

Université
de Toulouse

THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Institut National des Sciences Appliquées de Toulouse (INSA Toulouse)

Discipline ou spécialité :
Systèmes industriels

Présentée et soutenue par :
Fallou GUEYE

le : mardi 14 décembre 2010

Titre :

Algorithmes de recherche d'itinéraires en transport multimodal

JURY

Aziz MOUKRIM, Professeur, Université Technologique de Compiègne
Emmanuel NERON, Professeur, Université de Tours
Roberto WOLFLER-CALVO, Professeur, Université Paris 13
Frédéric SCHETTINI, Directeur de la société MobiGIS

Ecole doctorale :
Systèmes (EDSYS)

Unité de recherche :
LAAS-CNRS

Directeur(s) de Thèse :
Christian Artigues LAAS-CNRS
Marie José Huguet LAAS-CNRS

Rapporteurs :
Emmanuel NERON, Professeur, Université de Tours
Roberto WOLFLER-CALVO, Professeur, Université Paris 13



THESE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE TOULOUSE

délivré par l'institut national des sciences appliquées de Toulouse

ECOLE DOCTORALE SYSTEMES

Discipline : Systèmes industriels

Présenté par

Fallou GUEYE

Algorithmes de recherche d'itinéraires en transport multimodal

Directeur de thèse : **M. Christian Artigues**

Directrice de thèse : **Mme Marie José Huguet**

Soutenue le XXXXXXXX
Devant la Commission d'Examen

JURY

M. Emmanuel NERON	Professeur d'Université	Rapporteur
M. Roberto WOLFLER-CALVO	Professeur d'Université	Rapporteur
M. Aziz MOUKRIM	Professeur d'Université	Examineur
M. Frédéric SCHETTINI	PDG MobiGIS	Examineur
M. Christian ARTIGUES	Chargé de Recherche	Directeur de thèse
Mme Marie José HUGUET	Maitre de Conférences	Directeur de thèse

Table des matières

Table des figures	7
Liste des tableaux	11
Remerciement	13
Résumé	16
Introduction générale	1
I Modèles et algorithmes de recherche d'itinéraire en transport multimodal	7
1 Introduction	7
2 Les graphes	7
3 Modélisation des réseaux de transport	9
3.1 Réseaux de transport monomodaux	9
3.1.1 Cas des réseaux statiques	9
3.1.2 Cas des réseaux dépendants du temps	10
Modèle par tables horaires	10
Modèle espace-temps	11
3.2 Réseaux de transport multimodaux	12
3.2.1 Modélisation par couche	12
3.2.2 Modélisation par hypergraphe	14
4 Calcul d'itinéraires multimodaux	15
4.1 Position du problème	15

Table des matières

4.2	Différents problèmes de plus courts chemins	17
4.3	Plus courts chemins sur un graphe monomodal	17
4.3.1	Plus courts chemins sur des graphes statiques (Shortest Path Problem ou SPP)	17
	Algorithme de Dijkstra	18
	Algorithme A^*	19
	Algorithme bidirectionnel	21
	Méthodes de prétraitement : les landmarks	22
	Autres méthodes	24
	Remarque : calcul d'un distancier	25
4.3.2	Plus courts chemins dépendant du temps (Time-Dependant SPP ou TD-SPP)	25
4.4	Optimisation Combinatoire Multi-Objectifs : MOCO	27
4.4.1	Définitions	27
4.4.2	Les méthodes de résolution des problèmes d'optimisation multi-objectif	28
	Méthode par agrégation	28
	Méthodes ε -contraintes	30
	Méthodes de programmation par but	30
4.5	Plus courts chemins multi-objectifs : MOSPP	31
4.5.1	Notations	31
4.5.2	Formulation du problème	31
5	Plus courts chemins multimodaux : MM-SPP	32
5.1	Plus courts chemins multimodaux viables : V-MM-SPP	32
5.2	Plus courts chemins multimodaux et/ou multi-objectifs	35
6	Conclusion	36
II Modélisation et méthodes de résolution proposées		39
1	Introduction	39
2	Le problème étudié : définition informelle	40
3	Modèles du problème : graphe multi-couche et graphe d'états	42
3.1	Modélisation du réseau de transport	42
3.2	Modélisation de la viabilité	45
3.3	Définition du problème	46
4	Chemins partiels et règles de dominance	47

4.1	Étiquettes pour la représentation d'un chemin	47
4.2	Règle de dominance	48
4.3	Simplification de l'automate de viabilité	49
5	Algorithmes proposés pour le BI-MM-V-SPP	50
5.1	Algorithme de Dijkstra MultiModal DIJ-MM	50
5.1.1	Description de l'algorithme	50
5.1.2	Complexité de l'algorithme	52
5.1.3	Discussion et limites de l'algorithme	52
5.2	Algorithme TLS (Topological Label Setting) Multimodal	52
5.2.1	Description de l'algorithme TLS	53
5.2.2	Complexité de l'algorithme TLS	54
5.2.3	Exemple illustratif	56
5.2.4	Discussion et limites de l'algorithme	57
5.3	Algorithme Multi-labels-Multi-Heaps : MLMH	58
5.3.1	Description de l'algorithme MLMH	58
5.3.2	Complexité de l'algorithme MLMH	59
5.3.3	Exemple de fonctionnement de l'algorithme	60
5.4	Algorithme bidirectionnel basé sur MLMH : FB-MLMH	61
5.4.1	Modélisation de l'automate de viabilité pour la phase de re- cherche arrière	62
5.4.2	Connexion des labels avant et arrière	63
5.4.3	Description de l'algorithme	65
5.4.4	Exemple de fonctionnement de l'algorithme	66
6	Variantes A^* des différents algorithmes	70
7	Algorithmes proposés pour le problème BI-MM-V-TD-SPP	72
8	Conclusion	73

III Résultats expérimentaux 75

1	Introduction	75
2	Contexte général d'expérimentation	75
3	Réseaux de transport de tests	76
3.1	L'environnement des tests	77
3.2	Métriques de performance	78
4	Résultats des expérimentations dans le cas non dépendant du temps	78
4.1	Résultats des expérimentations sans A^*	79

Table des matières

4.1.1	Analyse des résultats en termes de temps de calcul et de nombre de labels	80
4.1.2	Analyse des résultats en termes de règles de dominance	81
4.2	Résultats des expérimentations avec A^*	84
4.2.1	Analyse des résultats en termes de temps de calcul et de nombre de labels explorés	84
4.2.2	Analyse des résultats en termes de dominance	85
	Comparaison des variantes avec et sans A^*	88
4.3	Conclusion	90
5	Résultats des expérimentations dépendant du temps	91
5.1	Résultats sans A^*	91
5.1.1	Analyse des résultats	92
5.2	Résultats des expérimentations avec A^*	94
5.3	Conclusion	98
6	Conclusion	98
IV Travaux d'Industrialisation et logiciel développé		101
1	Introduction	101
2	Objectifs du logiciel	102
3	Principales fonctionnalités	103
3.1	Fonction de modélisation de l'offre de transport dans une base de données géo-spatiale	103
3.2	Fonctions d'analyses spatiales et multimodales	104
4	Conception du logiciel	105
4.1	Architecture de l'application	105
4.1.1	Composant d'accès aux données et administration du logiciel	105
4.1.2	Composants applicatifs	106
4.1.3	Interface utilisateur	106
4.2	Diagramme des classes	107
4.3	Description des classes	107
4.3.1	Solvers	107
4.3.2	StateManager	109
4.3.3	Graph	109
5	Paramétrages du logiciel	109
6	Modélisation du réseau de transport multimodal	109

6.1	Modélisation des modes de transport	109
6.2	Création du réseau multimodal	111
6.2.1	les données du réseau TC	111
6.2.2	Création des autres couches du réseau multimodal	113
6.2.3	Récapitulatif des éléments du réseau multimodal	114
6.2.4	Le graphe de la geo-database	115
7	Exemples d'utilisation	117
7.1	Visualisation d'un réseau TC	117
7.2	Recherche d'itinéraires et matrices OD	118
7.3	Calcul d'accessibilité ou isochrones	121
8	Conclusion	125
 Conclusion générale et perspectives		127
 Bibliographie		131
 Annexe A		141
A	Réseaux de transports	141
A.1	Réseau de transport non dépendant du temps	141
A.2	Réseau de transport dépendant du temps	142
 Annexe B		143
B	Résultats des expérimentations sur le réseau non dépendant du temps	143
B.1	Résultats des performances des algorithmes	143
B.1.1	Résultats DIJ-MM	143
B.1.2	Résultats A*-DIJ-MM	144
B.1.3	Résultats TLS	144
B.1.4	Résultats A*-TLS	145
B.1.5	Résultats MLMH	145
B.1.6	Résultats A*-MLMH	146
B.1.7	Résultats FB-MLMH	146
B.1.8	Résultats A*-FB-MLMH	147
B.1.9	Résultats FB-MLMH-ND	147
B.1.10	Résultats A*-FB-MLMH	148

Table des matières

Annexe C	149
C Schéma TRIDENT UML	149

Table des figures

I.1	Exemple de graphe statique	10
I.2	Illustration du modèle par tables horaires pour un graphe dynamique	11
I.3	Exemple d'un modèle espace-temps	13
I.4	Exemple de graphes dépendants du temps FIFO et non-FIFO	14
I.5	Modélisation par couches d'un réseau de transport multimodal.	15
I.6	Exemple d'hypergraphe	15
I.7	Isochrones : points accessibles en 15 minutes des stations de métro de Toulouse	16
I.8	Itinéraires multimodaux	17
I.9	Comparaison des espaces de recherche entre les algorithmes Dijkstra avec et sans A^* [43]	20
I.10	Espaces de recherche des algorithmes Dijkstra et bidirectionnel	22
I.11	Front de Pareto.	29
I.12	Description des états d'un chemin viable	34
I.13	Modélisation de la viabilité par un graphe d'états (Lozano et Storchi) [62] . .	34
II.1	Outil d'aide à la décision de calcul d'itinéraires multimodaux.	40
II.2	Exemple d'itinéraires multimodaux.	41
II.3	Exemple de réseau multimodal.	45
II.4	Automate de viabilité	47
II.5	Automate de viabilité réduit par dominance	49
II.6	Courbe des solutions de l'algorithme TLS	53
II.7	Exemple applicatif	56
II.8	Courbe des solutions de l'algorithme MLMH	59

