robustness of a routing strategy has a price in term of time budget, in the sense that a user who requires a fixed level of travel time reliability can improve the robustness of his routing strategy if he accepts to extend his time budget. In order to illustrate this concept of price of robust-optimality, let us back to the academic example given above. From Table 3, if a user likes to reach the destination node within a time budget of 16 time units, then he gets a travel time reliability of 0.9742 by passing by a non robust route (route 1 with $\psi = 1$), and a

travel time reliability of 0.8772 by passing by a more robust route (route 4 with $\psi = 0.9$). Then, the value $0.097 = 0.9742 - 0.8772$ can be interpreted as the price of robustness (of passing from a robustness level corresponding to $\psi = 1$ to a robustness level corresponding to $\psi = 0.9$) to pay in term of travel time reliability; see Figure 2. On the other side, if the user likes to reach the destination node with a travel time reliability of at least 0.9742, he can select a non robust route (route 1 with $\psi = 1$) with a time budget of 16 time units, or a more robust route (route 4 with $\psi = 0.9$) with a time budget of 22 time units (assuring a travel time reliability of 0.9751). Then, the value $6 = 22 - 16$ can be interpreted as the price of robustness (of passing from a robustness corresponding to $\psi = 1$ to a robustness level corresponding to $\psi = 0.9$) to pay in term of travel time budge; see Figure 9.

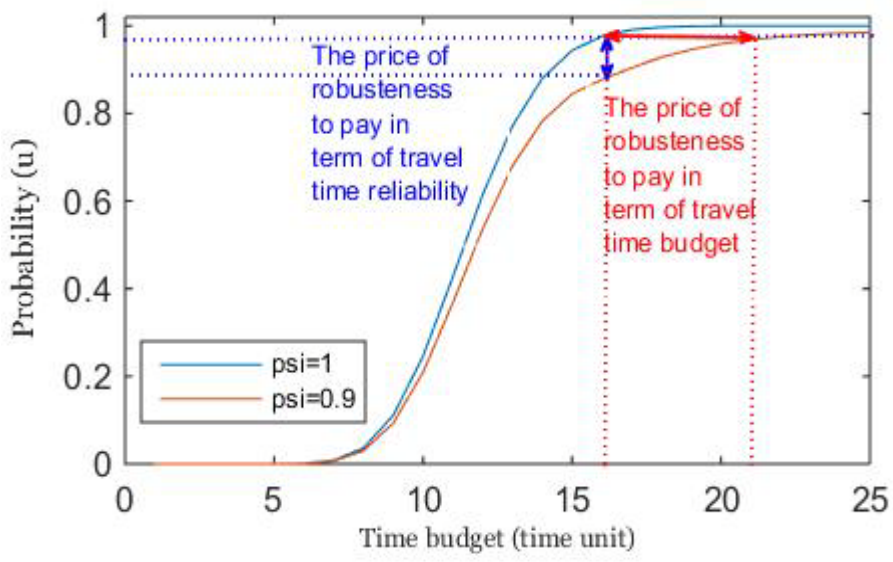


Figure 9. The price of robust-optimality

According to Figure 9 we can also say that, the price of travel-time reliability varies according to the desired time budget, and the price of travel-time budget varies according to the desired reliability. For example, if the user likes to ensure a travel time reliability of 0.6 instead of 0.9742 then the price of robustness that he will pay in term of travel-time budget will be the value 1=13-12(time unit) as illustrated by Figure 10.

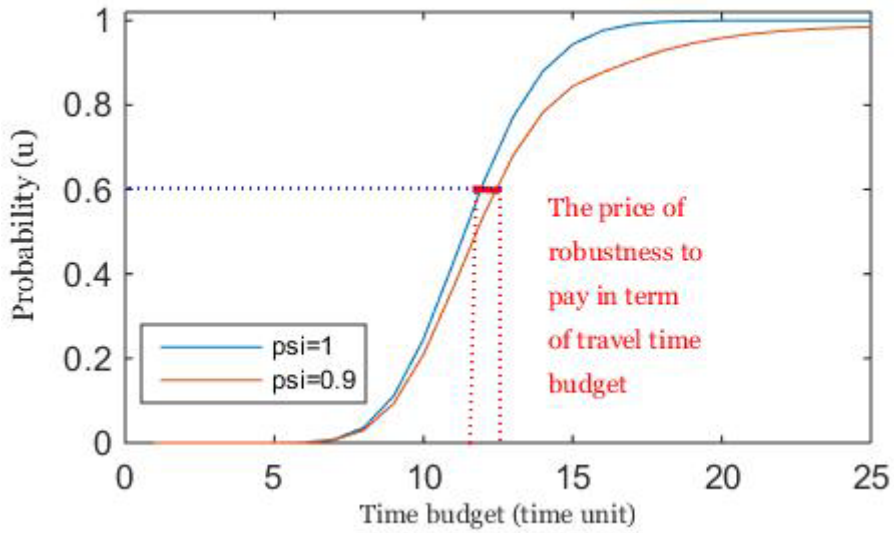


Figure 10

3.2. Formulation of the algorithm in the case of correlated link travel-times

In this section we generalize our algorithm to the case of correlated link travel-times. More precisely, we propose an extension of the model (11)-(13) that takes into account eventual correlation between any consecutive links in the network. For that, we propose to calculate for each node i the probability to reach the destination node d, where we take into account the case where the selected path fails before the users who selected it reach the destination node d; for which case, alternative neighboring paths are used. $u_{ki}(t,y)$ denotes, as above, the probability to reach the destination node d, parting from node i, in a time-budget t, and knowing that the user comes from node k upstream of i, and that the realized travel time from k to i is y. The mathematical definition of $u_{ki}(t,y)$ is the following. We proceed as in section 3. Let us consider the following notation.

$$A_{kij}(t,y) = \int_0^t p_{kij}(w \mid y)\, u_{ij}(t-w, w)\, dw,$$

$$\forall i \in N\backslash\{d\}, k \in \Gamma^{-1}(i), j \in \Gamma^{+1}(i), \forall 0 \le t \le T, 0 \le y \le T - t.$$

Where $p_{kij}(.|.)$ denotes the probability distribution function of the travel-time on link (i,j) conditioned by the one on link (k,i).

For given k, i, t and y, we sort the quantities $A_{kij}(t,y)$ in a decreasing order with respect to the index j as done above in section 3.1. We denote by $A_{ki}(t,y)$ the vector $A_{ki}(t,y) = \left(A_{ki1}(t,y), A_{ki2}(t,y), \dots, A_{kin_i}(t,y)\right)$, where n_i is the number of successor nodes of node i in the graph. We then define N maps $S_i, i = 1,2, \dots, N$, as follows.

$$\begin{aligned} S_i : \mathbb{R}^{n_i} &\to \mathbb{R}^{n_i} \\ A_{ki}(t,y) &\mapsto S_i\left(A_{ki}(t,y)\right) \end{aligned} \quad , \forall i \in \{1,2, \dots, N\}, \forall k \in \Gamma^{-1}(i)$$

where $S_i\left(A_{ki}(t,y)\right)$ is the vector whose components are the same as those of $A_{ki}(t,y)$ but sorted in a decreasing order. $S_{ij}\left(A_{ki}(t,y)\right)$ denotes here the j^{th} component $\left(S_i\left(A_{ki}(t,y)\right)\right)_j$ of vector $S_i\left(A_{ki}(t,y)\right)$.

We then rewrite the probability for a user to reach the destination node d from node i in a time budget t, knowing that the user comes from node k and that the travel-time from k to i is y, as follows.

$$u_{ki}(t,y) = \sum_{p=1}^{m} \psi_p S_{ip}\left(A_{ki}(t,y)\right), \quad \forall i \neq d, k \in \Gamma^{-1}(i), 0 \leq t \leq T, 0 \leq y \leq T - t \tag{14}$$

$$u_{kd}(t,y) = 1, \forall k \in \Gamma^{-1}(d), 0 \leq t \leq T, 0 \leq y \leq T - t, \tag{15}$$

where m is a parameter giving the number of successor nodes taken into account in the sum of formula (14), and ψ_p are weighting coefficients satisfying

$$\psi_p \geq 0, \forall p \in \{1,2, \dots, m\}, \qquad \sum_{p=1}^{m} \psi_p = 1, \qquad \text{and} \quad \psi_1 \geq \psi_2 \geq \cdots \geq \psi_m.$$

The optimal guidance strategy is then determined by the sequence of successor nodes $s_{ki}(t,y)$ as given by the formula (16) below.

$$s_{ki}(t,y) = \arg\max_{j\in\Gamma^{+1}(i)} \left(A_{kij}(t,y)\right) \quad (16)$$

4. How to fix the parameters ψ_p

As mentioned above, in order that the model (14)-(16) has a meaning, ψ_p have to be chosen such that $\psi_1 \geq \psi_2 \geq \cdots \geq \psi_m$. We will be interested here in the case where $m = 2$ (i.e. we only take into account the two best successor nodes of every node i). In this case, we have two weighting parameters ψ_1 and ψ_2, such that $\psi_1 + \psi_2 = 1$. In order to simplify the notations, we simply denote $\psi = \psi_1$, and ψ_2 is given by $\psi_2 = 1 - \psi$. Therefore, we have only one parameter ψ for the robustness, such that the case $\psi = 1$ corresponds to the case where robustness is not taken into account; while the obtained routing strategy is as robust as the parameter ψ is small. We notice here that ψ should satisfy $1/2 \leq \psi \leq 1$, since we have $\psi_1 \geq \psi_2$.

We propose in this section a method to fix the parameter ψ in such a way that a user will be able to manage his travel time budget to favour travel time reliability with respect to robustness. More precisely, the user fixes a desired travel time reliability that he aims to reach, and by that, the routing algorithm will use the entire available travel time budget in order to reach the desired level of travel time reliability; and the remaining travel time budget is used to improve the robustness of the routing strategy. The details are given below.

Given a travel time budget τ and a desired travel time reliability p (expressed as the probability that the destination will be reached on the time budget τ); given a time interval ψ to which the parameter ψ belongs (for example $\psi = (1/2, 1]$), the optimal weighting coefficient ψ^* is determined as follows (formulas (17)-(18) below).

The desired travel time reliability p being fixed, the algorithm first calculates the map $\psi \mapsto T_{kiy}(p, \psi)$ giving, for every value of ψ, the minimum time budget needed to satisfy the desired travel time reliability p.

$$T_{kiy}(p,\psi) = u_{ki}^{-1}(\underline{t},y,\psi) := \min(t, t \geq 0, u_{ki}(t,y,\psi) \geq p) \quad (17)$$

The notation $u_{ki}^{-1}(\underline{t},y,\psi)$ denotes the pseudo-inverse of the non-decreasing map $t \mapsto u_{ki}(t,y,\psi)$; see Figure 11.

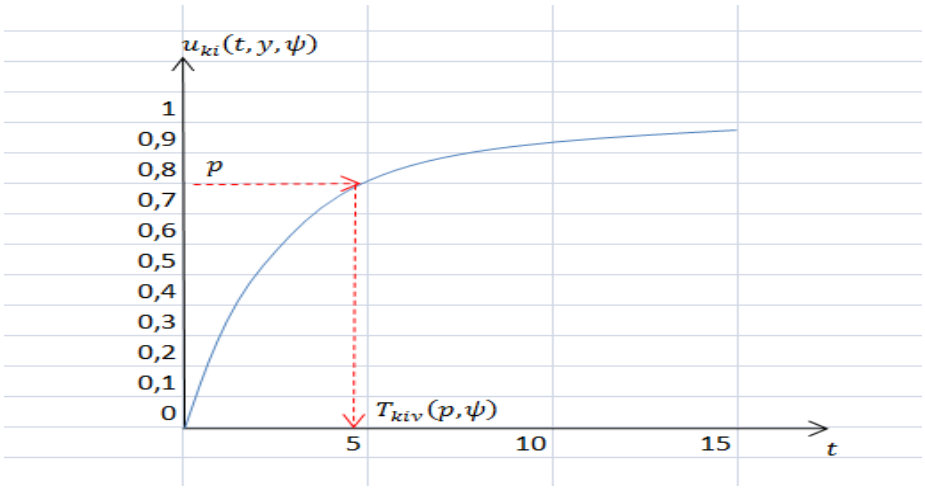


Figure 11. How to choose the minimum time budget $T_{kiy}(p,\psi)$?

Then, the desired travel time budget being fixed, the algorithm calculates the optimal robustness parameter ψ^* needed to satisfy the constraint of travel time budget.

$$\psi_{kiy}^{*}(\tau,p) = T_{kiy}^{-1}(p,\underline{\psi}) := \min(\psi, \psi \in [1/2,1], T_{kiy}(p,\psi) \leq \tau) \quad (18)$$

The notation $T_{kiy}^{-1}(p,\underline{\psi})$ denotes the pseudo-inverse of the non-increasing map $\psi \mapsto T_{kiy}(p,\psi)$; see Figure 12.

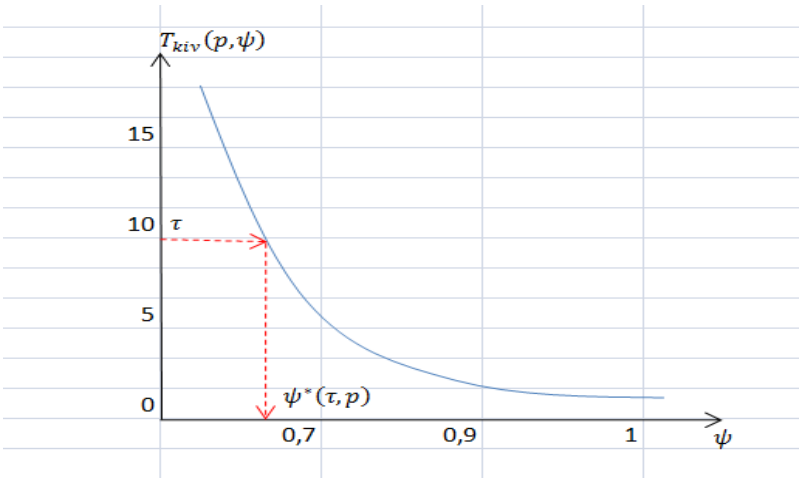


Figure 12. How to choose the optimal robustness parameter $\psi^*_{kiy}(\tau,p)$?

Therefore, the calculus of $\psi^*_{kiy}(\tau,p)$ consists in inversing the maximum cumulative probability distribution $u_{ki}(t,y,\psi)$ once on the variable t in order to derive, for every fixed value of robustness parameter ψ, the optimal time budget $T_{kiy}(p,\psi)$ needed to satisfy the desired travel time reliability p; and a second time (inversing $T_{kiy}(p,\psi)$) on the variable ψ in order to calculate the optimal robustness parameter ψ^* needed to satisfy the constraint on the travel time budget.

The optimization of robustness given by equations (17)-(18) is general, in the sense that it includes the case where one only likes to optimize travel time reliability, and not path-failure robustness. In this case, one can just let ψ belonging to the singleton $\{1\}$, i.e. set $\psi = 1$. Moreover, if one tries to optimize robustness, but he does not have any margin on the time budget that permits this optimization, equation (18) will fix systematically ψ^* to the value $\psi^* = 1$. In other terms, any margin on the travel time budget is first used to optimize travel-time reliability, and after that, the remaining margin is used to optimize robustness.

5. Generalized algorithm for time-varying distributions

The algorithm that we proposed in the previous section remains valid in the static case because the distribution of the travel times on the links are static i.e. do not vary over time. Networks in the real world are, nevertheless, stochastic and time-varying, and the link travel-times are stochastic processes (i.e. random variables with probability distribution functions varying in time). One of the most used approaches when solving shortest path problems on networks with time-dependent travel times is to consider the corresponding temporal extended graph with static travel times. If the network satisfies the First In First Out (FIFO) condition defined by Astrarita in [38], then the problem can be solved using a modified version of Dijkstra's algorithm with the original graph. Dean explored in [144] the shortest path problem in FIFO time-dependent network, with waiting policies, and assumed that waiting at a node is never beneficial. Samaranayake et al [128] explored the problem by the same way as Dean, showed that commonly used travel-times estimates satisfy the FIFO condition, and made propositions that waiting at node is not optimal. Some definitions and propositions made in [128] remain valid for our model. For this reason, we have just cited them before exposing the time-varying model. For more details please see [128-143-144].

Proposition 3. A stochastic discrete-time traffic estimate such that link travel-time distributions are fixed for each time discretization yields a stochastic FIFO path.

Definition 2 (Stochastic FIFO condition).

Let u_π^t denotes the cumulative travel-time distribution on path π when departing at time t. The graph satisfy the stochastic FIFO condition if and only if:

$$u_\pi^{t_1}(T) \geq u_\pi^{t_2}(T - (t_2 - t_1)), 0 \leq t_1 \leq t_2, t_2 - t_1 \leq T$$

This definition states that at any given time and on any given path on the network, departing as soon as possible yields a greater probability of arriving on time than delaying the departure.

Proposition 4. Under a stochastic FIFO network, waiting at a node cannot improve the probability on time arrival.

According to proposition 4, an optimal policy for the robust guidance problem does not prescribe waiting at node, when the assumption from proposition 3 is satisfied. Therefore, the problem with time-varying link travel times can be treated by time indexing the link travel-time distributions. We denote here by $p_{kij}^{\tau}(w, y)$ the probability distribution function of the travel time on link (i, j) at w, knowing that the user comes from the upstream node k of i, that the realized travel time on link (k, i) is y, and that the user leaves node i at time τ. Similarly, $u_{ki}^{\tau}(t, y)$ denotes the robust-optimal probability of reaching the destination node d within time budget t, parting from node i at time τ, and knowing that the user comes from the upstream node k of i, and that the realized travel time on link (k, i) is y. We use similar notations as above.

$$A_{kij}^{\tau}(t, y) = \int_0^t p_{kij}^{\tau}(w \mid y) u_{ij}^{\tau+w}(t - w, w) dw \tag{19}$$

$$\forall i \in N \backslash \{d\}, k \in \Gamma^{-1}(i), j \in \Gamma^{+1}(i), \forall 0 \leq t \leq T, 0 \leq y \leq T - t, 0 \leq \tau$$

As explained in section 3, For given τ, k, i, t and y, we sort the quantities $A_{kij}^{\tau}(t, y)$ in a decreasing order with respect to the index j. We denote by $A_{ki}^{\tau}(t, y)$ the vector $A_{ki}^{\tau}(t, y) = \left(A_{ki1}^{\tau}(t, y), A_{ki2}^{\tau}(t, y), \dots, A_{kin_i}^{\tau}(t, y)\right)$, where n_i is the number of successor nodes of node i in the graph. We then define N maps $S_i, i = 1,2, \dots., N$, as follows.

$$\begin{aligned} S_i : \mathbb{R}^{n_i} &\to \mathbb{R}^{n_i} \\ A_{ki}^{\tau}(t, y) &\mapsto S_i\left(A_{ki}^{\tau}(t, y)\right) \end{aligned} \quad , \forall i \in \{1,2, \dots., N\}, \forall k \in \Gamma^{-1}(i), 0 \leq \tau$$

Where $S_i\left(A_{ki}^{\tau}(t, y)\right)$ is the vector whose components are the same as those of $A_{ki}^{\tau}(t, y)$ but sorted in a decreasing order. $S_{ij}\left(A_{ki}^{\tau}(t, y)\right)$ denotes here the j^{th} component $\left(S_i\left(A_{ki}^{\tau}(t, y)\right)\right)_j$ of vector $S_i\left(A_{ki}^{\tau}(t, y)\right)$.

We then write $u_{ki}^{\tau}(t, y)$ as follows.

$$u_{ki}^{\tau}(t, y) = \sum_{p=1}^{m} \psi_p S_{ip}\left(A_{ki}^{\tau}(t, y)\right), \tag{20}$$

$$\forall i \neq d, k \in \Gamma^{-1}(i), 0 \leq t \leq T, 0 \leq y \leq T - t, 0 \leq \tau$$

$$u_{kd}^{\tau}(t, y) = 1, \forall k \in \Gamma^{-1}(d), 0 \leq t \leq T, 0 \leq y \leq T - t, 0 \leq \tau \tag{21}$$

This model is optimal according to proposition 4. As we can see in equation (19), the departure time from node i (superscript of $A^{\tau}_{kij}(.)$) is the same at the time spent on link (i,j) (superscript on $p^{\tau}_{kij}(.)$) , which justifies the fact that waiting at node is not beneficial for the optimal policy in the SOTA problem.

6. Static routing in Sioux Falls network

We use here the well known Sioux Falls network to test our algorithm. This network is simplified to 24 nodes and 76 links as illustrated in Figure 13 bellow.

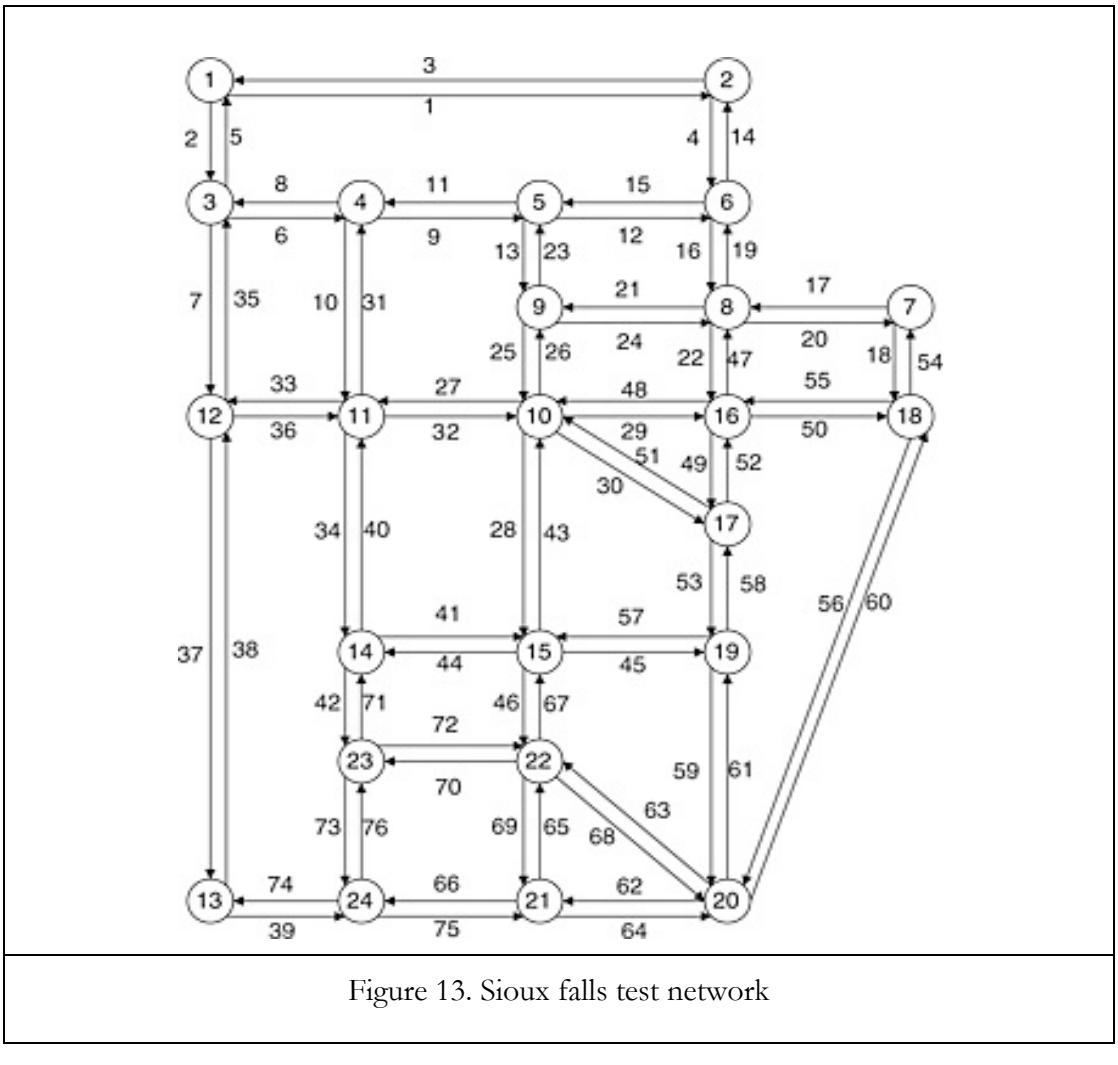


Figure 13. Sioux falls test network

We assume that the link travel times on the network of Figure 13 are drawn from bi-variate Gamma distribution. More precisely, we assume that the joint probability distribution of the travel times on two successive links of the network is a bi-variate Gamma. We base here on the bi-variate Gamma distribution given by Smith et al. (1982)

[142]. Indeed, the joint PDF of two positively correlated random variables W and Y, with Gamma marginal distributions f_W and f_Y with shape and scale parameters (α_W, β_W) and (α_Y, β_Y) respectively, was derived by Smith et al. (1982) as follow.

$$f(w,y) = \begin{cases} \frac{g_1}{g_2} \sum_{k_1=0}^{\infty} \sum_{k_2=0}^{\infty} c_{k_1 k_2} (\beta_W w)^{k_1} (\eta \beta_Y y)^{k_1+k_2} & if \ \rho > 0 \\ f_W(w) \cdot f_Y(y) & if \ \rho = 0 \end{cases} \quad (22)$$

where ρ is the product-moment correlation coefficient of W and Y, estimated from the sample data,

$$\rho = \frac{E[(\mathrm{W}-m_{\mathrm{W}})(\mathrm{Y}-m_{\mathrm{Y}})]}{\sigma_W \sigma_Y} \quad (23)$$

$\mathrm{m_W}, \mathrm{m_Y}$ and σ_{W}, σ_{Y} are the sample means and standard deviations of the variables W and Y respectively.

$$w, y \geq 0, \qquad 0 < \eta = \rho \sqrt{\alpha_y / \alpha_w} < 1, \qquad \alpha_y \geq \alpha_w, \qquad 0 \leq \rho \leq \eta \sqrt{\alpha_w / \alpha_y}$$

$$g_1 = (\beta_{\mathrm{W}} w)^{\alpha_{\mathrm{W}}-1} (\beta_{\mathrm{Y}} y)^{\alpha_{\mathrm{Y}}-1} \exp\left(-\frac{\beta_{\mathrm{W}} w + \beta_{\mathrm{Y}} y}{1-\eta}\right) \quad (24)$$

$$g_2 = (1-\eta)^{\alpha_W} \, \Gamma(\alpha_W) \, \Gamma(\alpha_Y - \alpha_W) \quad (25)$$

$$C_{k_1 k_2} = \frac{\eta^{k_1+k_2} \Gamma(\alpha_Y - \alpha_W + k_2)}{(1-\eta)^{2k_1+k_2} \Gamma(\alpha_Y + k_1 + k_2) k_1! k_2!} \quad (26)$$

$\Gamma(.)$ is the Gamma function $\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt$.

In the network of Figure 13, even though the maximum number of successors over all the nodes is equal to 5, we take here, for simplification, $\mathrm{m} = 2$, with weighting coefficients ψ_1 and ψ_2, and denote $\psi = \psi_1$ and then we have $\psi_2 = 1 - \psi$.

To reach the destination node 10 parting from node 1, we have 2979 elementary paths. We apply our algorithm (model (14-16)) and derive the probabilities $\mathrm{u_{ki}(t,y)}$ for all origin nodes i of the network. In order to illustrate our approach, let us consider the following parameters. Travel-times on two successor links of the network follow a bi-variate Gamma

probability distribution, with given average travel times tt_l for links l and variance-covariance matrix Cov, with $Cov_{ll} = 3, \forall l$ and $Cov_{ll'} = 1.5, \forall l \neq l'$. The average travel times we take here are shown in Figure 14.

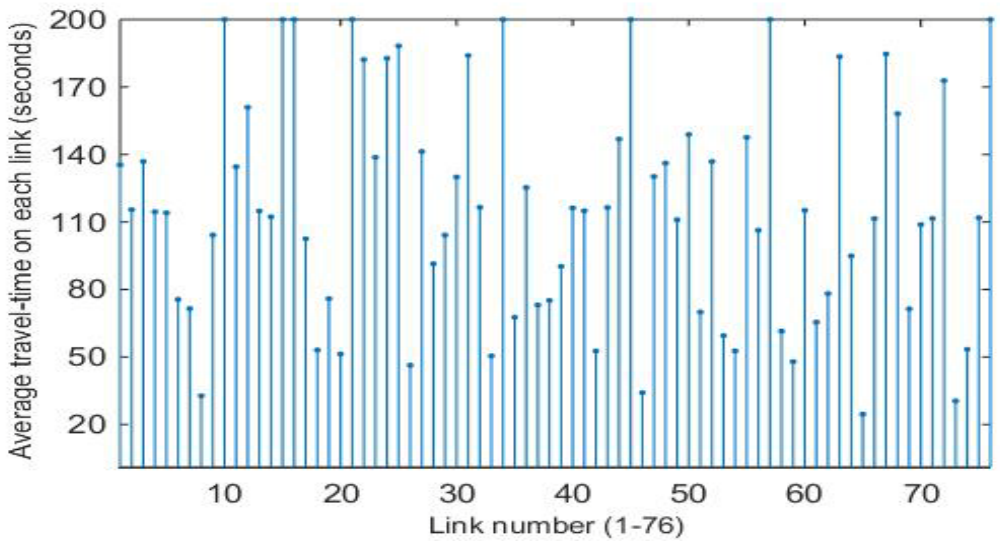


Figure 14. The average travel-time on each link (seconds)

As mentioned in section 3, our approach takes into account the existence and the performance of alternative detours of the selected paths, in the calculus of the guidance strategy. The proposed algorithm takes into account the fact that one or many links of the selected optimal path may fail during the travel. We consider that users may be sensitive to path changing. That is to say that they may prefer paths with efficient alternative detours, with respect to paths without, or with less efficient detours, even with a loss in the average travel time, and/or in its reliability. To illustrate our algorithm, we will be interested in what will happen at node 3 in two different scenarios presented in sub-section 6.1, and sub-section 6.2.

In the following, we apply our algorithm (model(14-16)) for a user being at node 1 and going to node 10. We assume that the user has a time budget equal to 2400 seconds (40 minutes). We apply here a version of Algorithm 1 which takes into account travel time correlations (model (14)-(16)). In order to illustrate the algorithm, we consider two scenarios. In scenario 1, we use the network of Figure 13. We show that the algorithm

gives the path passing by nodes (1,3,12,11,10) as the robust-optimal one in the two cases of $\psi = 1$ (without taking into account the robustness criterion) and $\psi \in (1/2, 1)$ (by taking into account the robustness criterion). For scenario 2, we remove link 37 form the network of Figure 13. By removing link 37, node 12 will have only 2 successor nodes. Then, the user being at node 3 will have to chose between the two successor nodes: node 4 (with 3 successor nodes) and node 12 (with only 2 successor nodes). We will show that the algorithm gives the same optimal path as for scenario 1, in the case $\psi = 1$, but it gives another optimal path, the one passing by nodes (1,3,4,11,10), in the case $\psi \in (1/2, 1)$. Indeed, path (1,3,4,11,10) has more alternatives than path (1,3,12,11,10), since, in scenario 2, node 4 has more successor nodes than node 12.