Table des figures

II.9	Graphe d'états non déterministe et déterministe pour la recherche arrière . . .	63
III.1	Comparaison des méthodes et des règles de dominances en termes de temps CPU	81
III.2	Comparaison des méthodes et règles de dominances en termes de nombre de nœuds empilés	82
III.3	Comparaison des méthodes et des règles de dominances en termes de nombre de nœuds dépilés	83
III.4	Comparaison des méthodes et des règles de dominances en termes de nombre de nœuds visités	83
III.5	Comparaison des méthodes et des règles de dominances en termes de temps CPU	86
III.6	Comparaison des méthodes et des règles de dominances en termes de nombre de nœuds empilés	86
III.7	Comparaison des méthodes et des règles de dominances en termes de nombre de nœuds dépilés	87
III.8	Comparaison des méthodes et des règles de dominances en termes de nombre de nœuds visités	88
III.9	Comparaison des variantes avec ou sans A^* en termes de temps CPU (sans dominance)	89
III.10	Comparaison des variantes avec ou sans A^* en termes de temps CPU (Dominance de base)	89
III.11	Comparaison des variantes avec ou sans A^* en termes de temps CPU (Dominance états)	90
III.12	Comparaison des méthodes en termes de temps CPU	92
III.13	Comparaison des méthodes en termes de nombre de nœuds empilés	93
III.14	Comparaison des méthodes en termes de nombre de nœuds dépilés	93
III.15	Comparaison des méthodes en termes de nombre de nœuds visités	94
III.16	Comparaison des méthodes en termes de temps CPU	95
III.17	Comparaison des méthodes en termes de nombre de nœuds empilés	96
III.18	Comparaison des méthodes en termes de nombre de nœuds dépilés	96
III.19	Comparaison des méthodes en termes de nombre de nœuds visités	97
III.20	Comparaison des variantes avec ou sans A^* dépendant du temps en termes de temps CPU (Dominance états)	97
IV.1	Étude de marché du logiciel	104

IV.2	Architecture de l'application	106
IV.3	Diagramme de classes	108
IV.4	Interface utilisateur	110
IV.5	Réseau de transport multimodal	111
IV.6	tronçons de bus (tronçons bleus et rouges) non connectés à la voirie (en gris) .	113
IV.7	connexion du mode bus à la voirie	113
IV.8	Représentation d'une station de métro avec parking relais	114
IV.9	Les sources du graphe logique multimodal (ArcGIS)	116
IV.10	Les connexions pour définir le réseau multimodal (ArcGIS)	116
IV.11	Les attributs du graphe logique multimodal (ArcGIS)	117
IV.12	Visualisation d'un réseau TC	118
IV.13	Zoom de l'offre TC avec horaires à un arrêt	118
IV.14	Représentation cartographique d'une ligne de bus	119
IV.15	Nombre de passages par arrêt	119
IV.16	Visualisation d'itinéraires multimodaux	120
IV.17	Calcul d'itinéraire multimodal - Résultats proposés à l'utilisateur	120
IV.18	Calcul de matrices Origines/Destinations : Impédance = temps de marche . .	121
IV.19	Calcul de matrices Origines/Destinations : Impédance = temps en TC	121
IV.20	Accessibilité multi-seuils	122
IV.21	Accessibilité en 10 minutes selon différents modes	123
IV.22	Accessibilité piétonne 10 et 20 mn - Représentation sous forme de tronçons parcourus	123
IV.23	Points d'intérêts situés dans une zone d'accessibilité	124
A.1	Réseau de transport multimodal avec vitesse moyenne	141
A.2	Réseau de transport multimodal avec horaires	142
A.3	Schéma UML TRIDENT (line type)	150

Table des figures

Liste des tableaux

II.1	Compatibilité entre états pour la connexion en recherche avant	64
II.2	Compatibilité entre états pour la connexion en recherche arrière	65
III.1	Caractéristiques du réseau statique	76
III.2	Données du réseau dynamique	77
III.3	Résultats des algorithmes (sans règle de dominance)	79
III.4	Résultats des algorithmes (avec la règle de dominance de base)	79
III.5	Résultats des algorithmes (avec la règle de dominance entre états)	79
III.6	Résultats des algorithmes A^* (sans règle de dominance)	84
III.7	Résultats des algorithmes A^* (avec la règle de dominance de base)	84
III.8	Résultats des algorithmes A^* (avec la règle de dominance entre états)	84
III.9	Résultats des algorithmes (dominance basée sur les états)	91
III.10	Résultats des algorithmes A^* (dominance basée sur les états)	94
IV.1	Récapitulatif des éléments constituant le réseau	115
A.1	Résultats de l'algorithme DIJ-MM	143
A.2	Résultats de l'algorithme A^* -DIJ-MM	144
A.3	Résultats de l'algorithme TLS	144
A.4	Résultats	145
A.5	Résultats de l'algorithme MLMH	145
A.6	Résultats de l'algorithme A^* -MLMH	146
A.7	Résultats de l'algorithme FB-MLMH	146
A.8	Résultats de l'algorithme A^* -FB-MLMH	147

Liste des tableaux

A.9	Résultats de l'algorithme FB-MLMH-ND	147
A.10	Résultats de l'algorithme A^* -FB-MLMH-ND	148

Remerciement

Cette thèse est l'aboutissement des travaux élaborés dans le cadre d'un contrat CIFRE avec la société MobiGIS et l'équipe MOGISA du LAAS-CNRS.

Je tiens à exprimer ici ma profonde reconnaissance à tous ceux qui de près ou de loin m'ont permis de mener à bien ce travail.

Je tiens d'abord à remercier chaleureusement Christian Artigues et Marie José Huguet mes directeurs de thèse pour m'avoir encadré durant ces trois années, pour l'échange scientifique fructueux que nous avons entretenu et pour leur soutien indéfectible, pour m'avoir suivi en tant qu'étudiant, pour m'avoir sensibilisé à la recherche et pour les travaux et réalisations que nous avons conduits ensemble avant et durant la thèse.

Je leur remercie aussi pour leur énorme soutien moral et scientifique qu'ils m'ont généreusement accordé durant ces trois années de thèse mais aussi pour leur grande qualité humaine.

Je tiens ensuite à remercier Frédéric Schettini et Laurent Dezou directeurs de la société MobiGIS pour m'avoir donné l'opportunité de réaliser ces travaux et aussi pour leur encadrement. Je remercie l'ensemble du personnel de la société MobiGIS et particulièrement Cyril pour son apport remarquable dans le domaine des SIG.

Mes sincères remerciements vont aussi au Professeur Aziz Mouckrim, Professeur à l'université technologie de Compiègne (UTC) pour l'honneur qu'il m'a fait en acceptant de présider ce jury.

Remerciement

J'exprime aussi toute ma reconnaissance aux :

- M. Emmanuel NERON, professeur à l'Université de Tours
- M. Roberto Wolfler Calvo, Professeur à l'Université paris 13

de m'avoir fait honneur en consacrant un temps important à la lecture de mon mémoire et dont les remarques m'ont permis d'améliorer sa qualité, et en acceptant de le rapporter.

Je souhaite aussi témoigner ma gratitude à toute l'équipe de MOGISA, à mes collègues pour les moments de bonheur et de labeur qu'on a passés ensemble.

Enfin, une pensée très particulière et une reconnaissance infinie :

- A mes parents, mes frères et sœurs, ma femme Amy pour leur amour, leur patience et leurs sacrifices.
- A mes amis pour leurs encouragements et pour leur soutien.

Résumé

Ce travail de thèse s'est intéressé au transport urbain de passagers dans un contexte d'offre de transport multimodale consistant en la coexistence de plusieurs modes de transport.

Dans la pratique, un problème de transport multimodal nécessite la prise en compte de plusieurs objectifs et de contraintes spécifiques liées aux modes ou à la séquence de modes utilisés. De telles contraintes sont appelées **contraintes de viabilité**.

Cette thèse CIFRE s'est déroulée en collaboration avec la société MobiGIS, spécialisée dans le conseil et le développement d'applications autour des Systèmes d'Information Géographiques.

Le problème étudié dans cette thèse est celui de la **recherche d'itinéraires viables multimodaux point à point bi-objectif** pour lequel il s'agit à la fois de minimiser le temps de trajet et le nombre de changements de mode. Compte tenu notamment des objectifs considérés, ce problème est de complexité polynomiale.

Sur la base d'une **modélisation multi-couches** des réseaux de transport multimodaux et d'une **modélisation par un automate à états finis** des contraintes de viabilité nous avons proposé différents algorithmes de résolution de ce problème basés sur le principe de fixation et extension de labels. Nous avons également proposé une règle de **dominance basée** sur les états de l'automate de viabilité et permettant d'élaguer le nombre de labels explorés par nos algorithmes.

Des adaptations en **bidirectionnel** ou en utilisant le principe de la recherche A^* ont également été proposées.

Les algorithmes proposés ont été évalués sur une partie du réseau de transport de la ville de Toulouse et les expérimentations ont mis en évidence l'intérêt de la règle de dominance basée sur les états ainsi que de l'approche bidirectionnelle développée.

Un prototype logiciel implémentant différentes fonctionnalités des algorithmes de plus courts chemins a été développé. Il permet notamment de réaliser des calculs d'itinéraires point à point, des calculs d'accessibilité ou des calculs de distancier.

Mots clés : plus court chemin - chemins viables bi-objectif - transport multimodal

Introduction générale

Le secteur des transports a connu ces dernières années un accroissement fort du trafic. Il a l'obligation de réduire les impacts environnementaux qu'il engendre tels que la pollution, la consommation énergétique et la congestion automobile, etc. Cela se traduit par le développement de véhicules moins polluants mais aussi par le développement de modes de transport alternatifs (transport collectif, ferroviaire, fluvial etc.). Depuis quelques années, la multi-modalité reflète une des principales évolutions du transport urbain ou inter-urbain de biens ou de personnes. La multimodalité des réseaux de transport revêt plusieurs facettes : tout d'abord une facette de conception de ces réseaux multimodaux pour les autorités en charge du développement de politiques de transport, puis une facette d'exploitation pour les opérateurs de transport et enfin une facette d'utilisation pour les usagers de réseaux multimodaux.

Nos travaux de recherche se situent au niveau des phases d'exploitation et d'utilisation des réseaux de transport multimodaux : pour des exploitants de réseaux il s'agit de disposer d'outils d'analyse de l'offre multimodale de transport afin de mieux cerner ses caractéristiques et, pour des passagers ou des entreprises de transport, il s'agit de disposer de services de calcul d'itinéraires complets et performants afin d'organiser au mieux leurs déplacements. Nous nous situons dans ce contexte de développement de modes de transport alternatifs et nous nous intéressons plus spécifiquement au transport urbain et péri-urbain de passagers.