6.1. Scenario 1

In this first scenario, we consider the network of Figure 13. Let us consider the probability $u_{13}(t, y)$ of reaching the destination node 10 from node 3, coming from node 1, and knowing that $tt_{13} = y$. We take here $y = 115$ seconds. In this first scenario, we have three routing actions from node 3 coming from node 1: go to successor node 12, which will give us three routing actions at the next step (go to successor node 11, or 13 or back to 3), or go to successor node 4, which will give us three routing actions at the next step (go to 5 or 11, or back to 3). The simulation results are given in Figure 15 and Figure 16. Figure 15 shows the robust-optimal probability $u_{13}(t, 115)$ in function of the time budget t, and for different values of ψ. Figure 16 gives the optimal successor nodes $s_{13}(t, 115)$ in function of the time budget t, and for different values of ψ.

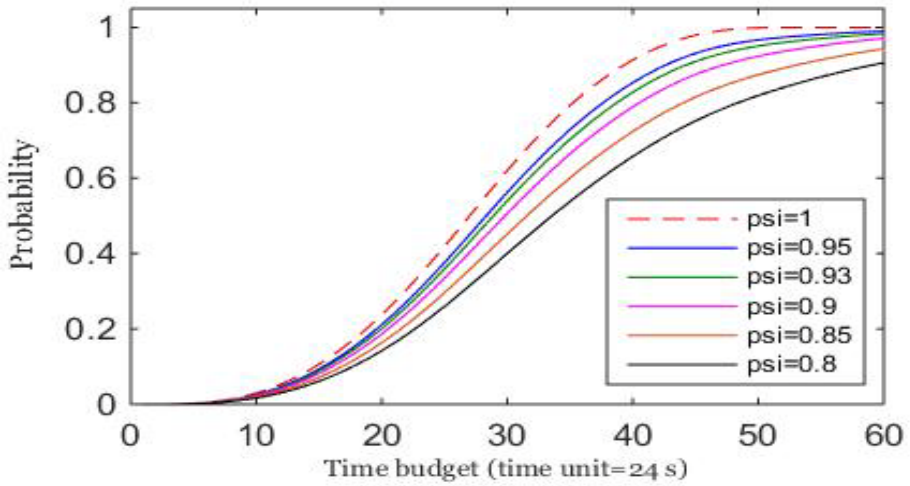


Figure 15. The robust-optimal probability of arriving on time in function of the time budget, and for different values of ψ.

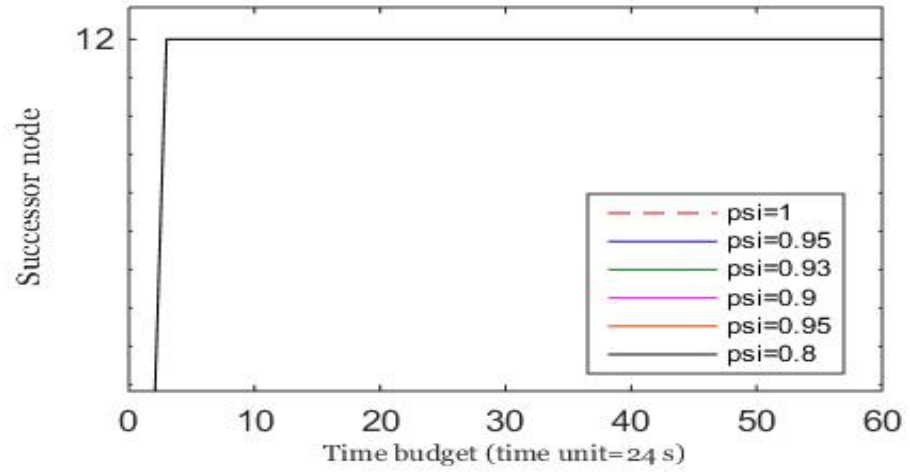


Figure 16. The robust-optimal routing policy on node 3 in function of the time budget, and for different values of ψ.

The case $\psi = 1$ (dashed line) corresponds to the version of the model (6)-(8) taking into account travel time correlations, while the cases $\psi \in (1/2, 1)$ (solid lines) correspond to the model (14)-(16). We see from Figure 7 that, for any time budget t, the robust-optimal

probabilities $u_{13}(t, 115)$ decrease as the values of ψ decreases. That is to say that for lower values of ψ, lower values of $u_{13}(t, 115)$ are obtained. This is because we replaced a maximum operator in (6)-(8) by a mean value in (14)-(16). Indeed, a user taking a lower value of ψ, asks for more path-robustness or path-flexibility, and, in the counterpart, he loses in term of travel-time reliability. The difference $u_{1,3}(t, 115, \psi_1) - u_{1,3}(t, 115, \psi_2)$ can then be interpreted as the price of path-robustness corresponding to a measure of it, given by the difference $\psi_1 - \psi_2$. We can see also from Figure 15 that, , for any desired probability u (travel time reliability), the time budget needed to attain this fixed level of reliability increase as the values of ψ decreases. That is to say that for lower values of ψ, bigger time budget are needed to satisfy the asked level of reliability. Indeed, a user taking a lower value of ψ, asks for more path-robustness or path-flexibility, and in order that he satisfies the fixed level of reliability, he will need more time budget. The difference $u_{1,3}^{-1}(t, 115, \psi_2) - u_{1,3}^{-1}(t, 115, \psi_1)$ can then be interpreted as the price of path-robustness corresponding to a measure of it, given by the difference $\psi_1 - \psi_2$. Figure 16 gives the robust-optimal routing policy in term of the robust-optimal successor nodes. We see from that figure that the optimal routing policy is the same for every value of ψ, and corresponds to the path passing through nodes (1,3,12,11,10).

6.2. Scenario 2

This second scenario aims to illustrate our approach with respect to the robustness of the optimal strategy offered by our algorithm, in the case of path failure. The scenario is the following. We take the same network of Figure 13, but we remove link 37. By that, we penalize (in term of robustness) the passage by node 12, since we decrease the number of successor nodes of it. Therefore, node 12 will have only two, instead of three successor nodes. At node 3, coming from node 1, we have three routing actions: go to successor node 12, which will give us only two routing options at the next step (go to successor node 11 or back to 3), or go to successor node 4, which will give us three routing options at the next step (go to successor nodes 5 or 11, or back to 3). Indeed, if we chose node 12 as successor of node 3, then at node 12, if link 36 fails, we have to back to node 3. However, if we chose node 4 as successor of node 3, then at node 4, if one of the links 9 or 10 fails, we have the possibility to change and take a detour using the other link; see Figure 13. Therefore, passing by node 4 allows us more options than passing by node 12. We will see

below that our algorithm is able to take into account such robustness criterion in the selection of the optimal routing strategy.

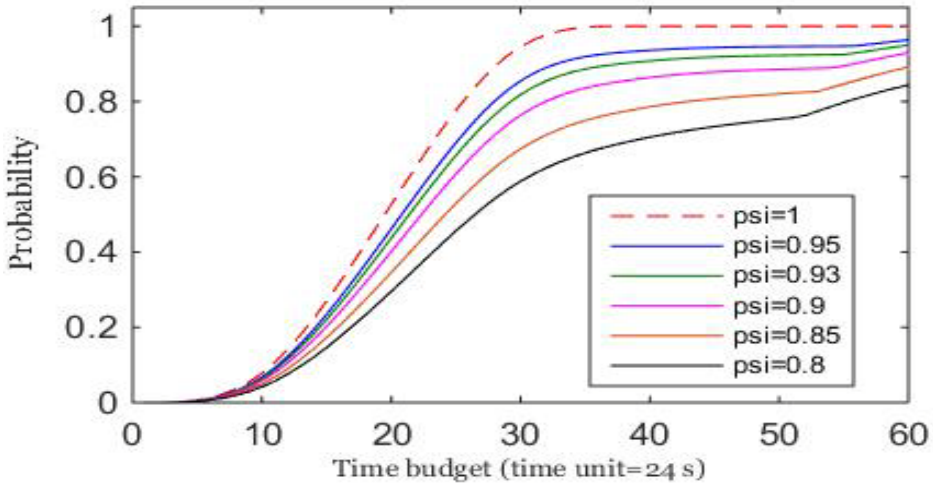


Figure 17. The probability of arriving on time in function of the time budget, and for different values of ψ

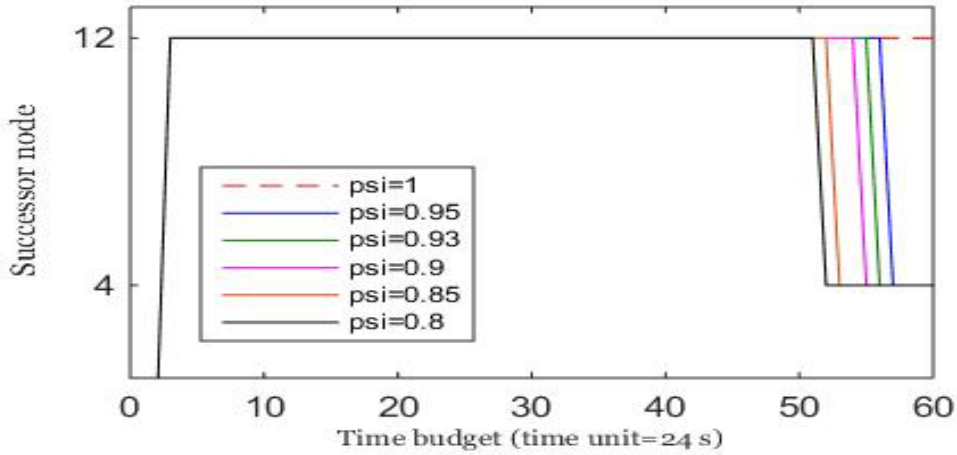


Figure 18. The routing policy on node 3 in function of the time budget, and for different values of ψ

In scenario 2, the optimal policies obtained by the algorithm for the two cases of $\psi = 1$ and $\psi \in (1/2, 1)$ are clearly different. For the case $\psi = 1$, we obtain the same optimal policy as in scenario 1 (dashed line in Figure 18). However, with low values of ψ ($\psi = 0.95$, respectively 0.93, 0.9, 0.85, 0.8), where path-robustness is considered, we see that, with a time budget equal or bigger than 57 time units (a time unit is 24 seconds here) (respectively 56, 55, 53 time units), the robust-optimal successor node of node 3 is node 4 rather than node 12, even though paths passing through node 12 have lower average travel time comparing to those passing through node 4. For example, for $\psi = 0.95$, we see that with a time budget lower than 56 time units, the algorithm prefers paths passing through node 12 (see blue line in Figure 18). However, with a time budget higher than 56 minutes, the optimal policy on node 3 changes and node 4 becomes the robust-optimal successor node (see blue line in Figure 18). That means that, node 12 which has only two successor nodes is penalized, i.e. it gets low values $u_{ki}(t, y)$. Therefore, paths that pass through that node i.e. paths with small number of alternatives or detours have low probability to be selected as robust-optimal paths.

In order to illustrate the price of robustness here, let us take a time budget t = 55 seconds, and distinguish two cases.

- **Case 1 :** $\psi = 0.95$. In this case we do not give a high importance to the robustness. Then we see clearly on Figure 18 (blue line) that the algorithm still gives node 12 as the optimal successor node of node 3.
- **Case 2 :** $\psi = 0.8$. In this case we give more importance to the robustness criterion. Then we see clearly on Figure 18 (black line) that the algorithm chooses node 4 as the optimal successor node of node 3.

Therefore, to pass from a robustness level of 0.95 to a higher robustness level of 0.8, a user needs to increase its time budget from 53 to 57 time units, i.e. to sacrifice 4 time units (96 seconds) of his time budget. We can conclude that, in this example, the price of 0.15 units of the robustness parameter ψ is 96 seconds.

7. Conclusion

In this chapter, we considered the optimal guidance problem of users in road networks, and proposed a new robust adaptive strategy. We based on an existing routing model, which is the SOTA algorithm, and extended it to take into account robustness of routing strategies against path failure. In order to include the performance of alternative detours of the selected paths, we extended the concept of reliability by introducing a new reliability index. The improvement we made here allows the selection of an optimal path according to two criteria: the reliability of the path in term of travel time and the robustness of the path in term of its flexibility (i.e., existence and performance of alternative detours). Finally, the developed algorithm was tested in static version (without considering traffic dynamics) and shown some interesting properties.

In the next chapter, we will illustrate the effectiveness of the guidance algorithm in some dynamic scenarios. For that, we will combine the guidance algorithm with a dynamic traffic model by using the traffic simulator SUMO.

Chapter 3

Combination of the robust routing algorithm with a dynamic traffic model

Summary

1. Introduction

In this chapter we combine the robust guidance algorithm presented in the previous chapter with a dynamic traffic model in a closed loop, in order to see how the algorithm reacts dynamically to the state of traffic. We use here the microscopic traffic simulator SUMO (Simulation of Urban MObility) and we illustrate our results on the Sioux Falls network. We begin this chapter by explaining in a general way the procedure we followed in our work to recover the state of traffic and implement our robust-optimal guidance algorithm. We then introduce SUMO and present the different packages used, before describing the configuration steps. We conclude by presenting and commenting on the results obtained.

2. How to proceed?

In order to see how the algorithm reacts dynamically to the state of traffic, we combine it with a dynamic traffic model in a closed loop. We use the microscopic traffic simulator SUMO (Simulation of Urban MObility) [146], with the traffic control interface Traci (Traffic control interface), where we implement our robust-optimal guidance algorithm. We illustrate our results on the Sioux Falls network below (Figure 19).

We apply the guidance here only for the traffic demand parting from the zone around origin 1 to the zone around destination 10 because the optimization of the code is not yet done which leads to a significant calculation time. The routing of all other origin-destination traffic demands is done by SUMO, by means of DUAROUTER algorithm that we will present in the next section. The application of our algorithm in parallel with another dynamic routing algorithm permits also to evaluate its efficiency in case where only some travelers are optimizing the robustness of the guidance against link failures.

We define a time period (10 minutes here) at the end of which we apply our robust-optimal guidance algorithm. The algorithm is applied periodically on the network. More precisely, at every time period, we retrieve the state of traffic (average travel times and variations) and give it to the algorithm. The latter proposes an itinerary from origin 1 to destination 10.

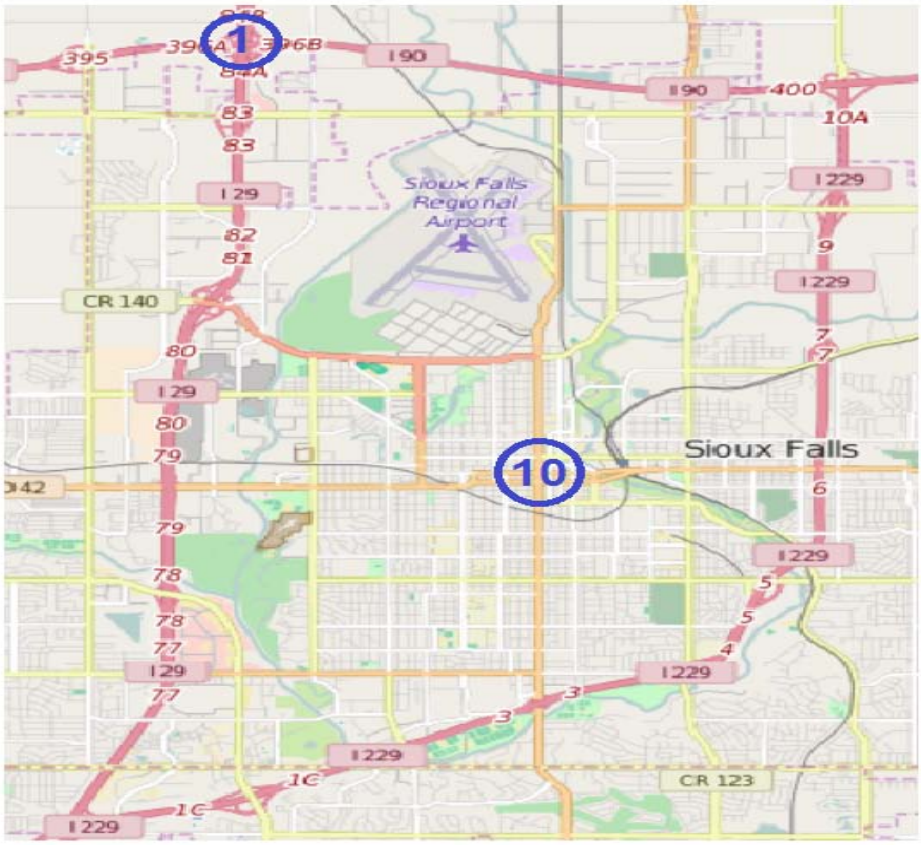


Figure 19. Sioux Falls network, image from OpenStreetMap

We apply the obtained itinerary for the considered time period, and for the associated traffic demand, and apply the itineraries obtained by DUAROUTER for all the other traffic demands. We iterate this process for a given simulation time (40 minutes).