Un mode de transport désigne une forme particulière de transport qui se distingue principalement par le véhicule utilisé et donc par l'infrastructure qu'il met en œuvre. On peut ainsi caractériser différents modes de transport :

- les modes collectifs ou transport en commun correspondent à des moyens de transport effectués à plusieurs. Les modes collectifs urbains les plus fréquents sont le bus, le tramway

Introduction générale

- ou le métro, ainsi que dans certaines agglomérations : le trolleybus, le transport à la demande (TAD), le train ou le bateau ;
- les modes individuels motorisés tels que les véhicules personnels, les taxis, les deux-roues à moteur ou le covoiturage ;
 - les modes individuels doux ou non polluant comme les vélos, les rollers, la marche à pied ou les trottinettes ;

Différentes définitions de la multimodalité ont été proposées par plusieurs organismes en relation avec le secteur des transports.

Le *GART*, Groupement des Autorités Responsables des Transport rassemblant des élus des collectivités locales ayant la charge de l'organisation des transports publics, propose sur son site internet [40] la définition suivante : « *L'intermodalité est un concept qui permet l'utilisation de plusieurs modes de transport au cours d'un même déplacement. Cette définition peut s'appliquer tant aux déplacements de personnes qu'au transport de marchandises. On emploie le terme de **multimodalité** pour envisager plusieurs déplacements ou chaînes de transports distincts empruntant chacun un mode ou une combinaison de modes différents. Le passage d'un mode à un autre s'appelle la rupture de charge. Pour désigner les lieux où l'on change de mode - dans le domaine du transport de passagers - on parle de pôles d'échanges, tandis que dans le fret on utilise la notion de plate-forme multimodale. L'intermodalité passe aussi par la mise en place d'outils relatifs à l'information multimodale ou à la billettique.* »

L'association ATEC/ITS-France, Association pour le développement des Transports de l'Environnement et de la Circulation/Intelligent Transport System réunissant des acteurs du transport (publics ou privés, syndicats, formation,) ayant comme objectif de promouvoir une exploitation durable des transports terrestres de biens ou de personnes, propose une définition assez similaire [3] : « *la multimodalité est l'offre de plusieurs moyens de transport pour un déplacement entre une origine et une destination. Elle se place donc en amont, et couvre une proposition faite au client dans laquelle chaque possibilité de choix peut être monomodale (un seul moyen à utiliser) ou intermodale (plusieurs moyens successifs à utiliser).* »

Ces définitions s'inscrivent dans un cadre de neutralité du choix des modes de transport et met en relief les notions d'individu (ou de trajet) multimodal et d'offre multimodale. La première notion est le recours à plusieurs modes de transport en fonction des circonstances et de la nature des déplacements. La seconde notion est l'infrastructure ou service permettant le choix d'un des modes de transport proposés ou de les combiner successivement.

Le CERTU (Centre d'Etude sur les Réseaux, les Transports, l'Urbanisme et les constructions publiques dont le rôle est de mener des études dans ses domaines de compétences pour

l'état ou les collectivités locales) a proposé une autre définition de la multimodalité dans trois rapports entre 2000 et 2003 [18, 69, 70] : « *La fonction essentielle d'un système d'information multimodale est de fournir à l'utilisateur des transports toute l'information nécessaire à la réalisation de son voyage. Cette information vise à réduire l'incertitude des usagers sur les itinéraires, les modes de déplacement envisageables, la durée et le coût de ces déplacements selon le mode utilisé, les ruptures de charge éventuelles, et si possible, à orienter le comportement des usagers au bénéfice d'une utilisation optimale des infrastructures et d'une priorité aux transports collectifs.* »

Ainsi, pour le CERTU, le développement de la multimodalité vise à donner la priorité au développement des transports collectifs au détriment des véhicules personnels et à améliorer la complémentarité des deux modes [31]. Cette définition de la multimodalité s'accorde avec les Plans de Déplacement Urbain (PDU) mis en place par les Autorités Organisatrices (AO) de transport suite à différentes lois (LOTI, LAURE, LOADDT, SRU) de 1982 à 2000. Les PDU ont pour objectifs un usage coordonné de tous les modes de déplacement, mais aussi la diminution du trafic automobile, le développement des transports collectifs et la mise en place de moyens de déplacements économes, comme vélo ou la marche à pied.

Pour synthétiser ces définitions de la multimodalité, nous considérons que la multimodalité correspond à un principe d'organisation et d'articulation de l'offre du transport visant à combiner plusieurs modes de transports pour la réalisation de déplacements permettant la mise en place d'alternatives à la voiture individuelle.

Les Systèmes d'Information Multimodale (SIM) servent à accompagner les voyageurs dans leurs déplacements. Ces systèmes peuvent proposer des services liés au transport (calcul d'itinéraires, consultation d'horaires, informations sur les perturbations et retards, tarifs des déplacements, disponibilités des places de parking, etc.) ainsi que des services connexes au transport (événements culturels, météo, informations touristiques). Les récentes évolutions technologiques de l'Internet et des Systèmes d'Information Géographique (SIG) ont fortement contribué à l'émergence de nouveaux services dans les domaines du transport de passagers et de la mobilité des personnes (navigateur embarqué dans les véhicules, sites Internet, etc.). Les SIG permettent par exemple de proposer aux usagers de nombreux services tels que la génération de cartes thématiques et interactives, la recherche et le positionnement d'adresses sur une carte et la visualisation d'itinéraires sur une carte.

On considèrera par la suite que l'information multimodale est l'information permettant de renseigner les usagers ou les exploitants d'un réseau de transport sur leurs déplacements.

Introduction générale

Elle présente plusieurs caractéristiques en termes de :

- mode de transport : transport en commun, transport individuel, parkings, etc. ;
- type d’information et de données : information temps réel, information théorique, informations sur les horaires, sur les tarifs, sur les itinéraires, etc. ;
- moment du déplacement : information avant ou pendant le trajet ;
- couverture géographique : suivant la longueur du trajet, l’information porte sur une agglomération, une région, etc. ;
- médias : personnel d’information, service téléphonique, service web, panneaux, etc.

La plupart des systèmes d’information pour des réseaux de transport existants sont soit monomodaux soit multimodaux mais ils ne concernent alors qu’un seul opérateur de transport (comme par exemple la RATP). Plusieurs régions et exploitants de transport en commun cherchent à développer de l’information multimodale sous différents supports : bornes interactives dans des lieux publics, panneaux d’affichage ou sites web (SNCF, RATP, LePilote à Marseille) qui deviennent de plus en plus répandus. Toutefois ces sites fournissent de l’information sur un seul réseau ou sur des agglomérations limitées (comme par exemple le système MobiTrans dans différentes agglomérations en France) et plus rarement sur tout un territoire, par exemple le système DELFI en Allemagne.

Dans un SIM, il est déterminant de pouvoir évaluer et proposer des informations élaborées pour aider les voyageurs à choisir le ou les modes de transport les plus appropriés par rapport à leurs besoins en déplacement.

Il est également déterminant, pour les exploitants des réseaux de transport de disposer d’un outil permettant de d’optimiser leurs réseaux de transport multimodal afin de mieux évaluer les besoins des clients mais aussi de mieux maîtriser leurs coûts d’exploitation.

Ainsi, nous nous intéressons à la définition d’outils en adéquation avec la volonté de mettre une place une politique d’optimisation de l’usage des transports alternatifs à la voiture individuelle permettant aux différents types d’usagers (particuliers, entreprises etc.) de rationaliser leurs choix en matière de déplacements compte tenu de données socio-économiques et environnementales. En considérant un réseau de transport multimodal, notre objectif est de réaliser un logiciel de résolution des problèmes de recherche d’itinéraires permettant ainsi de mettre en place ces divers outils d’analyse multimodale.

Ce travail a été réalisé dans le groupe MOGISA du LAAS-CNRS dans le cadre d’une thèse

CIFRE avec la société **MobiGIS**¹, jeune entreprise spécialisée dans les SIG et les Systèmes de Transport Intelligent.

Ce mémoire de thèse est articulé en quatre chapitres.

Le premier chapitre présente un état de l'art sur la modélisation et la recherche d'itinéraires dans des réseaux de transport. La première partie de ce chapitre énumère différentes méthodes de modélisation des réseaux de transports dans plusieurs cas : statiques ou dépendant du temps ainsi que monomodaux ou multimodaux.

La deuxième partie de ce chapitre expose les différents algorithmes existants pour résoudre des problèmes de recherche d'itinéraires dans les réseaux de transport en particulier les réseaux de transport dépendant du temps et multi-modaux.

Le deuxième chapitre est dédié à la présentation du problème considéré : calcul des plus courts chemins bi-objectifs viables multimodaux puis à la présentation de différents algorithmes que nous proposons pour sa résolution. Dans la première partie de ce chapitre, nous présentons la modélisation retenue pour ce problème. Les deux parties suivantes sont consacrées aux algorithmes de résolution : dans le cas statique pour la première partie et dans le cas dépendant du temps pour la seconde.

Le troisième chapitre est consacré à des évaluations comparatives des différents algorithmes proposés. Les tests sont effectués sur le réseau de transport multimodal de la ville de Toulouse. Les résultats sont présentés dans le cas statique puis dépendant du temps.

Le quatrième chapitre traite de l'industrialisation, au sein de la société MobiGIS, des travaux de recherche. Les différents algorithmes ont été évalués et testés dans un contexte industriel et l'un d'entre eux a été dans un logiciel d'aide à la décision pour l'analyse multimodale de réseaux de transport.

Enfin nous terminons ce mémoire en présentant une conclusion des travaux menés ainsi que des perspectives d'extension envisagées.

1. www.mobigis.fr

Chapitre I

Modèles et algorithmes de recherche d'itinéraire en transport multimodal

1 Introduction

La multimodalité peut apporter des réponses à nombre de préoccupations générées par le transport de voyageurs. Elle peut en effet aider à la gestion de la mobilité, contribuer à l'amélioration des conditions de circulation et à la protection de l'environnement. Pour ce faire, elle doit proposer des systèmes d'information multimodaux (SIM) efficaces. L'une des fonctionnalités les plus importantes des SIM pour l'utilisateur est le système d'aide aux déplacements.

Pour cela, un SIM doit au moins comporter un calculateur d'itinéraires efficace capable d'optimiser et de planifier un déplacement entre une origine O et une destination D dans un réseau de transport multimodal. Pour ce faire, il doit disposer d'une modélisation pertinente du réseau de transport. En effet, la modélisation se situe en amont de la résolution et influe directement sur les méthodes envisageables pour la résolution des problèmes de calcul d'itinéraires multimodaux.

Ainsi, dans ce chapitre, nous allons dresser un état de l'art sur les modélisations proposées des réseaux de transport, sur les différents problèmes de recherche d'itinéraires déjà étudiés ainsi que sur les algorithmes proposés pour les résoudre.

2 Les graphes

Nous allons rappeler brièvement quelques définitions sur les graphes qui sont utiles pour la suite de ce mémoire. Considérons tout d'abord le cas de graphe non orienté noté $G = (N, A)$ dans lequel N est l'ensemble des n nœuds ou sommets et A l'ensemble des m arêtes pouvant

Chapitre I. Modèles et algorithmes de recherche d'itinéraire en transport multimodal

être ou non valuées :

- On note $a = \{x_i, x_j\}$ est une arête de A d'extrémités x_i et x_j . Une arête dont les deux extrémités sont identiques est appelée boucle.
- L'ensemble des voisins d'un nœud x_i correspond à l'ensemble des nœuds x_j tels que $\{x_i, x_j\} \in A$. Un nœud isolé est un nœud sans aucun voisin.
- Le degré d'un sommet x_i , noté $d(x_i)$, est égal au nombre de voisins du nœud x_i lorsque le graphe ne comporte pas de boucle sur le nœud x_i . Sinon, il faut rajouter 1 au nombre de voisin pour obtenir le degré du nœud x_i ;
- Une chaîne est une suite finie non vide de nœuds de $G : C = (x_1, x_2, \dots, x_k)$ de telle sorte que $\forall i, 1 \leq i \leq k$, l'arête $\{x_i, x_{i+1}\} \in A$. La longueur d'une chaîne correspond à son nombre d'arêtes (qui ne sont pas forcément toutes distinctes).
- Un graphe non orienté est dit connexe s'il existe une chaîne entre toute paire de sommets. Dans le cas contraire, il comporte plusieurs composantes connexes.

Plaçons nous maintenant dans le cas de graphe orienté noté $G = (N, E)$ dans lequel N est l'ensemble des n nœuds ou sommets et E l'ensemble des m arcs pouvant être ou non valués :

- On note $e = (x_i, x_j)$ un arc de E ayant comme origine le sommet x_i et comme extrémité le sommet x_j . Un arc reliant un même sommet est une boucle.
- L'ensemble des successeurs d'un nœud x_i , noté $Succ(x_i)$ correspond à l'ensemble des nœuds x_j tels que l'arc $(x_i, x_j) \in E$. L'ensemble des arcs $(x_i, x_j) \in E$ est appelé ensemble des arcs sortants de x_i .
- L'ensemble des prédécesseurs d'un nœud x_i , noté $Pred(x_i)$ correspond à l'ensemble des nœuds x_j tels que l'arc $(x_j, x_i) \in E$. L'ensemble des arcs $(x_j, x_i) \in E$ est appelé ensemble des arcs entrants en x_i .
- Le degré entrant (respectivement le degré sortant) d'un sommet x_i , noté $d^-(i)$ (respectivement $d^+(i)$) est le nombre d'arcs entrants (respectivement sortant) en x_i . Le degré du sommet x_i , noté $d(x_i)$ est égal à $d^-(i) + d^+(i)$.
- Un chemin est une suite finie non vide de nœuds de $G : C = (x_1, x_2, \dots, x_k)$ de telle sorte que $\forall i, 1 \leq i \leq k$, l'arc $(x_i, x_{i+1}) \in A$. La longueur d'un chemin correspond à son nombre d'arcs (qui ne sont pas forcément tous distincts).
- Un graphe orienté est connexe si en ignorant son orientation, le graphe non orienté correspondant l'est.
- Un graphe orienté est fortement connexe s'il existe un chemin entre toute paire de sommets. Sinon il comporte plusieurs composantes fortement connexes.
- Le graphe G^b inverse d'un graphe valué $G = (E, N)$ est noté $G^{-1} = (N, E^{-1})$. L'ensemble des arcs E^{-1} correspond aux arcs de E en sens inverse.

- Un chemin (respectivement une chaîne) est dit simple lorsque les arcs (respectivement arêtes) qui le composent sont deux à deux distinct(e)s. Un chemin (respectivement une chaîne) est dit élémentaire lorsque les sommets qui le composent sont deux à deux distincts

Les représentations des réseaux de transport par des graphes diffèrent selon les caractéristiques des réseaux (monomodaux ou multimodaux, statiques ou dépendants du temps) mais aussi selon les problèmes que l'on cherche à résoudre. Nous allons dans la suite de ce chapitre présenter différents réseaux de transport et les modélisations qui ont été proposées pour ces réseaux.

3 Modélisation des réseaux de transport

3.1 Réseaux de transport monomodaux

3.1.1 Cas des réseaux statiques

On parle de réseau monomodal lorsque les arcs et nœuds du graphe sont banalisés : ils correspondent tous au même moyen de transport ou on ne souhaite pas distinguer les modes. Ainsi, dans le cas monomodal, le réseau de transport est habituellement représenté par un graphe orienté tel que présenté précédemment.

On parle de réseau statique lorsque les valuations associées aux arcs sont des constantes au cours du temps : si on note t_i la date d'arrivée au nœud i , la valuation de l'arc (i, j) , $d_{i,j}$ est une constante quelque soit t_i . Par exemple, le temps de parcours d'un trajet à pieds est toujours le même quelle que soit l'heure du début du parcours. La figure I.1 donne un exemple d'un graphe statique à 5 sommets dont les valuations sont des constantes

Dans le cadre de grands graphes, des modélisations basées sur le concept de réseaux hiérarchisés par niveaux ou par couches ont été proposées. La structure hiérarchique est un moyen efficace pour modéliser les réseaux par niveaux [64, 78]. Une modélisation hiérarchique HEPV (Hierarchical Encoded Path View) proposée par [54] consiste à partitionner le graphe en sous graphes hiérarchisés. Bielli et al. [12] définissent un réseau hiérarchique à deux niveaux : le réseau national qui relie les différentes villes et le réseau urbain.

Cette modélisation hiérarchique est également utilisée pour modéliser les réseaux de transports multimodaux (voir partie 3.2).

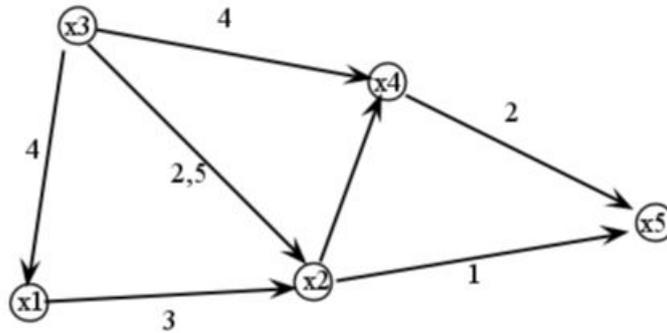


Figure I.1 – Exemple de graphe statique

3.1.2 Cas des réseaux dépendants du temps

On parle de réseau de transport dépendant du temps lorsque les valuations associées aux arcs, telles que le temps ou le coût, ont une valeur qui dépend du moment où l'arc est emprunté ce qui est fréquent dans les réseaux de transport. Ainsi pour tout arc (i, j) , la valuation $d_{i,j}$ dépend de la date de départ t_i du nœud i et est alors notée $d_{i,j}(t_i)$. Différents modèles ont été proposés dans la littérature pour traduire cette dynamique.

Modèle par tables horaires

Cette modélisation d'un réseau de transport dépendant du temps s'appuie sur un graphe orienté tel que défini précédemment dans lequel les valuations des arcs vont intégrer l'aspect dynamique. Ce graphe dynamique est noté par la suite $G(t) = (N, E, D(t))$. Dans un graphe dynamique, à chaque arc (i, j) on associe une liste de valuations correspondant aux différentes valeurs que peut prendre l'arc dans le temps.

On retrouve ce modèle de graphe dynamique dans les travaux de [19, 88] qui s'intéressent de manière plus spécifique au temps de trajet dépendant du temps et considèrent que les valuations sont constantes par morceaux. Ainsi, à chaque arc (i, j) on associe à une liste d'horaires de départs possibles notée $HD(i, j)$:

$$HD(i, j) = (hd_1(i, j), hd_2(i, j), \dots, hd_k(i, j), \dots, hd_{N_{i,j}}(i, j))$$

avec $hd_k(i, j)$ la k -ième date de départ associée à l'arc (i, j) et $N_{i,j}$ le nombre total de ses départs. A chaque date de départ $hd_k(i, j)$ permettant d'aller de i à j , on associe une valuation $d_{i,j}(hd_k(i, j))$. Pour toute date t_i de départ du nœud i , la valuation $d_{i,j}(t_i)$ est alors définie

comme suit :

$$d_{ij}(t_i) = \begin{cases} d_{ij}(hd_k(i, j)) & \text{si } t_i = hd_k(i, j) \\ hd_{k+1}(i, j) - t_i + d_{ij}(hd_{k+1}(i, j)) & \text{si } hd_k(i, j) < t_i < hd_{k+1}(i, j) \end{cases} \quad (I.1)$$

La figure (Fig I.2) représente un graphe dépendant du temps modélisé par un graphe dynamique.

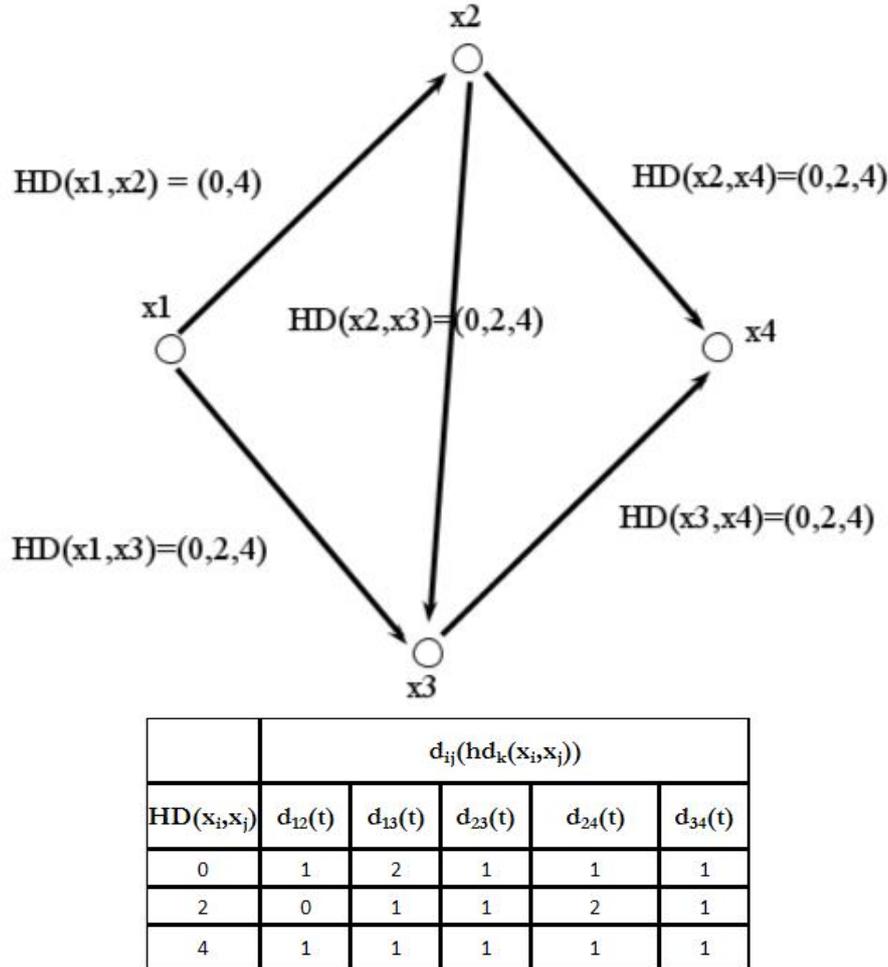


Figure I.2 – Illustration du modèle par tables horaires pour un graphe dynamique

Modèle espace-temps

Dans [19, 83], les auteurs transforment le graphe dépendant du temps $G(t) = (N, E, D(t))$ en un graphe statique appelé graphe espace-temps et noté $G_{ET} = (N_{ET}, E_{ET}, D_{ET})$ dans lequel les nœuds sont dupliqués pour les différents instants caractérisant des changements dans les

valuations de leurs arcs sortants.