3. Configuration of the traffic simulation with SUMO

For the simulation phase of our work, we have chosen microscopic simulation tool for urban mobility SUMO. It is an open source and highly portable traffic simulator [147]. It is a purely microscopic traffic simulation tool where each vehicle is explicitly represented (identifier, start time, distance, source, destination, speed...). It lends itself well to various types of research, for example the route choice problem, management algorithms of traffic lights or simulating the inter-vehicular communication. This architecture is used in many projects to simulate and study various traffic management strategies. SUMO also offers the opportunity to interact with an external application via a socket connection: it is an

interface to operate the simulations in client-server mode. In this mode, the client can dynamically control the execution of road simulation through a traffic control interface called Traci (Traffic Control Interface).

We briefly explain in this sub-section the requirements for the initialization of the urban traffic simulator SUMO. Several applications contained in the SUMO package were used to generate the road network and road traffic (vehicle routes). The parameters and the interactions between applications that we used to create our urban environment simulation are shown schematically in Figure 20.

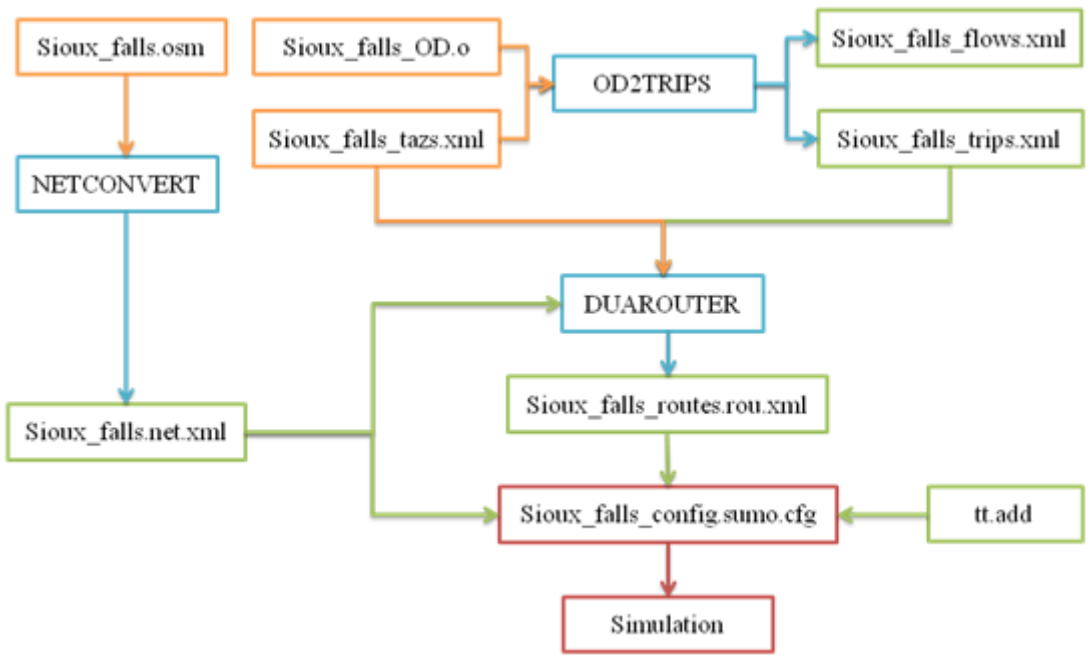


Figure 20. Files configuration of Sioux falls network

We use three applications in the SUMO package to generate the route network and the vehicle routes. The first one is NETCONVERT tool which imports digital road networks from different sources and convert them into a format read by SUMO. The second application is OD2TRIPS which can generate traffic information (trips, flows) from the OD matrix into road networks. The third application is DUAROUTER which imports

produced definitions (trips) by OD2TRIPS and generates vehicles and routes into road networks. We explain each of these configuration steps in the following.

3.1. Edit and modify the network with JOSM

First of all, we take the same network (Sioux Falls road network in South Dakota of the United States of America) (Figure 19). The network is downloaded from the OpenStreetMap as a simulation case. To bring changes in the Sioux Falls network, the OSM xml files are edited using the JAVA editor (JOSM) in order to remove streets, buildings, railway, etc which are not used in the simulation. The real network downloaded from OpenSteetMap contains 12440 nodes and 31162 edges (Figure 19). After modification, the final network is given in Figure 21 below.

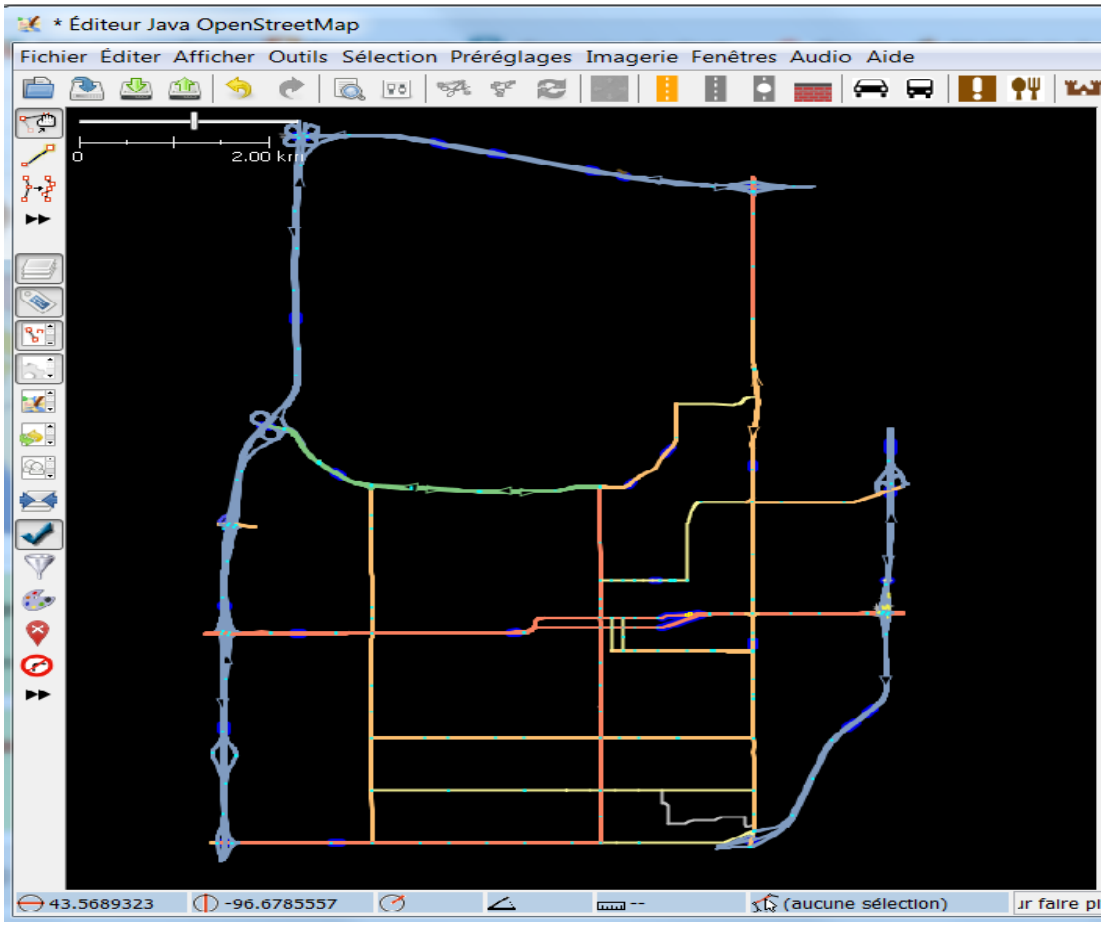


Figure 21. Sioux Falls network, simplified diagram on the JOSM (Sioux_falls.osm)

The road properties (such as speed limits, traffic lights, length edges, etc.) are included in the OSM data, and with the NETCONVERT tool, theses OSM data are converted into the network xml file <Sioux_falls.net.xml> as explained in the following.

3.2. Import Sioux Falls network with NETCONVERT

NETCONVERT imports digital road networks from different sources and converts them into a format read by SUMO. In this work, we imported our network presented in Figure 2. Traffic lights and access to the motorway junctions are instantiated by NETCONVERT using the geometry of the network nodes. The simplified network consists of 24 zones that contains 502 nodes and 774 small edges as illustrated in Figure 22.

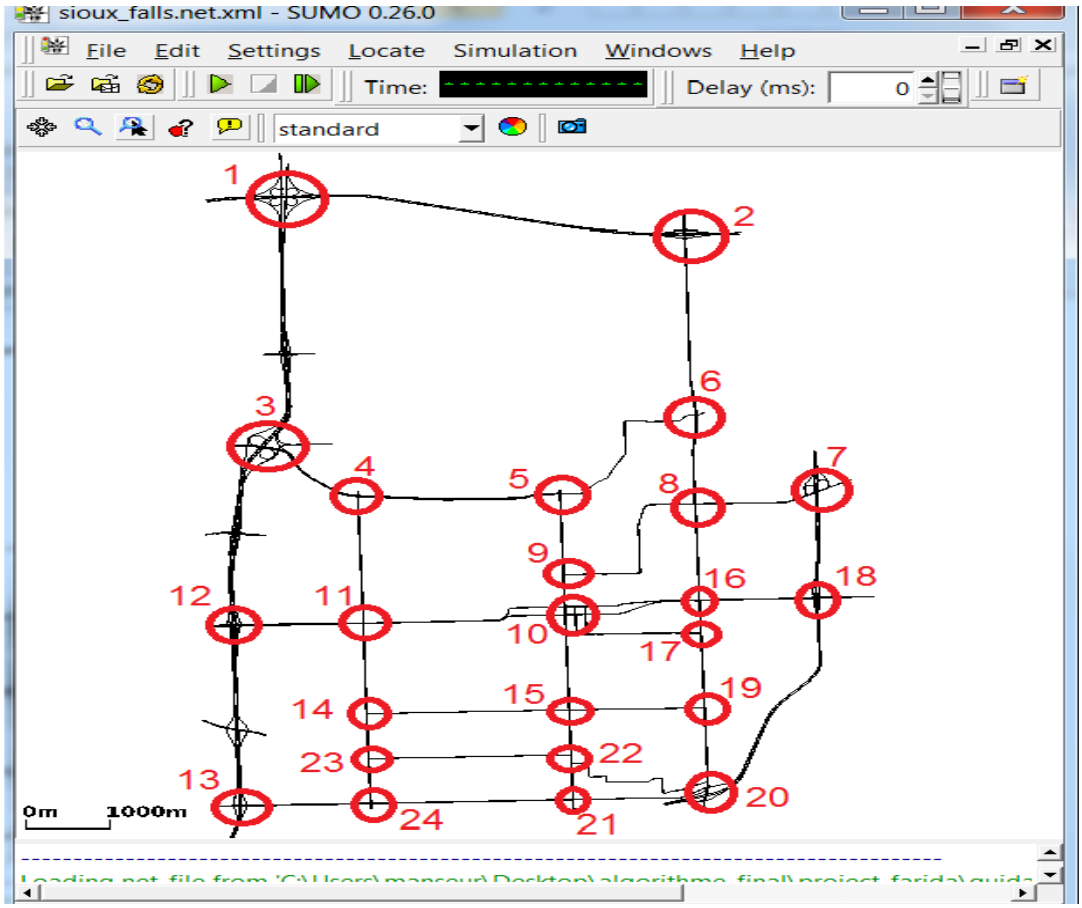


Figure 22. Sioux Falls network, simplified diagram on the SUMO (Sioux_falls.net.xml)

3.3. Generate a single vehicle trips by OD2TRIPS

Once the network is ready, the next step consisted in the generation of the traffic. For this, we built in SUMO a file containing a mapping from traffic assignment zones to edges (TAZ) <Sioux_falls_tazs.xml>. We collected data on demand from [147] in order to build the Origine/Destination matrix <sioux_falls_od.O>. This matrix describes the movements between so called traffic analysis zones (TAZ) in vehicle number per time. The program OD2TRIPS of SUMO reads O/D matrices and TAZ files to generate trip definitions <Sioux_falls_trips.xml> and flows <Sious_falls_flows.xml>. The resulting trips obtained from OD2TRIPS consist of a start and an end road together with a departure time. However, the simulation requires the complete list of edges to pass. Such routes are usually calculated by performing a dynamic user assignment (DUAROUTER). This is an iterative process employing a routing procedure such as shortest path calculation under different cost functions.

3.4. Generate a complete specification of the vehicles and their routes by DUAROUTER

DUAROUTER is an algorithm for Dynamic User Assignment which has been proposed in [151-152]. With DUAROUTER algorithm, vehicles are guided along routes selected according to calculated probabilities. The travel time for each car is measured, and the probabilities are adjusted accordingly, so that all the traffic is distributed. This is repeated for a defined number of iteration steps. However, since DUAROUTER does not optimize the travel time for each user, some users are guided along detour paths.

In our case, we imported produced definitions <Sioux_falls_trips.xml> and calculate vehicle routes <Sioux_falls_routes.rou.xml> using DUAROUTER. The result is a complete specification of the vehicles and their routes: ID, color, maximum speed, destination, time of departure, route, etc.

The final step before initiating SUMO is to create a configuration file <Sioux_falls_config.sumo.cfg> pointing to all files that use SUMO to its initialization. In our case, we use the net file <Sioux_falls.net.xml>, the routes file <Sioux_falls_routes.rou.xml> and the additional file <tt.add>. The configuration file

contains also the port value <8813> for Traci-Server to establish the connection as illustrated in Figure 20.

The <tt.add> is an an additional file we created to save all generated <ttOutput(id).xml> files between the time period "PeriodTime" and the next one "periodTime=+periodTime". In our simulation, we initialized the time period to "periodTime=60 seconds". The generation of <ttOutput(id).xml> files is described in the following

3.5. Generate a sumo-outputs SUMO-GUI

There are two versions of the traffic simulation. The application "SUMO" is a pure command line application for efficient batch simulation. The application "SUMO-GUI" is basically the same application as "sumo", just extended by a graphical user interface "GUI" rendering the simulation network and vehicles using open GL. "sumo" and "SUMO-GUI" allow generating various outputs for each simulation run.

We have generated the <sumoOutput.xml> file by SUMO-GUI simulation using summary <File> as summarized in Figure 23.