Pour cela, à chaque date de départ t_i d'un nœud i de $G(t)$, on associe un couple (i, t_i) qui représente un nœud dans G_{ET} .

Ainsi, à partir de l'ensemble N des nœuds de $G(t)$, on construit tout d'abord l'ensemble des nœuds suivants : $\{(i, t), \forall i \in N, \forall t \in \{hd_1(i, j), hd_2(i, j), \dots, hd_k(i, j), \dots, hd_{N_{i,j}}(i, j)\}, \forall j \in Succ(i)\}$.

De plus, pour tout arc (i, j) du graphe dynamique $G(t)$, il y a un ensemble d'arcs reliant les nœuds (i, t) aux nœuds $(j, t + d_{i,j}(t))$. Si la date $t + d_{i,j}(t)$ ne correspond pas à une date de départ définie pour j , on crée le nœud $(j, t + d_{i,j}(t))$. Dans ce cas, l'instant associé au nœud j correspond à une date d'arrivée en j .

Enfin, si l'attente est autorisée dans les nœuds du graphe $G(t)$, on peut créer dans le graphe G_{ET} des arcs reliant le même nœud i de $G(t)$ à des instants différents : $((i, hd_k(i, j)), (i, hd_{k+1}(i, j)))$. Ainsi dans le graphe G_{ET} , chaque nœud de $G(t)$ est dupliqué en un ensemble de nœuds étiquetés par des événements de départ ou d'arrivée à ce nœud. Les arcs de G_{ET} modélisent alors les déplacements entre les nœuds ou les attentes dans les mêmes nœuds. Le graphe G_{ET} ainsi obtenu devient statique et résoudre un problème de plus court chemin sur le graphe $G(t)$ revient à résoudre un problème de plus court chemin classique sur G_{ET} [19, 83, 88].

La figure (Fig I.3) reprend le graphe de l'exemple précédent (Fig I.2) et le modélise à l'aide d'un graphe espace-temps G_{ET} .

Les réseaux de transport dépendants du temps sont classés selon qu'ils respectent ou non la propriété FIFO (First In First Out) : partir plus tôt d'un point i permet toujours d'arriver plus tôt en un point j . En d'autres termes un réseau de transport est dit FIFO si et seulement si pour tout arc (i, j) du graphe G , on vérifie la propriété FIFO suivante illustrée sur la figure (Fig I.4) :

$$t + d_{ij}(t) \leq t' + d_{ij}(t') \quad \forall t, t' \quad t < t'$$

3.2 Réseaux de transport multimodaux

3.2.1 Modélisation par couche

En se basant sur les travaux existant de modélisation des graphes par niveaux, dans les travaux de [14, 62], le réseau multimodal est modélisé comme une généralisation d'un réseau monomodal par l'ajout d'un ensemble de paramètres (étiquettes sur les arcs et sur les nœuds)

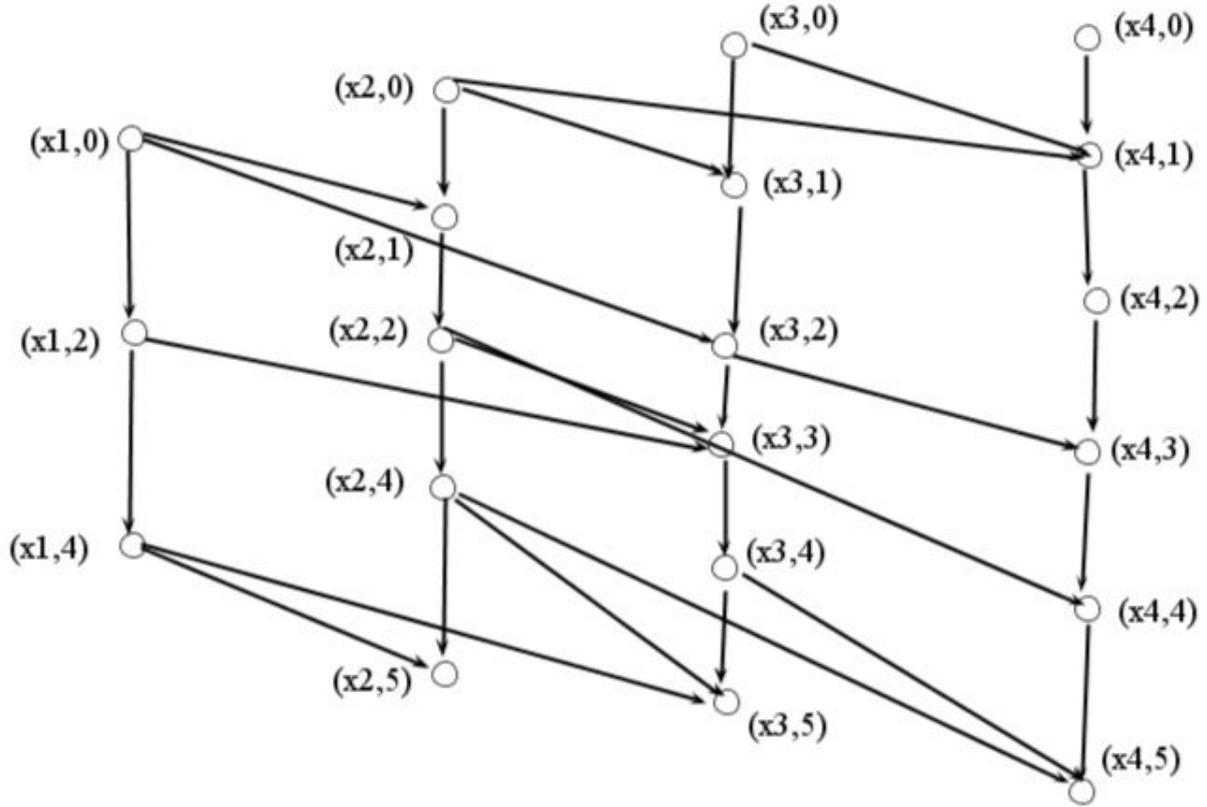


Figure I.3 – Exemple d'un modèle espace-temps

relatifs aux différents modes. Certains modes de transport (bus, train, tram, etc.) fonctionnant en pratique sur différentes lignes prédéfinies sont considérés comme des couches à part.

De ce fait, le réseau de transport est modélisé par couches et est représenté par un graphe, noté $G_C(N, E, D, M)$ où M correspond à l'ensemble des modes existant dans le réseau. Le graphe G_C correspond à un ensemble de M graphes associés chacun à un mode de transport (i.e. une couche de transport). Un exemple de modélisation par couches est représenté dans la figure I.5.

A chaque nœud $i \in N$, on associe une étiquette notée m_i donnant le mode auquel il appartient. De même à chaque arc (i, j) , en plus de la valuation $d_{i,j}$, on associe une étiquette représentant le mode auquel cet arc est associé. Un arc (i, j) reliant deux nœuds appartenant à des modes différents ($m_i \neq m_j$) est appelé un arc de transfert. Dans la pratique, les transferts sont considérés comme une couche spécifique [62] et donc un mode spécifique que l'on ajoute à l'ensemble M des modes de transport effectifs.

Parmi l'ensemble des modes, on peut distinguer les modes dépendant du temps notés M_{TD}

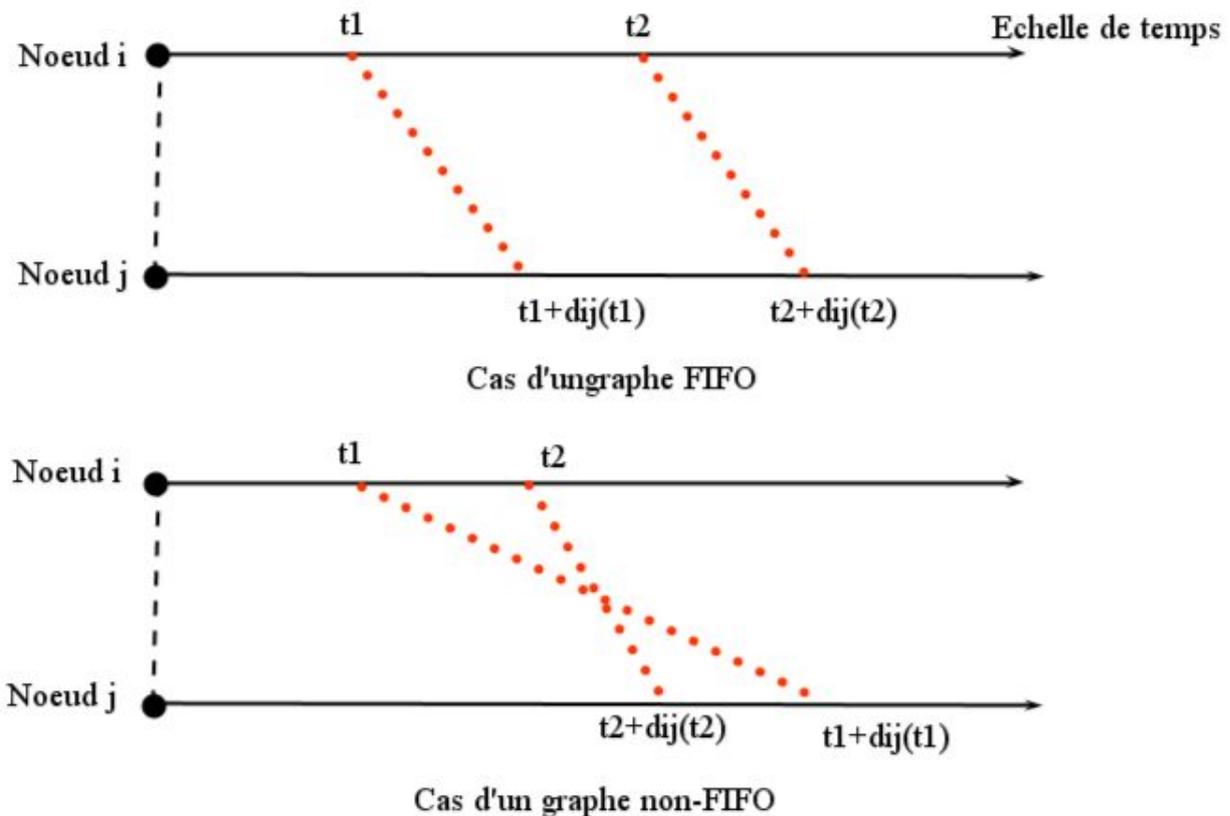


Figure I.4 – Exemple de graphes dépendants du temps FIFO et non-FIFO

des modes statiques M_S . Les couches correspondant à des modes dépendants du temps peuvent être représentées par un graphe dynamique ou un graphe espace-temps présentés ci-dessus. Les couches correspondant à des modes statiques sont modélisées par des graphes statiques.

3.2.2 Modélisation par hypergraphe

Un réseau de transport multimodal peut également être modélisé en utilisant la notion d'hypergraphe qui généralise la notion de graphe dans le sens où les arêtes appelées hyperarêtes ne relient plus deux nœuds, mais un nombre quelconque de nœuds compris entre 2 et le nombre de sommets de l'hypergraphe (cf figure I.6). En effet, certains modes de transport peuvent être modélisés par un hypergraphe (par exemple des lignes de bus).

Ainsi dans Lozano et al. [63], les auteurs définissent un graphe multimodal $H = (N, E, D, M)$ où N est l'ensemble des nœuds, E l'ensemble des hyper-arcs (h-arcs), D la valuation des h-arcs et M l'ensemble des modes associés aux h-arcs.

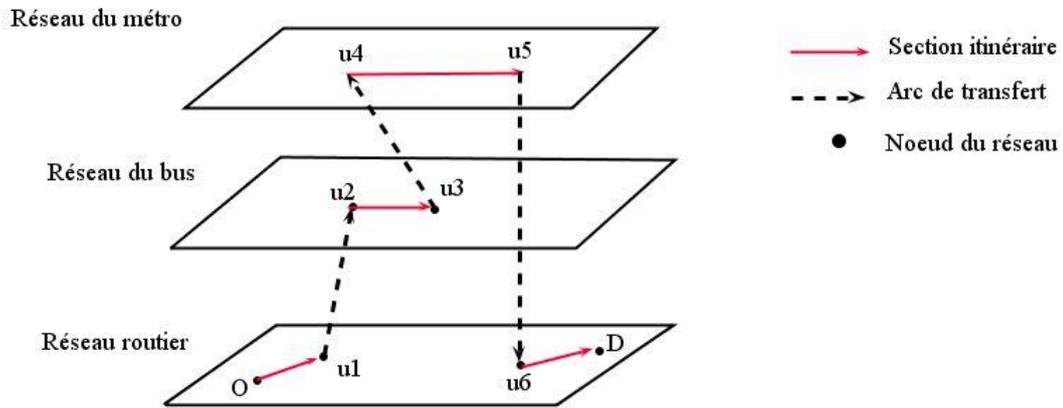


Figure I.5 – Modélisation par couches d'un réseau de transport multimodal.

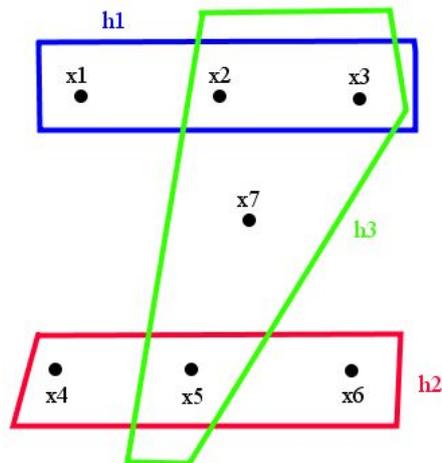


Figure I.6 – Exemple d'hypergraphe

4 Calcul d'itinéraires multimodaux

4.1 Position du problème

Dans le domaine des transports, le problème de calcul d'itinéraires a été largement étudié. Il est principalement utilisé pour deux activités :

- l'analyse de la performance des réseaux de transport. En effet, lors de la conception de ces réseaux, les urbanistes vont analyser l'accessibilité de certains lieux publics moyennant ces réseaux. On peut aussi effectuer des calculs d'isochrones pour déterminer les zones

Chapitre I. Modèles et algorithmes de recherche d'itinéraire en transport multimodal

atteignables avec une durée limitée à partir de certains points spécifiques (cf. figure I.7). Ces analyses permettent généralement de faire un choix sur l'emplacement des lignes de transport.

- l'aide aux déplacements et à la planification des itinéraires qui consiste à proposer des outils de calculs d'itinéraires point à point afin de répondre à des requêtes de transport émises par des utilisateurs du réseau (cf. figure I.8).

Ces deux activités de calculs d'accessibilité ou d'isochrones et de calcul d'itinéraires s'appuient sur des algorithmes de recherche de plus courts chemins que nous allons présenter dans la suite de ce chapitre.

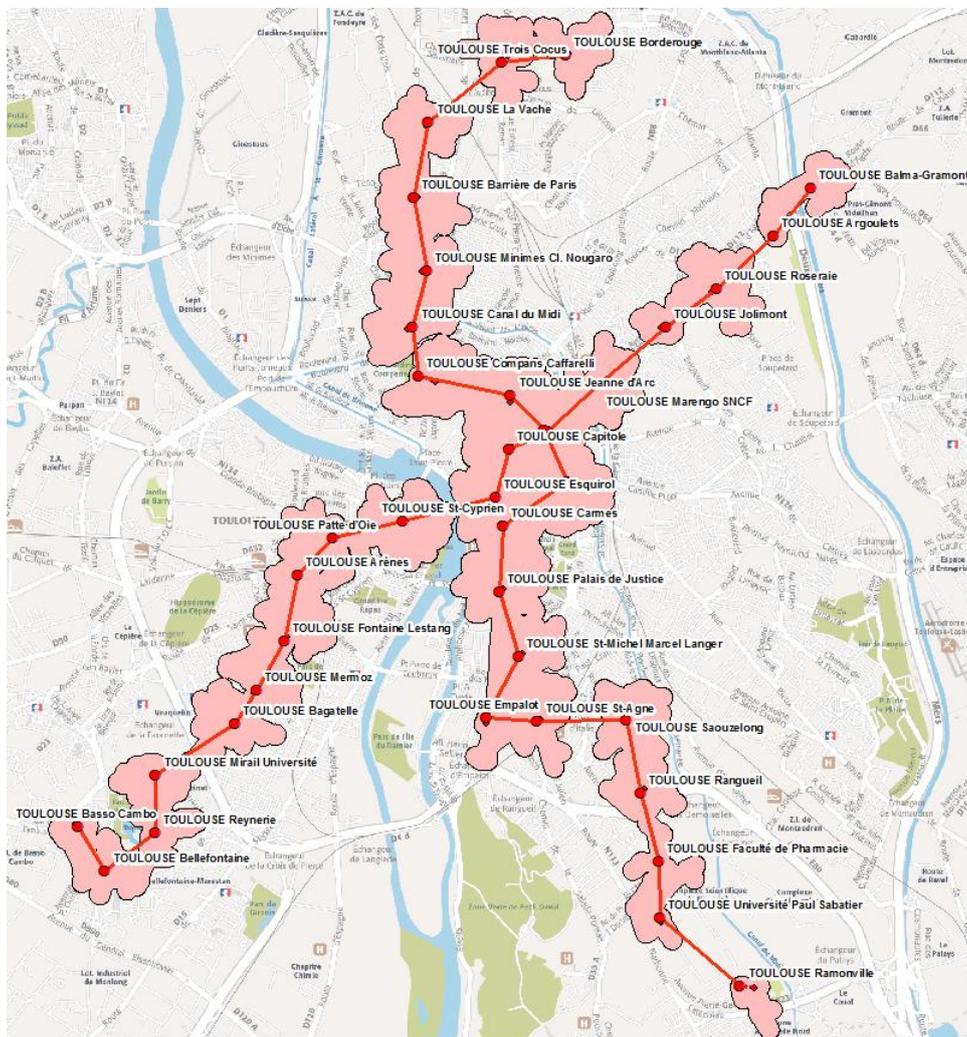


Figure I.7 – Isochrones : points accessibles en 15 minutes des stations de métro de Toulouse

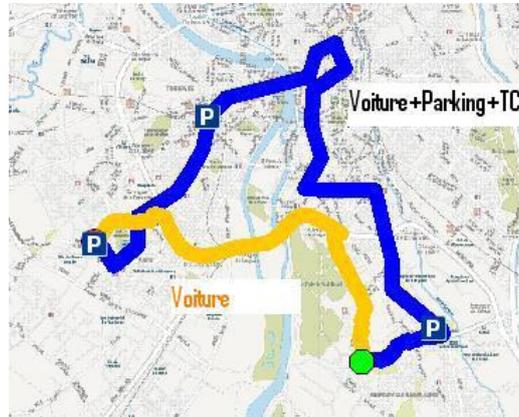


Figure I.8 – Itinéraires multimodaux

4.2 Différents problèmes de plus courts chemins

L'un des problèmes d'optimisation les plus anciens et les plus traités dans le domaine du transport est le problème de plus court chemin [1, 13, 42]. Sa résolution a suscité une multitude de travaux, car son efficacité et la qualité des solutions obtenues sont fortement liées au choix de la méthode implémentée ainsi qu'aux types de contraintes considérées.

Ce problème consiste à trouver le chemin partant d'un nœud origine O vers un nœud destination D qui minimise un coût.

Ce problème général recouvre différents problèmes en fonction :

- du type d'itinéraire recherché : chemin point à point de O vers D , chemin d'une origine O vers tout autre sommet, chemin entre tout couple de sommet (ou distancier) ;
- du nombre de chemins souhaités : 1 ou k plus courts chemins ;
- des objectifs à optimiser : un ou plusieurs objectifs à considérer.