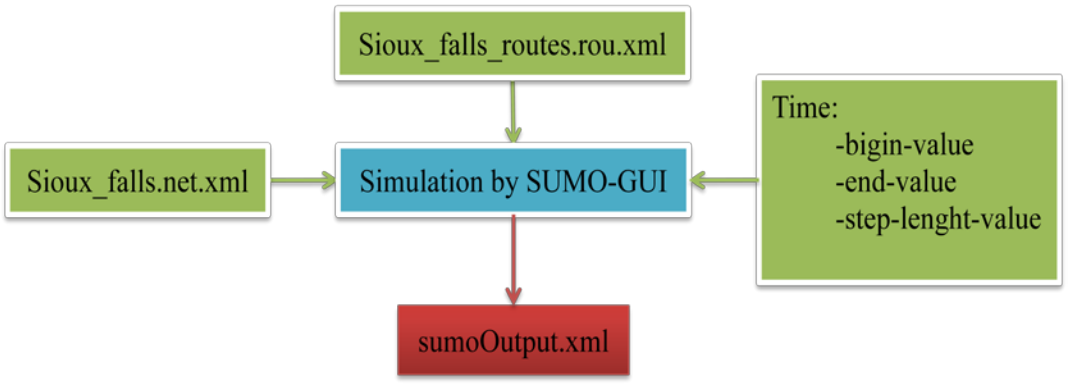


Figure 23. Data output about vehicles using Simulation/ output/ Summary

This output <sumoOutput.xml> file contains the simulation-wide number of vehicles that are loaded, inserted, running, waiting to be inserted, have reached their destination and how long they needed to finish the route. The last value is normalized over all vehicles that

have reached their destination so far. The information containing all those values is computed for each time step of 0.0001 seconds. The time begin value is started to 0 seconds, and the time end value is 2400 seconds .

3.6. Control by using Traffic Control Interface (TraCI)

The Traci interface is an extension of SUMO, which provides a communication channel with SUMO. The command library in Traci is extensive and includes control of several items such as vehicles, traffic lights, roads, links, etc. SUMO package comes with Traci interface coded in Python. However, several researchers have developed interfaces compatible with other programming languages. Recently, Traci for Matlab [148] and Traci for Java [149] have been introduced but are not as comprehensive as those coded in Python. We coded our algorithm on python, so we use Traci interface for Python [150] and we have completed the necessary commands to control our simulation.

4. The car driver model

SUMO uses by default an extension of the stochastic car-following model (the Gipps model extension) proposed by Stefan Krauss [17]. The model reproduces the main features of traffic, namely free and congested flow. At each time step, the vehicle's speed is adapted to the speed of the leading vehicle in a way that it results in a collision-free system behavior within the following simulation step. The model of Krauss is chosen because of its simplicity and its high execution speed. In this section, we give a brief reminder of this model.

The model developed by Stefan Krauss, 1998, is a microscopic, space-continuous, car-following model based on the safe speed paradigm. That means that a pilot tries to stay away from the driver who precedes him at a safe distance and speed that allows him to adapt his movement with respect to eventual decelerations of the leader. The model assumes the driver to have a reaction time τ of about one second. The safe velocity is computed using the following equation.

$$v_{safe}(t) = v_l(t) + \frac{g(t) - g_{des}(t)}{\tau_b + \tau} \quad (27)$$

where

$v_l(t)$ represents the speed of the leading vehicle in time t.

$g(t)$ is the gap to the leading vehicle at time t

τ is the driver reaction time, usually 1s.

The time scale τ_b is defined as $\frac{\bar{v}}{b(\bar{v})}$, where b is the maximum deceleration of the vehicle (m/s^2) and $\bar{v}$ is the average velocity of the leader and the follower $\bar{v} = (v_l + v_f)/2$

g_{des} is the desired gap. It can be chosen in different ways. For example, in [6] it was chosen to be $v_l \tau$.

As $v_{safe}(t)$ may be larger than the maximum speed v_{max} allowed on the road or larger than the vehicle is capable to reach until the next step due to his acceleration capabilities, the minimum of those values is called the desired speed $v_{des}(t)$ computed as next :

$$v_{des}(t) = \min(v_{safe}(t),\ v(t) + at,\ v_{max}) \tag{28}$$

where a represents maximum acceleration of the vehicle (m/s^2).

Assuming that the driver is note able to perfectly adapt his desired velocity $v_{des}(t)$, the driver's imperfection value ϵ multiplied with the car's acceleration ability a and a random number is subtracted from the desired velocity. One must assume that the vehicle is not driving backwards. Due to this, the last of the model's equation is:

$$v(t) = \max(0, \text{rand}[v_{des}(t) - \epsilon at, v_{des}(t)]) \tag{29}$$

ϵ is a parameter between 0 and 1. $\text{rand}[x_1, x_2]$ denotes a random number between x_1 and x_2.

There are four free parameters in this model: the maximum velocity v_{max}, the maximum acceleration a, the maximum deceleration b, and the noise parameter ϵ.

The velocity, multiplied with the simulation step duration, which is constantly equal to one second, here, is added to the vehicle's current position to achieve the position for the next time step.

This model presented here is the single lane traffic model. In reality, however, this situation is hardly ever found on highways. Instead, a road generally is made up of two or more lanes, which allow vehicles to pass. Stefan Krauss has generalized the model to multilane traffic. The model formulation had been based on a few very simple assumptions:

- A lane change is performed, if it is favourable and safe.
- Passing on the right side is allowed only under congested conditions.
- There is a small probability p_{change} that a safe lane change is performed, even if it is not favourable.

5. Dynamic travel-times estimation

The dynamic data of travel-times on the network of Figure 4 are obtained by using the SUMO function "getTraveltime [String edgeID]" that returns the estimated travel time (in seconds) on the given edge within the last time step. This function needs as input parameters a string identifying of the edge. To apply this function, we need to retrieve all identifiers of short edges on the network, knowing that this network contains 24 zones and 76 large links between zones as shown in Figure 21. However, each big link consists of a set of nodes and a number of short edges. The network of Figure 21 contains a total of 502 nodes and 774 short edges. In order to estimate the travel time on the 76 big links, we proceeded as follows.

- Retrieve identifiers of all the 774 short edges on the network.
- Calculate the estimated travel-times on these short edges.
- Calculate the travel-times on the 76 big links.

In the following, we will detail each of these three parts.

5.1. Retrieve identifiers of 774 short edges

To recover these identities, we first created a first file <zone.trip.xml>, which contains the input edge and the output edge for a zone, and a second file <link.xml> which contains the input edge and the output edge for a big link. We have merged these two files in order to have a single file <zone_link_i.trip.xml>, where i is an index on big link ($i = 1,2, \dots ,76$). We have proceeded in the same way for each zone and for each big link, so we have built 76 files, one for each link.

Thereafter, we used DUAROUTER, in order to retrieve in a file <edges_route_i.rou.xml> for the route edges for each big link as shown in the following diagram (Figure 24).

Figure 24. Retrieving identifiers for short edges

We have retrieved 76 files, one for each big link of the network, and each file <edges_route_i.rou.xml> contains all the identifiers of the short edges that build a big link.

5.2. Calculate the estimated travel-times on the short edges

After having retrieved the identifiers of the short edges, we used the sumo function "getTraveltime [String edgeID]", in order to obtain the estimated travel time on each short edge as illustrated in Figure 25.

Figure 25. Estimating travel-times on short edges

5.3. Calculate the travel-times on the big edges

To have the travel time on a big link, we simply added all the travel times obtained on all short edges that build this big link, and we got an estimated travel time every 60 seconds. In our case, we launched a simulation for a total time $Tsim = 2400$ seconds and we retrieved results every 600 seconds, that is to say we simulated for each of the following rang time [0,600], [600,1200], [1200,1800] and [1800,2400] seconds. Therefore, for each of these intervals, we took the average of the estimated travel times. We take the same covariance matrix (variance = 3, covariance = 1.5).

The average travel-times on the big links are given in Figures 26, 27, 28 and 29 for the time periods [0,600], [600,1200], [1200,1800] and [1800,2400] seconds respectively.

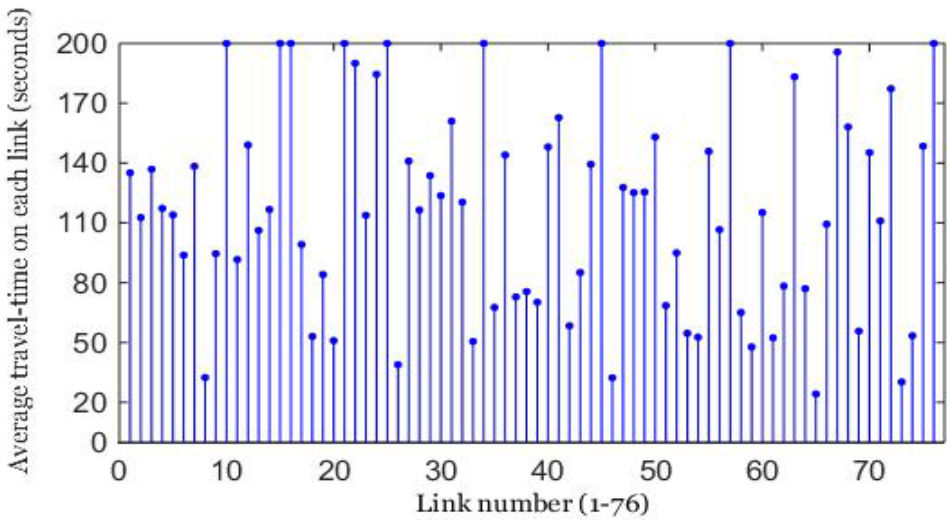


Figure 26. Average travel -time on each link for [0, 600]s

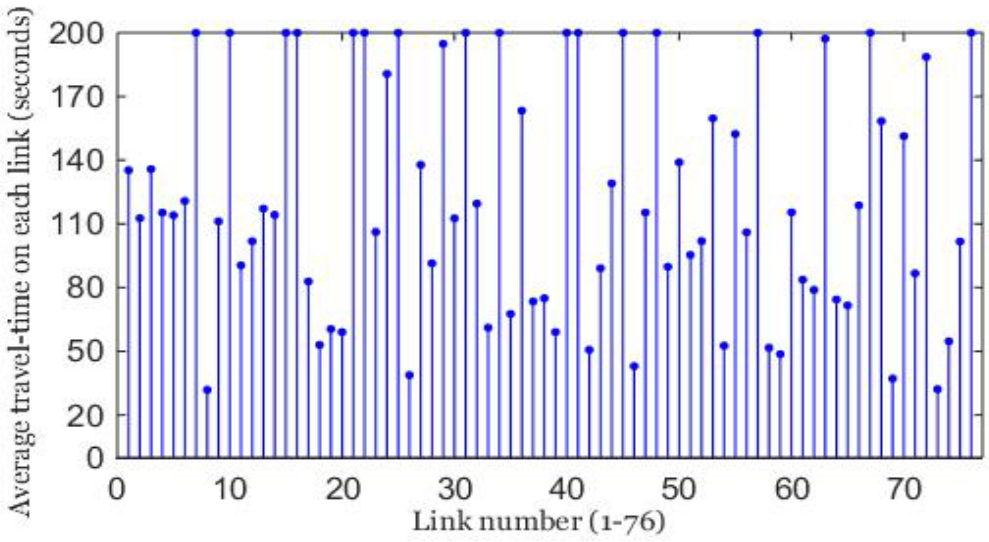


Figure 27. Average travel -time on each link for [600, 1200]s

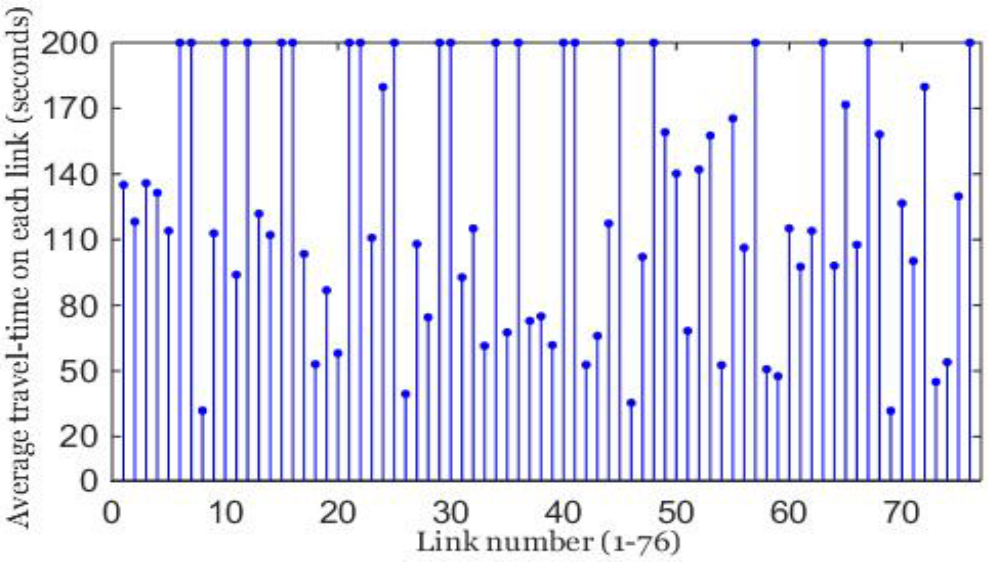


Figure 28. Average travel -time on each link for [1200, 1800]s

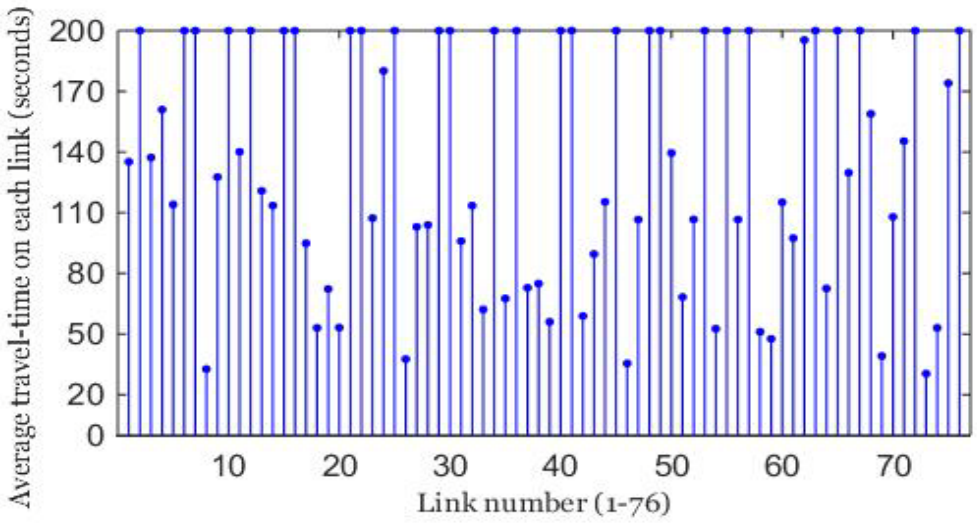


Figure 29. Average travel -time on each link for [1200, 1800]s

6. Simulation procedure

The guidance is applied in its static form but periodically in time. The traffic is simulated by SUMO for a fixed period of time. At the end of the time period, travel times on the links of the network are retrieved, and estimations are made on the average travel times and on their variations. By that, we obtain an estimation of the bi-variate Gamma probability distribution of every couple of successive links in the network. With these travel time distributions, we apply the robust guidance algorithm for the traffic demand corresponding to the origin destination pair (1, 10). We recall here that for the traffic demand corresponding to the other origin-destination pairs, the assignment is done by SUMO using DUAROUTER algorithm. The optimal and robust itineraries (obtained by our algorithm for the origin-destination pair (1, 10), and obtained by DUAROUTER for the other origin-destination pairs) are then used by SUMO to load the network and simulate the traffic for another time period. We iterate this procedure up to a given total simulation time. An example of simulation during the second time period (i.e. [600, 1200] seconds) is illustrated in Figure 30 below.