De plus, les méthodes de résolution varient selon la nature du réseau de transport (mono ou multimodal, statistique ou dépendant du temps, FIFO ou non FIFO) et selon l'intégration ou non de contraintes sur la nature des chemins obtenus comme des contraintes de viabilité ou des contraintes de ressources.

4.3 Plus courts chemins sur un graphe monomodal

4.3.1 Plus courts chemins sur des graphes statiques (Shortest Path Problem ou SPP)

Les problèmes de calcul d'itinéraires dits statiques, se basent sur des graphes dans lesquels les valuations des arcs n'évoluent pas dans le temps.

Dans le cas de graphes statiques, les algorithmes de calcul d'itinéraires (point à point, origine vers tout autre sommet ou distancier) ont une complexité temporelle polynomiale dans le pire cas. La plupart des algorithmes de calcul de plus court chemin point à point sont des adaptations des algorithmes de plus court chemin origine vers tout autre sommet que l'on répartit classiquement en deux familles [81] : ceux à fixation d'étiquettes (algorithme de Dijkstra [30]) adaptés au cas où toutes les valuations du graphe sont positives ou nulles, ce qui est le cas de graphes représentant des réseaux de transport et ceux à correction d'étiquettes (algorithmes de Bellman [11], de Ford [37] ou de Moore [76]) applicables dans le cas de valuations quelconques. Dans ces algorithmes, on associe une étiquette ou label à chaque sommet du graphe contenant le coût du chemin allant de l'origine à ce sommet (ainsi que l'information du sommet prédécesseur ayant permis d'obtenir ce coût).

Algorithme de Dijkstra

Les réseaux de transport ayant des valuations positives, on s'intéresse par la suite de manière plus spécifique à l'algorithme de Dijkstra. Le principe de cet algorithme est de sélectionner à chaque itération le sommet de plus petit coût et d'étendre ce coût vers les sommets successeurs de la manière suivante : si i est le sommet sélectionné et π_i le coût de ce sommet, le coût des sommets successeurs vaut alors $\pi_j \leftarrow \min(\pi_j, \pi_i + d_{i,j}, \forall j \in Succ(i))$ (où $d_{i,j}$ est la valuation de l'arc (i, j)).

Il existe plusieurs variantes de cet algorithme de Dijkstra dont le principe est donné dans l'algorithme 1 qui ont visé à améliorer sa complexité temporelle. Une implémentation standard de l'algorithme de Dijkstra est en $O(n^2)$ [10, 22, 44, 68]. Une implémentation basée sur une structure de données appelée « tas binaire » ou « file de priorité », permettant d'accéder de manière efficace au sommet de plus petit coût, conduit à une complexité au pire en $O((m + n).\log n)$, ou en $O(m \log n)$ dans le cas de graphe connexes c'est-à-dire lorsque $m \geq n - 1$ [57]. La version dite à « bucket » de l'algorithme de Dijkstra est adaptée au cas où les valuations des arcs ont une valeur maximale pas trop importante. Elle consiste à partitionner les sommets en fonction de leur coût et à ranger les sommets ayant des coûts proches dans une même structure de données (un bucket) pour y accéder de manière efficace. La complexité temporelle au pire est en $O(U + n^2)$ où U est la valeur maximale des valuations des arcs, mais en pratique si peu de sommets ont des coûts identiques cette complexité peut se ramener à $O(U + M)$ pour des buckets mémorisant uniquement les sommets de même coût [27, 29, 33, 52, 57].

Malgré leurs complexités polynomiales, ces algorithmes peuvent néanmoins engendrer des temps de calculs importants pour des graphes de grande taille, ce qui a suscité pour le calcul point à point le développement des techniques d'accélération préservant généralement l'optimalité des solutions comme l'algorithme A^* , le parcours bidirectionnel, ainsi que différentes méthodes présentées dans [43].

Algorithme 1 Algorithme DIJKSTRA

ENTRÉES: $G(N, E)$; O : Origine; D : Destination; Q : file de priorité contenant l'ensemble des nœuds courants.

- 1: $\forall x \in N, x \neq O \pi_x \leftarrow +\infty, pred_x \leftarrow \emptyset$
 - 2: $\pi_O \leftarrow 0, pred_O \leftarrow O, Q = \{O\}$
 - 3: **tant que** ($Q \neq \emptyset$) **faire**
 - 4: Choisir $x \in Q$ tel que π_x minimal
 - 5: $Q \leftarrow Q - \{x\}$
 - 6: **pour tout** $y \in Succ(x)$ **faire**
 - 7: $\pi'_y \leftarrow \pi_x + d_{x,y}$
 - 8: **si** $\pi'_y < \pi_y$ **alors**
 - 9: $\pi_y \leftarrow \pi'_y$
 - 10: $pred_y \leftarrow x$
 - 11: **si** $y \notin Q$ **alors**
 - 12: $Q \leftarrow Q \cup \{y\}$
 - 13: **fin**
 - 14: **fin**
 - 15: **fin pour**
 - 16: **fin tant que**
 - 17: Pour toute destination D , déterminer le plus court chemin (PCC) en remontant les prédécesseurs en partant de D .
-

Algorithme A^*

Dans le domaine de l'intelligence artificielle, l'algorithme A^* est utilisé pour parcourir des graphes d'états d'un état initial vers un état but tout en guidant de manière heuristique le parcours dans le graphe en fonction du but à atteindre de telle sorte que l'état but soit obtenu par le moins de changement d'états possibles. Cet algorithme permet la recherche de plus court chemin d'une origine vers une destination (point à point) dans des réseaux de transport [48, 56, 80, 84]. Il fonctionne de manière similaire à l'algorithme de Dijkstra en rajoutant à chaque nœud une fonction heuristique en plus de la fonction de coût (ie. coût du plus court chemin depuis l'origine).

Chapitre I. Modèles et algorithmes de recherche d'itinéraire en transport multimodal

Pour un nœud donné i , la fonction heuristique est une estimation du coût nécessaire pour atteindre la destination depuis ce nœud. On note dans la suite de ce paragraphe :

- $h(i)$: la fonction heuristique estimant le coût du nœud i à la destination ;
- $g(i)$: la fonction donnant le coût réel du trajet de l'origine au nœud i ;
- $f(i) = g(i) + h(i)$: la fonction d'évaluation d'un nœud lors de l'algorithme A^* .

Dans l'algorithme A^* (Algorithme 2), le nœud i sélectionné à chaque étape de l'algorithme est celui ayant la plus petite valeur $f(i) = g(i) + h(i)$. L'algorithme A^* est aussi dit dirigé par le but : il sélectionne les nœuds ayant à la fois le plus petit coût depuis l'origine et le plus petit coût par rapport à la destination.

L'algorithme A^* garantit d'obtenir les plus courts chemins souhaités tant que la fonction d'évaluation $h(i)$ est une borne inférieure du plus court chemin de i vers la destination.

La figure ci-dessous (figure I.9), tirée de [43], illustre la réduction de l'espace de recherche obtenue par un algorithme A^* en comparaison avec l'algorithme de Dijkstra. Elle permet de mettre en évidence l'importance du principe de A^* : les nœuds explorés sont en grande majorité ceux en direction de la destination alors qu'avec l'algorithme de Dijkstra, des nœuds éloignés de la destination sont explorés car à une étape de l'algorithme ce sont ceux qui avaient le plus petit coût par rapport à l'origine.

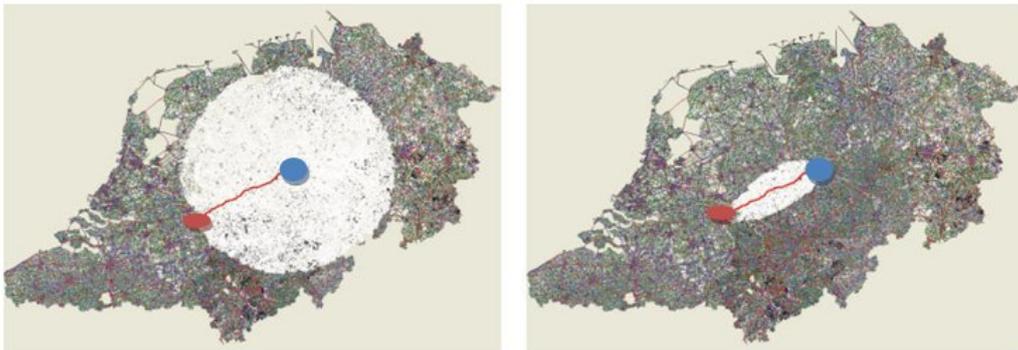


Figure I.9 – Comparaison des espaces de recherche entre les algorithmes Dijkstra avec et sans A^* [43]

L'algorithme de Sedgewick et Vitter [57] est également explicitement dédié à la recherche de plus court chemin point à point. Il est adapté au cas de graphes non orientés euclidiens et est basé sur le même principe que l'algorithme A^* . En effet pour chaque sommet il maintient à jour à la fois son coût depuis l'origine mais aussi un minorant de son coût par rapport à la destination exprimée en tant que distance euclidienne. Sa complexité temporelle au pire est en

Algorithme 2 Algorithme A^*

ENTRÉES: $G(N, E)$; Origine O ; D : Destination, Q : file de priorité contenant l'ensemble des nœuds courants.

```

1:  $\forall x \in N, x \neq O \quad f(x) \leftarrow +\infty, pred_x \leftarrow \emptyset$ 
2:  $f(O) \leftarrow h(O), pred_x \leftarrow O, Q = \{O\}$ 
3: tant que  $x \neq D$  faire
4:   Choisir  $x \in Q$  tel que  $f(x)$  soit minimal
5:    $Q \leftarrow Q - \{x\}$ 
6:   pour tout  $y \in Succ(x)$  faire
7:      $g'(y) \leftarrow g(x) + d_{x,y}$ 
8:     si  $g'(y) < g(y)$  alors
9:        $g(y) \leftarrow g'(y)$ 
10:       $f(y) \leftarrow g(y) + h(y)$ 
11:       $pred_y \leftarrow x$ 
12:      si  $y \notin Q$  alors
13:         $Q \leftarrow Q \cup \{y\}$ 
14:      finsi
15:    finsi
16:  fin pour
17: fin tantque
18: Déterminer le PCC en remontant les prédécesseurs en partant de  $D$ .
```

$O(m \log n)$.