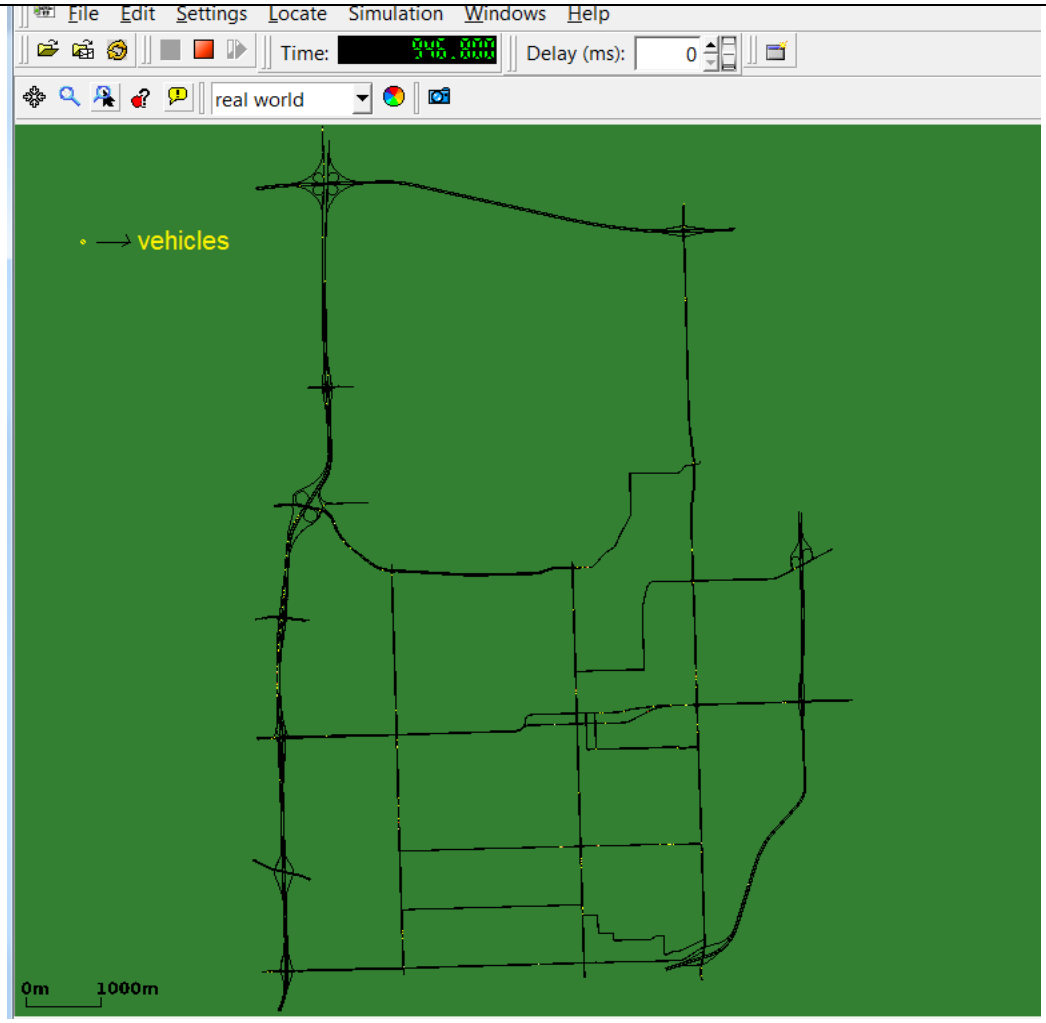


Figure 30. A simulation example during the second time period.

7. Simulations results

For the simulation with SUMO, we have used the network of Figure 22, and set node 1 as the origin, and node 10 as the destination. We considered a total simulation time of 40 minutes, with a time period of 10 minutes, at the end of which, we apply periodically the guidance algorithm for the traffic demand from 1 to 10. The guidance algorithm is applied as follows. We assume that we are at node 1 and that we have spent a time y on the upstream link, and we seek to reach destination node 10. The maximum time budget we consider here is $T = 40$ minutes. The routes obtained at different times are given in Table 6 and Figures 31, 33 and 34, for $\psi = 1$ (without taking into account robustness criterion) and for $\psi = 0.9$ (with taking into account robustness criterion). Figure 33 gives the

maximum probability of arriving on time at destination node 10 from node 1, coming from node 2, corresponding to the case $\psi = 1$. Figure 34 gives the robust-optimal probability of arriving on time at destination node 10 from node 1, coming from node 2 corresponding to the case $\psi = 0.9$.

Table 6. Optimal route for $\psi = 1$ and optimal robust route for $\psi = 0.9$

	$\psi = 1$	$\psi = 0.9$
Time periods (minutes)	Optimal route	Robust-optimal route
[0, 10]	**2 → 7 → 36 → 32** (Figure 30.a)	**2 → 6 → 10 → 32** (Figure 30.b)
[10, 20]	**2 → 7 → 36 → 32** (Figure 30.a)	**2 → 6 → 10 → 32** (Figure 30.b)
[20, 30]	**2 → 6 → 10 → 32** (Figure 30.b)	**2 → 6 → 10 → 32** (Figure 30.b)
[30, 40]	**2 → 6 → 10 → 32** (Figure 30.b)	**2 → 6 → 10 → 32** (Figure 30.b)

This table chows the different routes obtained at each time period, in the case where $\psi = 1$ (without taking into account robustness criterion) and in the case where $\psi = 0.9$ (with taking into account robustness criterion). Figure 31 show the optimal route and the optimal robust route from zone 1 to zone 10.

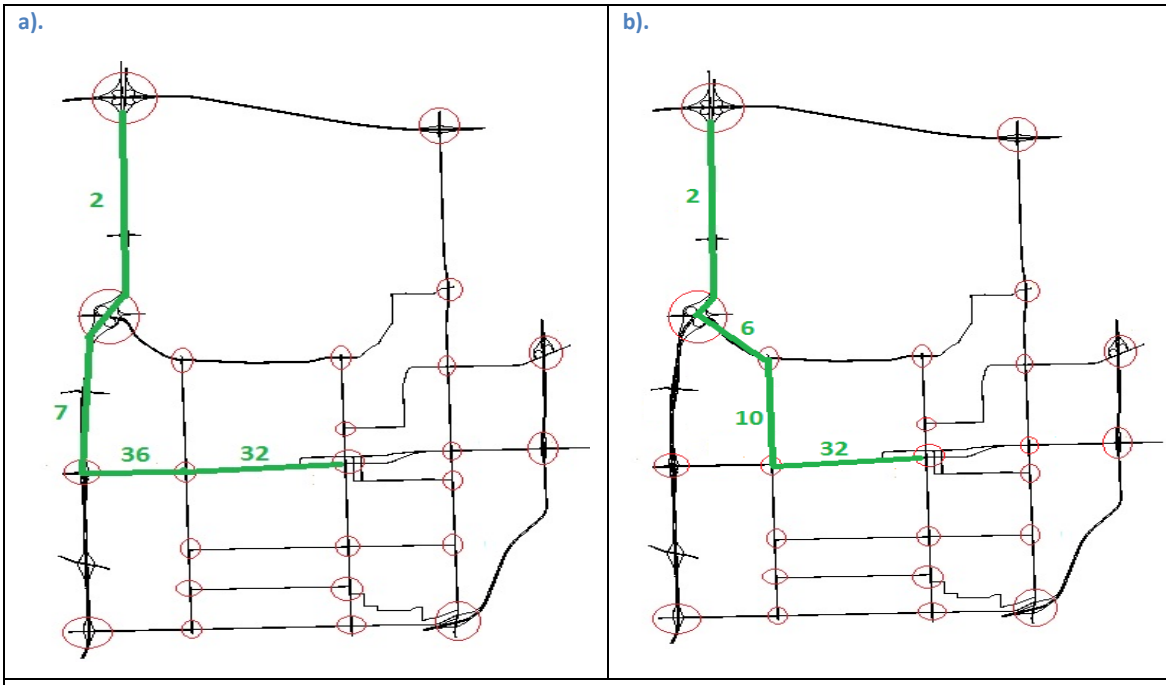


Figure 31. The optimal route and the optimal robust route from zone 1 to zone 10

Figure 32 gives the estimated travel-times on links constructing the optimal path and the robust optimal path.

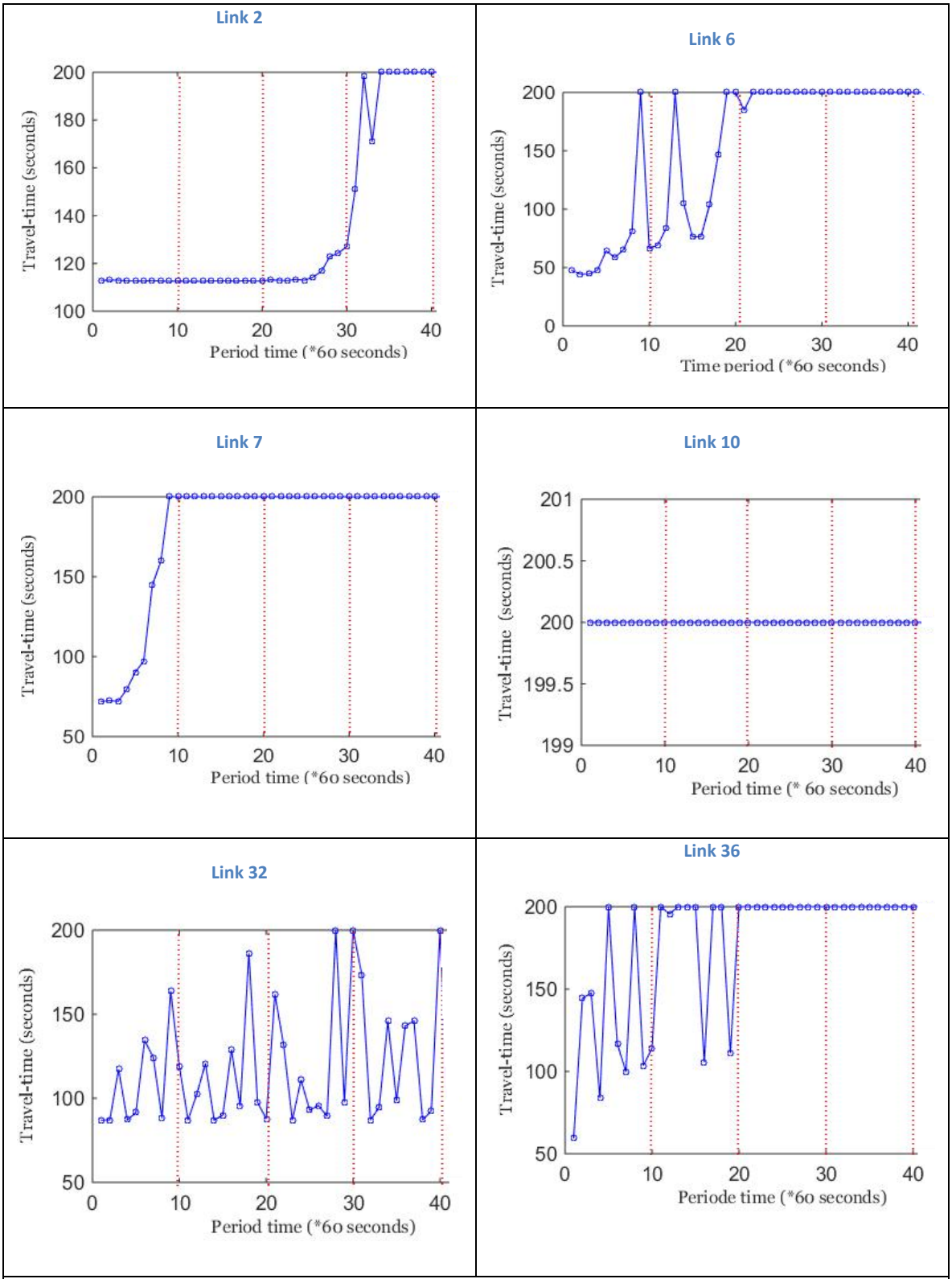


Figure 32. Estimated travel-times on links

Figure 33 gives the maximum probability of arriving on time at destination node 10 from node 1, coming from node 2, corresponding to the case $\psi = 1$.

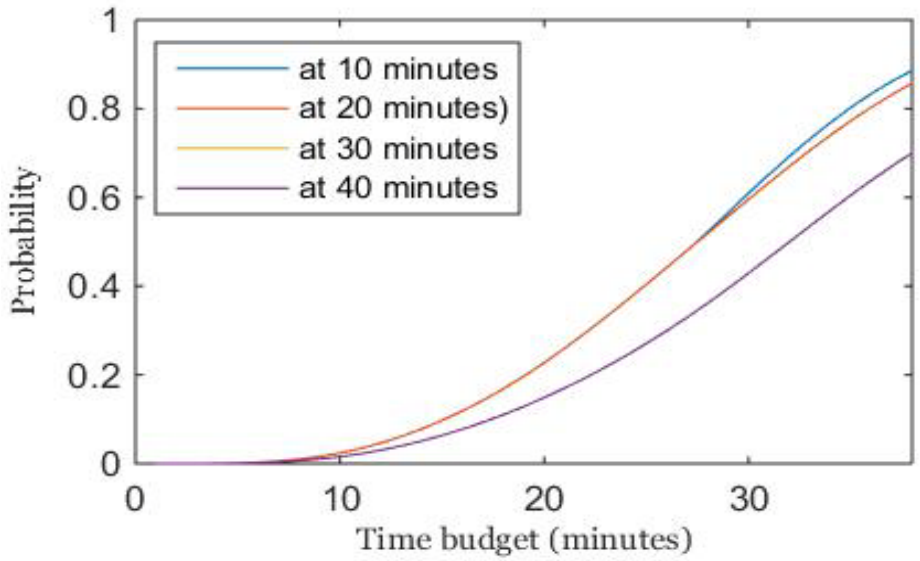


Figure 33. The maximum probability of arriving on time at destination node 10 from node 1 coming from node 2, for $\psi = 1$.

The optimal paths corresponding to the maximum cumulative probability distributions are given in Table 6.

Figure 34 gives the robust-optimal probability of arriving on time at destination node 10 from node 1, coming from node 2 corresponding to the case $\psi = 0.9$.

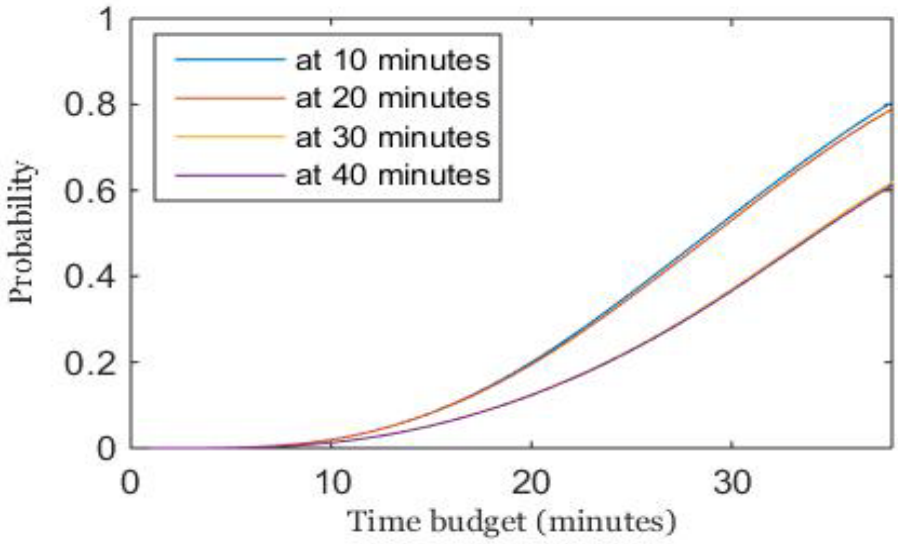


Figure 34. The probability of arriving on time at destination node 10 from node 1 coming from node 2, for $\psi = 0.9$.

The robust-optimal paths corresponding to the robust-optimal cumulative probability distributions are given in Table 6.

For $\psi = 1$, the optimal route obtained at 10 and 20 minutes is the one constructed of links [2-7-36-32]. At 30 minutes, the optimal strategy changes to [2-6-10-32]. We obtain the same route at 40 minutes. This optimal route is different from that obtained at 10 minutes. The optimal strategy obtained from node 3 changes, and node 4 becomes the optimal successor node, instead of node 12. Indeed, we observed that the link 7 is highly congested during the time periods [20,30] and [30,40] minutes. The guidance algorithm then proposes to rather pass by link 6, during these two periods of time.

For $\psi = 0.9$, where robustness is taken into account by the guidance algorithm, the robust-optimal route obtained during all the period times is the one constructed of links [2-6-10-32]. We explain here why the algorithm proposes this robust-optimal route during the time periods [0,10] and [10,20] minutes, where the link 7 is not yet congested, and where the guidance algorithm with $\psi = 1$ proposes rather path [2-7-36-32]. From node 3 coming from node 1 we have three routing options: go to successor node 12, which will give us

three routing actions at the next step (back to 3, go to 11, or go to 13), or go to successor node 4, which will give us also three routing actions at the next step (back to 3, go to 11, or go to 5), or back to node 1. It is clear that the option of backing to node 1 is not very interesting. Let us now compare the two options of going to node 4, and going to node 12. The two nodes 4 and 12 have two common successor nodes 3 and 11. The difference is in the third successor node (node 5, successor for node 4, and node 13, successor for node 12). Thus, we need to compare the two nodes 5 and 13. In terms of routing robustness, node 5 is better than node 13, since node 5 has three successor nodes, while node 13 has only two successor nodes. Therefore, it is better to pass by node 4 rather than passing by node 12. We recall here that the number of successor nodes is important for the robust guidance because, if we are at a given node with many successor nodes, we have more options in case of failures of any successor link. In this example, being at node 4 is better than being at node 12 because, if we are at node 4 and if link 10 fails, we need to choose between backing to node 3 or to go to node 5, which has three routing options (successor nodes), but if we are at node 12 and if link 36 fails, we have to choose between backing to node 3 (which option is also offered by node 4) or to go to node 13, which has also two routing options.

8. Conclusion

In order to see how the algorithm reacts dynamically to the state of traffic, we combined it with a dynamic traffic model in a closed loop. We used the microscopic traffic simulator SUMO (Simulation of Urban MObility), with the traffic control interface Traci (Traffic control interface), where we implemented our robust-optimal guidance algorithm. We illustrated our results on the Sioux Falls network.

Conclusion générale

Les travaux réalisés dans cette thèse concernent le problème du guidage des usagers dans les réseaux routiers. Un algorithme de guidage a été développé et une nouvelle stratégie adaptative robuste a été proposée. Le travail a été basé sur un modèle de routage existant, qui est l'algorithme SOTA. Cet algorithme a été étendu pour tenir compte de la robustesse des stratégies de routage face à une faille qui peut survenir sur l'itinéraire. Afin d'inclure la possibilité ainsi que la performance d'éventuels détours alternatifs des chemins sélectionnés, le concept de fiabilité a été étendu en introduisant un nouvel indice de fiabilité. Les améliorations portées sur cet algorithme permettent de sélectionner un chemin optimal selon deux critères: la fiabilité du chemin en termes de temps de parcours et la robustesse du chemin en termes de flexibilité (c.-à-d. l'existence et la performance de détours alternatifs). L'algorithme développé a été testé en version statique (sans considérer la dynamique du trafic) et les simulations ont été exécutées sur le réseau connu de Sioux Falls. Ceci a permis de montrer quelques propriétés intéressantes de l'algorithme. L'algorithme de guidage robuste présenté dans ce travail a été ensuite combiné avec un modèle dynamique du trafic en boucle fermée, afin de voir comment l'algorithme réagit dynamiquement à l'état du trafic. Le simulateur de trafic microscopique SUMO (Simulation of Urban MObility) a été utilisé avec l'interface de contrôle du trafic TRACI (TRAffic Control Interface) sur laquelle l'algorithme de guidage proposé a été implémenté. Les résultats ont été illustrés sur le réseau Sioux Falls, et ont été comparés dans les deux aspects statique et dynamique, à ceux obtenus par l'approche SOTA de Samaranayake (2012). Ces résultats, aussi satisfaisant soient-ils, nous indiquent tout de même que la performance générale du réseau et de l'algorithme peut encore être améliorée en proposant une série de perspectives.

Perspectives

Afin de compléter ce travail nous proposons une série de perspectives qui devraient améliorer l'algorithme de guidage proposé dans cette thèse.

L'implémentation de l'algorithme développé ici peut encore être améliorée. Une amélioration possible serait d'appliquer les mêmes techniques de prétraitement utilisées par Samaranayake en 2012 pour réduire le temps de calcul.

L'algorithme proposé ne prend en considération que les corrélations entre les liens successifs (c.à.d. la corrélation entre le lien et son prédécesseur). Il serait donc désirable d'inclure les corrélations entre les liens non successifs sur le réseau.

L'algorithme développé dans cette thèse à été testé sur un réseau bien connu Sioux Falls. Cependant l'algorithme développé par Samaranayake en 2012 a été testé sur le réseau de San Francisco. Une application de notre algorithme sur le réseau de San Francisco est souhaitable afin de bien comparer les deux stratégies offertes par les deux algorithmes et également les différents résultats obtenus.

Le guidage a été uniquement appliqué pour les demandes partantes de la zone 1 à la zone 10 du réseau Sioux Falls. Le guidage pour les autres demandes sur le réseau a été calculé par l'algorithme DUAROUTER de SUMO. Il serait donc intéressant de pouvoir appliquer l'algorithme de guidage pour toutes les paires origines-destinations.

L'algorithme développé a été testé dans le contexte de l'affectation du trafic grâce aux outils de simulation microscopique SUMO. Il serait intéressant de l'intégrer dans une application mobile.

Bibliography

[1] V. HENN. Information routière et affectation du trafic : vers une modélisation floue. Thèse de doctorat, Université de Saint-Etienne, France, 2001.

[2] J. Wardrop. Some Theoretical Aspects of Road Traffic Research. Proceedings of the Institute of Civil Engineers, Part II, 1, 325-378, 1952. http://dx.doi.org/10.1680/ipeds.1952.11362

[3] W.S .Vickrey. Congestion theory and transport investment. American Economic Review, Papers and Proceedings59, 251-260, 1969

[4] de Palma A., Ben Akiva M., Lefèvre M., Litinas N., Stochastic equilibriummodel of peak period congestion, Transportation Science, volume 17, pages 430–453, 1983.

[5] Szeto W.Y., Lo H.K., A celled-based simultaneous route choice and departure time choice model with elastic demand, Transportation Research – Part B, volume 38, n°7, pages 593-612, 2004

[6] Mahmassani H.S., Chang G.-L., On boundedlyrational user equilibriumin transpor-tation systems, Transportation Science,volume 21, n°2, pages 89-99, 1987

[7] Lago A., Daganzo C. F., Spillovers, merging traffic and the morning commute, Transportation Research – Part B, volume 41, n°6, pages 670-683, 2007.

[8] D. GAZIS, R. HERMAN et B.POTTS. Car following theory of steady state traffic flow. Operations Research, 7(4):499–501, 1959.

[9] GETRAM/AIMSUN. A model for simulating vehicular traffic on multi-lane and arterial road, version 4.1 users manuals. Rapport technique, Transport Simulation Systems, 2002.

[10] J.P. LEBACQUE. The Godunov scheme and what it means for first-order traffic flow models. In In : Lesort,-J.B. (Ed.),Proceedings of the 13 th International Symposium on Transportation and Traffic theory, pages 647–677, 1996.

[11] M. LIGHTHILL and G. WHITHAM. On kinematic waves ii : A theory of traffic flow on long crowded roads. ProRoySoc, pages 317–345, 1955.

[12] P. RICHARDS. Shock waves on the highway. Operations. Research, 4(1):42–51, 1956.

[13] A. AW et M. RASCLE. Resurrection of second order models of traffic flow. SIAM J. Appl. Math, 60:916–944, 2000.

[14] H. ZHANG. A non-equilibrium traffic model devoid of gas-like behavior. Transportation Research Part B, 36:275–290., 2002.

[15] H. ZHANG. Anisotropic property revisited- does it hold in multi-lane traffic? Transportation Research Part B, 37(6):561–577, 2003.

[16] E. ELLOUMI, H. HAJ-SALEM, et M. PAPAGEORGIOU. "Metacor" a macroscopic modelling tool for urban corridors. In TRISTAN II Int. Conf, Capri, Italie, 1994.

[17] S. KRAUSS. Microscopic modelling of traffic flow : Investigation of collision free vehicle dynamics. Thèse de doctorat, Université de Cologne Allemagne, 1997.

[18] S. Kerner. Experimental features of self-organization in traffic flow. Physcal Review E, 81 :3797–3800, 1998.

[19] A. Kesting. Microscopic modeling of human and automated driving: towards traffic-adaptive cruise control. PhD thesis, Faculty of Traffic Sciences. Technische Universität Dresden (Germany), 2008.

[20] H. Holden, N.H. Risebro. Model of traffic flow on a network of unidirectional roads. SIAM mathematical analysis, 26(4):999–1017, 1995.

[21] M. Treiber, A. Hennecke and D. Helbing. Microscopic simulation of congested traffic. Physical Review E, 62:1805–1824, 2004.

[22] I. Prigogine and R. Herman. Kinetic theory of vehicular traffic. Elsevier, 1971.

[23] T. Peter. Modeling nonlinear road traffic networks for junction control. Applied math, 22 :723–732, 2012.

[24] R. Mahnke and J. Kaupuzs. Stochastic theory of freeway traffic. Physical Review E, 59 :117–125, 1999.

[25] K. Gorkem. Modélisation agent de la perception visuelle humaine limitée appliquée à la simulation du comportement des conducteurs en carrefour. PhD thesis, Université de Valenciennes et du Hainaut-Cambresis, 2013

[26] P. Hidas. Modelling vehicle interactions in microscopic simulation of merging and weaving. Transportation Research Part C : Emerging Technologies, 13(1) :37 – 62, 2005.

[27] J.M. Molina. Commande de l'inter-distance entre deux véhicules. PhD thesis, Institut National Polytechnique de Grenoble, 2005.

[28] S. Germann and R. Isermann. Nonlinear distance and cruise control for passenger cars. In Proceedings of the American Control Conference, 1 :3081–3085,1995

[29] P.A. Ioannou and C.C Chen. Autonomous intelligent cruise control. IEEE Transactions on Vehicular Technology, 42(4):657 – 672, 1993.

[30] H. Al-Jameel, H. Examining and improving the limitations of the gazisherman-rothery car-following model. In Salford Postgraduate Annual Research Conference, 2009.

[31] L. Leclerc and C. Becarie. Meso lighthill-whitham and Richards's model designed for network applications. Transportation Research Part B: Methodological, page 15. 25, 2012

[32] P.GIPPS, «A behavioural car following model for computer simulation,» transportation research board, 1981.

[33] P.GIPPS, «A model for the structure of lane-changing decisions,» transportation research , part B, 20(5):403-414, 1986

[34] R. B. D.GAZIS, «car following theory of steady state traffic flow,» operations research, 7(4):499-501, 1959.

[35] R. Diestel, Graph theory, Graduate Texts in Mathematics Series, vol. 173, electronicedition, http ://www.math.uni-hambourg.de/home/diestel/books/graph.theory, Springer-Verlag, New York 1997, 2000,

[36] C. Berge, Graphes, ISBN 2-04-15555-4, Gauthiers-Villars, Bordas, Paris, 1983.

[37] R.B. Dial. Algorithm 360: Shortest path forest with topological ordering. Communications of the ACM,12:632-633, 1969

[38] N. Deo, Graph theory with applications to engineering and computer science, Prentice-Hall, Englewood cliffs (N.J.), 1975.

[39] E.L. Johnson. On shortest paths and sorting .Dans Proceedings of 25th ACM Annual Conference, pages510-517, 1972.

[40] D. Van Vliet. Improved shortest path algorithms for transport networks. Transportation Research, 12:720, 1978.

[41] I. Murthy et S. Sarkar. A relaxation-based pruning technique for a class of stochastic shortest path problems. Transportation science, 30(3):220-236, 1996.

[42] P.B Mirchandani et H.Soroush. Optimal paths in probabilistic networks: A case with temporary preferences. Computers & Operations Research, 12:365383,1985

[43] D.M. Rasteiro et AB. Anjo. Metaheuristics for stochastic shortest path problem. DansProceedings of 4th MetaHeuristics International Conference (MIC2001), 2001.

[44] R.P. Loui. Optimal paths in graphs with stochastic or multidimensional weights. Communications of the ACM, 26(9):670-676, 1983.

[45] H. Frank. Shortest paths in probabilistic graphs. Operations Research, 17:583-599, 1969.

[46] G. Andreatta, F.Ricaldone and L.Romeo. Exploring stochastic shortest path problems. Rapport Technique, ATTI Giornaledi Lavaro, 1985.

[47] V. G.Adlakha. An improved conditional Monte Carlo technique for stochastic shortest route problem. Management Science, 32(10):1360-1367, 1986.

[48] J. Kamburowski. A note on the Stochastic Shortest Route Problem. Operations Research, 33(6):696-698, 1985.

[49] C.E. Sigal, A.A.B. PritskeretJ .J.Solberg. Stochastic shortest route problem. Operations Research, 28(5):1122 -1129, 1980.

[50] M.Jamali. Learning to Solve Stochastic Shortest Path Problems. Rapport Technique, Sharif University of Technology, 2006.

[51] L.Cooke et E.Halsey. The Shortest Route through a Network with Time-Dependent Intermodal Transit Times. Journal of Mathematical Analysis and Applications, 14:492-498, 1966.

[52] I.Chabini. Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal runtime. Transportation Research Record, 1645:170-175, 1998.

[53] S.E.D reyfus. An appraisal of some shortest-path algorithms. Operations Research, 17:395-412, 1969.

[54] E.Klafszky. Determination of shortest path in a network with time-dependent edge-lengths. Optimization, 3(4): 255-257, 1972.

[55] A.Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge length. Journal of the ACM, 37:607–625, 1990.

[56] A. B. Philpott. Continuous-time shortest path problems and linear programming. SIAM Journal on Control and Optimization, 32:538–552, 1994.

[57] A. B. Philpott and A. I. Mees. Continuous-time shortest path problems with stoppingand starting costs. Applied Mathematics Letters, 5:63–66, 1992.

[58] D.E.Kaufman et R.L. Smith. Fastest Paths in Time-Dependent Networks for Intelligent Vehicle-Highway Systems Application. Journal of Intelligent Transportation Systems, 1(1):111, 1993.

[59] B.H.AhnetJ. Y.Shin. Vehicle-Routing with Time Windows and Time-Varying Congestion. The Journal of the Operational Research Society, 42(5):393-400, 1991.

[60] S.Pallottino et M.G.Scutella. Shortest path algorithms in Transportation models: classical and innovative aspects. Dans Proceedings of the Equilibrium and Advanced Transportation Modelling Colloquium, 1998.

[61] R.W.Hall. The fastest path through a network with random time-dependent travel times. Transportation Science, 20(3):182-188, 1986.

[62] D.Pretolani. A directed hyper graph model for random time-dependent shortest paths. European Journal of Operational Research, 123:315-324, 2000.

[63] E.D.Miller-Hooks et H.S. Mahmassani. Least expected time paths in stochastic time-varying transportation networks .Transportation Science, 34(2):198-215, 2000.

[64] Fredman, M. L. and Tarjan, R. E. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM (JACM), 34(3):596–615, 1987.

[65] A Ben-Tal, A. Nemirovski. Robust truss topology design via semidefinite programming. SIAM Journal on Optimization 7 (4), 991-1016, 1997.

[66] Mikhlin SG, Smolitskiy KL. Approximate methods for solution of differential and integral equations. Elsevier, New York, 1967

[67] A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. Operations Research, 21:1154–1157, 1973.

[68] A. Ben-Tal and A. Nemirovski. Robust convex optimization. Mathematics of Operations Research, 23(4):769–805, 1998.

[69] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. Operations Research Letters, 25:1–13, 1999.

[70] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. Math. Programming A, 88:411-424, 2000.

[71] L. Jenkins. Selecting scenarios for environmental disaster planning. European Journal of Operational Research, 121(2):275 – 286, 2000.

[72] M. Goerigk. ROPI - a Robust Optimization Programming Interface for C++. Optimization Methods and Software, 29(6):1261–1280, 2014.

[73] M. Goerigk, A. Schöbel. Algorithm Engineering in Robust Optimization. arXiv:1505.04901v3 [math.OC] 11 Jan 2016

[74] D. J. Thuente. Duality theory for generalized linear programs with computational methods. Operations Research, 28(4):1005-1011, 1980.

[75] J. E. Falk. Exact solutions of inexact linear programs. Operations Research, 24(4):783-787, 1976.

[76] L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. SIAM Journal of Matrix Anal. Appl., 18:1034-1064, 1997.

[77] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. Robust Optimization. Princeton University Press, Princeton and Oxford, 2009.

[78] D. Bertsimas and M. Sim. The price of robustness. Operations Research, 52(1):35{53, 2004.

[79] A. Atamturk. Strong formulations of robust mixed 0-1 programming. Math. Program., 108(2):235-250, September 2006.

[80] K.-S. Goetzmann, S. Stiller, and C. Telha. Optimization over integers with robustness in cost and few constraints. In R. Solis-Oba andG. Persiano, editors, Approximation and Online Algorithms (WAOA 2011), volume 7164 of Lecture Notes in Computer Science, pages 89-101. Springer Berlin / Heidelberg, 2012.

[81] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. Math. Programming A, 99:351-376, 2003.

[82] D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. Automatic Control, IEEE Transactions on, 55(12):2751-2766, 2010.

[83] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenios. Robust optimization of large-scale systems. Operations Research, 43(2):264{281, 1995.

[84] A. Schobel. Generalized light robustness and the trade-obetween robustness and nominal quality. Mathematical Methods of Operations Research, 80(2):161-191, 2014.

[85] M. Fischetti and M. Monaci. Light robustness. In R. K. Ahuja, R.H. M□ ohring, and C.D. Zaroliagis, editors, Robust and online large-scale optimization, volume 5868 of Lecture Note on Computer Science, pages 61-84. Springer, 2008.

[86] M. Fischetti, D. Salvagnin, A. Zanette. Fast approaches to improve the robustness of a railway timetable. Transp Sci 43:321–335, 2009.

[87] M. Goerigk , M. Knoth, M. Müller-Hannemann, M. Schmidt.A. Schöbel. The price of strict and light robustness in timetable information. Transp Sci 48:225–242, 2014.

[88] M. Goerigk,A. Schöbel. A scenario-based approach for robust linear optimization. In: Proceedings of the 1st international ICST conference on practice and theory of algorithms in (computer) systems (TAPAS), lecture notes in computer science. Springer, Berlin, pp 139–150, 2011.

[89] M. Goerigk, A. Schöbel. An empirical analysis of robustness concepts for timetabling. In: Erlebach T, Lübbecke M (eds) Proceedings of ATMOS10, volume 14 of OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik Dagstuhl, Germany, pp 100–113, 2010.

[90] K. Klamroth, E. Köbis, A. Schöbel, C. Tammer. A unified approach for different concepts of robustness and stochastic programming via nonlinear scalarizing functionals. Optimization 62(5):649–671, 2013.

[91] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and A. Navarra. Robust Algorithms and Price of Robustness in Shunting Problems. In Proc. of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS07), 2007.

[92] S. Stiller. Extending concepts of reliability. Network creation games, real-time scheduling, and robust optimization. PhD thesis, TU Berlin, 2008.

[93] C. Liebchen, M. Lubbecke, R. H. Mohring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja, R.H. M□ ohring, and C.D. Zaroliagis, editors, Robust and online large-scale optimization, volume 5868 of Lecture Note on Computer Science, pages 1-27. Springer, 2009.

[94] G. D'Angelo, G. Di Stefano, and A. Navarra. Recoverable-robust timetables for trains on single-line corridors. In Proceedings of the 3rd International Seminar on Railway Operations Modelling and Analysis - Engineering and Optimisation Approaches (RailZurich2009), 2009.

[95] A.L. Erera, J.C. Morales, and M. Svalesbergh. Robust optimization for empty repositioning problems. Operations Research, 57(2):468-483, 2009.

[96] M. Goerigk. Algorithms and Concepts for Robust Optimization. PhD thesis, Georg-August Universit at G ottingen, 2012.

[97] M. Goerigk and A. Schobel. Recovery-to-optimality: A new two-stage approach to robustness with an application to aperiodic timetabling. Computers & Operations Research, 52, Part A(0):1 - 15, 2014.

[98] H. Aissi, C. Bazgan, and D. Vanderpooten. Min{max and min{max regret versions of combinatorial optimization problems: A survey. European Journal of Operational Research, 197(2):427 - 438, 2009.

[99] P. Kouvelis and G. Yu. Robust Discrete Optimization and Its Applications. Kluwer Academic Publishers, 1997.

[100] R. Kalai, C. Lamboray, and D. Vanderpooten. Lexicographic -robustness: An alternative to min{max criteria. European Journal of Operational Research, 220(3):722 - 728, 2012.

[101] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. Math. Programming A, 88:411-424, 2000.

[102] A. Ben-Tal, D. Bertsimas, and D. B. Brown. A soft robust model for optimization under ambiguity. Operations Research, 58(4-Part-2):1220- 1234, July 2010

[103] A. Ben-Tal, S. Boyd, and A. Nemirovski. Extending scope of robust optimization: Comprehensive robust counterparts of uncertain problems. Mathematical Programming, 107(1-2):63-89, 2006.

[104] N. Eggenberg, M. Salani, and M. Bierlaire. Uncertainty feature optimization: An implicit paradigm for problems with noisy data. Networks, 57(3):270-284, 2011.

[105] ARR: Arrival project under contract no. FP6-021235-2. http://arrival.cti.gr/index.php/Main /HomePage.

[106] P.C. Bouman, J.M. Akker, and J.A. van den Hoogeveen. Recoverable robustness by column generation. In European Symposium on Algorithms, volume 6942 of Lecture Notes in Computer Science, pages 215- 226. Springer, 2011.

[107] V.A. Caprara, L. Galli, L. Kroon, G. Maroti, and P. Toth. Railway rolling stock planning: Robustness against large disruptions. Transportation Science, 46(2):217-232, 2012.

[108] F. Manseur, N. Farhi, H. Haj-Salem, J-P. Lebacque. An algorithm for robust routing strategies in networks. Journal of Traffic and Transportation Engineering 5 (2017) 8-20. doi : 10.17265/2328-2142/2017.01.002

[109] F. Manseur, N. Farhi, H. Haj-Salem, J-P. Lebacque. Robust adaptive strategies for the guidance of users in road networks. Transportation Research Procedia. 22 (2017) 645-654. doi : 10.1016/j.trpro.2017.03.060

[110] N. Farhi, H. Haj.Salem, J.P. Lebacque. On the robust guidance of users in road traffic networks. AIP Conference Proceedings, (2014).

[111] E. W. Dijkstra. A note on two problems in connection with graphs. Numerical Mathematics 1, (1959) 269-271.

[112] S.M. LaValle. Planning algorithms. Cambridge University Press (2006)

[113] L. Fu. An adaptive routing algorithm for in-vehicle route guidance systems with real-time information.Transp. Res. B Methodol. 35, 749–765 (2001). doi:10.1016/S0191-2615(00)00019-9

[114] L. Fu, L. Rilett. Expected shortest paths in dynamic and stochastic traffic networks. Transp. Res. BMethodol.32 (7), 499–516 (1998). doi:10.1016/S0191-2615(98)00016-2

[115] S. Gao, I. Chabini. Optimal routing policy problems in stochastic time-dependent networks. Transp.Res. B Methodol.40 (2), 93–122 (2006). doi:10.1016/j.trb.2005.02.001

[116] S. Gao. Real-time traveler information for optimal adaptive routing in stochastic time-dependent net-works. Transp. Res. Part C: Emerging Technologies21, 196–213 (2012). doi:10.1016/j.trc.2011.09.007

[117] Abraham, I., Fiat, A., Goldberg, A. V., and Werneck, R. F. Highway Dimension, Shortest Paths, and Provably Efficient Algorithms. In Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA10), Society for Industrial and Applied Mathematics (2010).

[118] E. Miller-Hooks. Adaptive least-expected time paths in stochastic, time-varying transportation and datanetworks. Networks 37 (1), 35–52, 2001.

[119] E. Miller-Hooks, H.S. Mahmassani. Least expected time paths in stochastic, time-varying transportation networks. Transplant. Sci. 34 (2), 198–215, 2000.

[120] H. Huang, S. Gao. Optimal paths in dynamic networks with dependent random link travel times. Transp.Res. B Methodol. 46, 579–598, 2012. doi:10.1016/j.trb.2012.01.005

[121] R.P. Louis. Optimal paths in graphs with stochastic or multidimensional weights. Commun. ACM26 (9), 670–676, 1983. doi:10.1145/358172.358406

[122] I. Murthy, S. Sarkar. Stochastic shortest path problems with piecewise-linear concave utility functions.Manag. Sci.44 (11-part-2), S125–S136, 1998. doi:10.1287/mnsc.44.11.S125

[123] Y. Fan, R. Kalaba, J.E.I. Moore. Arriving on time. J. Optim. Theory Appl.127(3), 497–513 , 2005.doi:10.1007/s10957-005-7498-5

[124] Y. Fan, Y. Nie. Optimal routing for maximizing the travel time reliability. Networks and Spatial Economics6 (3-4), 333–344 (2006). doi:10.1007/s11067-006-9287-6

[125] Flajolet, A., Blandin, S., and Jaillet, P. Robust Adaptive Routing Under Uncertainty. arXiv preprint arXiv:1408.3374 (2014)

[126] S. Lim, C. Sommer, E. Nikolova, D. Rus, D. Practical route planning under delay uncertainty: Stochastic shortest path queries. In: Proceedings of Robotics: Science and Systems. Sydney, Australia, 2012.

[127] Y. Nie, Y. Fan. Arriving-on-time problem: discrete algorithm that ensures convergence. Transp. Res.Record: J. Transp. Res. Board1964, 193–200, 2006.

[128] E. Nikolova. Approximation algorithms for reliable stochastic combinatorial optimization. In: LectureNotes in Computer Science, vol. 6302, pp. 338–351, 2010. doi:10.1007/978-3-642-15369-326

[129] S. Samaranayake, S. Blandin, A. Bayen. A tractable class of algorithms for reliable routing in stochastic networks. Transportation Research Part C: Emerging Technologies 20 (1), 199–217, 2012. doi:10.1016/j.trc.2011.05.009

[130] A. Chen, Z. Ji. Path finding under uncertainty. J. Adv. Transp. 39 (1), 19–37, 2005.

[131] Y. Nie, X. Wu. Shortest path problem considering on-time arrival probability. Transp. Res. BMethodol.43 (6), 597–613, 2009. doi:10.1016/j.trb.2009.01.008

[132] Y. Nie, X. Wu. Dillenburg, J.F., Nelson, and P.C.: Reliable route guidance: A case study from Chicago.Transp. Res. A Policy Pract.46 (2), 403–419, 2012. doi:10.1016/j.tra.2011.10.008

[133] Y. Pan, L. Sun, M. Ge. Finding reliable shortest path in stochastic time-dependent network. Procedia - Social and Behavioral Sciences 96, 451–460, 2013. doi:10.1016/j.sbspro.2013.08.053

[134] Y. Chen, W.H. Lam, A. Sumalee, Z. Li. Reliable shortest path finding in stochastic net-works with spatial correlated link travel times. Int. J. Geogr. Inf. Sci.26 (2), 365-386, 2012.doi:10.1080/13658816.2011.598133.

[135] A. Zockaie, Y. Nie, H. Mahmassani. Simulation-based method for finding minimum travel time budget paths in stochastic networks with correlated link times. Transp. Res. Record: J. Transp. Res. Board2467,140–148, 2014. doi:10.3141/2467-15

[136] Bertsekas, D. P. Dynamic Programming and Optimal Control. Athena Scientific, 2005.

[137] S. French, D.R. Insua. Statistical decision theory, Kendall's Library of Statistics, vol. 9. Wiley, 2000.

[138] J. Neumann, O. Morgenstern. Theory of games and economic behavior. Princeton University Press, Princeton, 1947.

[139] S. Samaranayake, S. Blandin, A. Bayen. Speedup techniques for the stochastic on-time arrival problem. In OASIcs-OpenAccess Series in Informatics, volume 25. Schloss Dagstuhl- Leibniz-Zentrum fuer Informatik, 2012.

[140] B. C. Dean. Speeding up stochastic dynamic programming with zero-delay convolution. Algorithmic Operations Research, 5(2), 2010.

[141] G. Sabran, S. Samaranayake, A. Bayen. Precomputation techniques for the stochastic on-time arrival problem. In ALENEX, SIAM, 138-146, 2014.

[142] M. Kobitzsch, S. Samaranayake, D. Schieferdecker. Pruning techniques for the stochastic on-time arrival problem, An experimental study. Computer sciences, data structures and algorithm, 2014.

[143] O.A.Smith, S.Tubbs. Abivariate gamma probability distribution with application to gust modelling. NASA Technical Memorandum-82483, 1982.

[144] B.C.Dean. Algorithms for minimum-cost paths in time-dependent networks with waiting policies. Networks 44,41-46, 2004.

[145] B.C.Dean. Shortest paths in FIFO time-dependent networks: Theory and algorithms. Rapport technique, Massachusetts Institute of Technology, 2004.

[146] V. Astarita. A continuous time link model for dynamic network loading based on travel time function. In: 13th International Symposium on Transportation and Traffic Theory. Lyon, France, pp. 79–102, 1996.

[147] sumo.dlr.de. SUMO

[148] http://www.bgu.ac.il/~bargera/tntp/. Transportation network test problems

[149] TraCI4Matlab.http://www.mathworks.com/matlabcentral/fileexchange/44805-traci4matlab

[150] TraCI4J. https://github.com/egueli/TraCI4J

[151] http://www.sumo.dlr.de/pydoc/traci.html

[152] M. Behrisch, D. Krajzewicz, P. Wagner, and Yun-Pang Wang. Comparison of Methods for Increasing the Performance of a DUA Computation, in Proc. of DTA2008 International Symposium on Dynamic Traffic Assignment, No. EPFL-CONF-154987, Jun. 2008.

[153] D. Krajzewicz, M. Behrisch and Y. Wang. Comparing performance and quality of traffic assignment techniques for microscopic road traffic simulations, in Proc. ofDTA2008 International Symposium on Dynamic Traffic Assignment No. EPFL-CONF-154987, Jun. 2008