Algorithme bidirectionnel

La plupart des méthodes de recherche de plus courts chemins sont unidirectionnelles c'est-à-dire qu'elles commencent à partir du nœud origine et s'arrêtent lorsque la destination est atteinte. La méthode bidirectionnelle d'abord suggérée par [5] et plus tard par [79] procède de manière différente en divisant la procédure de recherche en deux procédures séparées : la recherche avant (ou *Forward search*) gérée par un ensemble de labels Q_F et la recherche arrière (ou *Backward search*) gérée par un ensemble de labels Q_B .

L'idée de cet algorithme est qu'une solution est trouvée quand ces deux procédures de recherche se rencontrent en un sommet (cf figure I.10). L'algorithme calcule alternativement le plus court chemin à partir de l'origine π_i^f et à partir de la destination π_j^f tant qu'un critère d'arrêt n'est pas atteint.

Il existe deux facteurs influençant l'efficacité de l'algorithme [38] :

– le premier est le choix sur la manière d'alterner les deux procédures. En effet, itérer

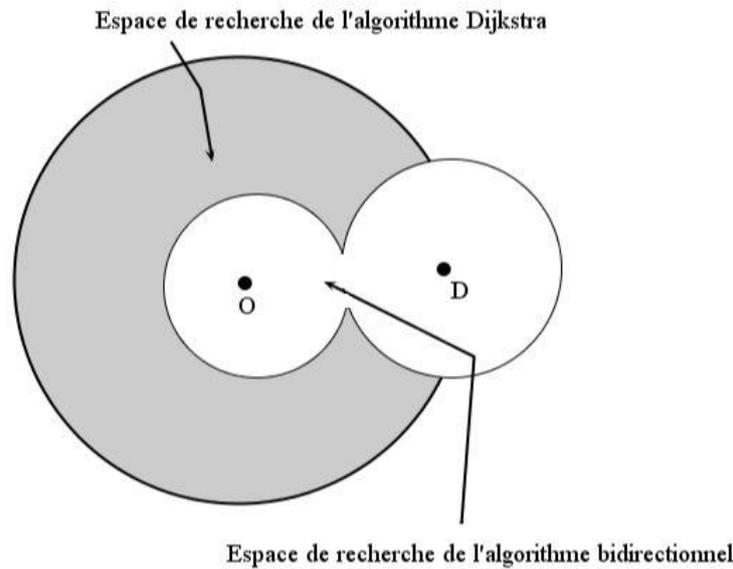


Figure I.10 – Espaces de recherche des algorithmes Dijkstra et bidirectionnel

de manière successive les deux procédures est la méthode la plus simple mais n'est pas obligatoirement la plus efficace ;

- le deuxième est la condition d'arrêt de l'algorithme afin de garantir l'optimalité de la solution. Cette condition d'arrêt a été introduit par Nicholson [79]. A chaque étape, elle considère la connexion (ie le chemin) de plus petit coût rencontré depuis le lancement de l'algorithme $\min_{i \in N} (\pi_i^f + \pi_i^b)$, cette connexion de coût minimal est le chemin optimal entre l'origine O et la destination D si la condition suivante est vérifiée (équation I.2) :

$$\min_{i \in N} (\pi_i^f + \pi_i^b) \leq \min_{(i') \in Q_F} \pi_{i'}^f + \min_{(i') \in Q_B} \pi_{i'}^f \quad (\text{I.2})$$

Cette condition exprime le fait que, si le coût du plus petit des chemins déjà obtenus est inférieur au coût de tout autre chemin susceptible de se connecter plus tard, alors ce chemin est optimal. L'algorithme 3 décrit le principe de l'algorithme bidirectionnelle.

Méthodes de prétraitement : les landmarks

A partir d'un graphe $G = (N, E)$ et d'un sous ensemble de nœuds D , appelés *landmarks*, pour lesquels on précalcule les plus courts chemins entre chaque sommet de G et chaque landmarks de D . Supposons que le coût du plus court chemin entre une origine O et une destination

Algorithme 3 Algorithme bidirectionnel

ENTRÉES: $G = (N, E)$, O : Origine; D : destination; Q_F et Q_B : des files de priorités resp pour le forward et le backward.

```

1:  $\forall x \in N, \pi_x^f \leftarrow +\infty, pred_x^f \leftarrow \emptyset, Q_F \leftarrow Q_F \cup \{O\}$ 
2:  $\forall x \in N, \pi_x^b \leftarrow +\infty, pred_x^b \leftarrow \emptyset, Q_B \leftarrow Q_B \cup \{D\}$ 
3:  $\pi_O^f \leftarrow 0, \pi_D^b \leftarrow 0, sortie \leftarrow FAUX$ 
4: tant que ( $Q_F \neq \emptyset$  et  $Q_B \neq \emptyset$  et  $Sortie = FAUX$ ) faire
5:   Choisir  $x \in Q_F$  tel que  $\pi_x^f$  soit minimal
6:   Choisir  $x' \in Q_B$  tel que  $\pi_{x'}^b$  soit minimal
7:   si  $\pi_x^f \leq \pi_{x'}^b$  alors
8:      $Q_F \leftarrow Q_F - \{x\}$ 
9:     pour tout  $y \in Succ(x)$  faire
10:       $\pi'(y) \leftarrow \pi_x^f + d_{x,y}$ 
11:      si  $\pi'(y) < \pi_y^f$  alors
12:         $\pi_y^f \leftarrow \pi'(y)$ 
13:         $pred_y^f \leftarrow x$ 
14:        si  $y \notin Q_F$  alors
15:           $Q_F \leftarrow Q_F \cup \{y\}$ 
16:        finsi
17:      finsi
18:    fin pour
19:    finsi
20:    si  $x' \in Q_B$  alors
21:       $Q_B \leftarrow Q_B - \{x'\}$ 
22:      pour tout  $(x', y), y \in N$  faire
23:         $\pi'(y) \leftarrow \pi_{x'}^b + d_{x',y}$ 
24:        si  $\pi'(y) < \pi_y^b$  alors
25:           $\pi_y^b \leftarrow \pi'(y)$ 
26:           $pred_y^b \leftarrow x'$ 
27:          si  $y \notin Q_B$  alors
28:             $Q_B \leftarrow Q_B \cup \{y\}$ 
29:          finsi
30:        finsi
31:      fin pour
32:    finsi
33:    si  $\min_{i \in N} (\pi_i^f + \pi_i^b) \leq \min_{(i') \in Q_F} \pi_{i'}^f + \min_{(i') \in Q_B} \pi_{i'}^b$  alors
34:       $Sortie = VRAI$ 
35:    finsi
36: fin tant que

```

D dans le graphe G , noté $d_G(O, D)$ vérifie l'inégalité triangulaire. On a alors :

$$d_G(O, D) \leq d_G(O, u) + d_G(u, D) \quad \forall u \in D$$

$$d_G(O, D) \geq |d_G(O, u) - d_G(u, D)|$$

A partir de ces deux inégalités on déduit :

$$\max |d_G(O, u) - d_G(u, D)| \leq d_G(O, D) \leq \min\{d_G(O, u) + d_G(u, D)\} \quad \forall u \in D$$

En d'autres termes, le coût du plus court chemin $d_G(O, D)$ est compris dans l'intervalle $[L, U]$, avec $L = \max |d_G(O, u) - d_G(u, D)|$ et $U = \min\{d_G(O, u) + d_G(u, D)\}$. Ainsi on peut noter qu'un landmark peut fournir la borne inférieure et un autre la borne supérieure.

Les algorithmes utilisant les landmarks sont généralement des algorithmes qui se déroulent en deux phases. Une phase consiste à choisir les d landmarks, puis à calculer les chemins entre les landmarks et les autres nœud de G . Cette phase est dite phase de preprocessing. Dans la deuxième phase, pour trouver le plus court chemin $d_G(O, D)$, on peut appliquer les algorithmes classiques. Pendant l'exécution de l'algorithme si un nœud landmark est dépilé alors on a trouvé le plus court chemin de $O - D$. La sélection des d landmarks est un problème NP-difficile [85].

Autres méthodes

Plusieurs autres approches ont également été proposées. La méthode ALT proposée par [47] est une méthode basée sur la recherche A^* , des Landmarks et de l'inégalité Triangulaire. Elle consiste à faire un prétraitement de manière à choisir un nombre constant de landmarks et de calculer et stocker les plus courts chemins entre tous les sommets et chacun de ces landmarks. Les bornes inférieures obtenues sont calculées en utilisant les coûts des plus courts chemins combinés de l'inégalité triangulaire.

La méthode SHARC (Shortcuts + Arc-Flags) [9, 24] qui est une approche rapide et robuste pour les grands graphes. Elle est basée sur l'adaptation des techniques développées pour les réseaux hiérarchiques [86] et pour l'approche Arcs-Flag qui est une étape de prétraitement dont l'idée est d'examiner une partition $C \subset N$ en utilisant un vecteur de flags pour chaque région C_i de C [59, 60].

On peut également citer la méthode de contraction hiérarchique [8] qui est basée sur le concept d'ordonner les nœuds selon leur importance. Ainsi la recherche utilise d'abord les nœuds les plus importants.

Depuis une dizaine d'années, de nombreuses techniques d'accélération de l'algorithme de Disjkstra ont été proposées, on peut trouver dans l'article [26] une présentation de l'ensemble

de ces techniques basées sur la topologie du graphe, sur le pré-calcul d'un certain nombre de plus courts chemins ou sur la combinaison de plusieurs de ces techniques. Ces améliorations ont permis de réduire à la fois les temps de calcul et l'espace mémoire nécessaire lors des pré-calculs de plus court chemin. Pour avoir un ordre de grandeur des résultats obtenus sur des graphes de très grande taille (de 18 à 33 millions de sommets pour 42 à 75 millions d'arcs), les temps de précalcul sont exprimés en dizaine de minutes et les temps de calcul d'un itinéraire sont exprimés en millisecondes.

Remarque : calcul d'un distancier

Pour déterminer les plus courts chemins de tout sommet vers tout sommet. Les algorithmes vus précédemment peuvent être appliqués de manière incrémentale en faisant varier le sommet origine. Cette approche ajoute un facteur n à la complexité des algorithmes. Il existe également un algorithme dédié, dit de Floyd et Warshall [36] qui est basé sur une représentation matricielle du graphe et dont la complexité au pire est en $O(n^3)$.

4.3.2 Plus courts chemins dépendant du temps (Time-Dependant SPP ou TD-SPP)

Le but du problème de plus court chemin dépendant du temps est de trouver le plus court chemin en temps de trajet entre un sommet origine O et un sommet destination D ou bien vers tous les sommets à une date de départ t_0 . D'autres variantes non présentées cherchent à calculer ces plus courts chemins pour toutes les dates de départ possibles. Plusieurs travaux se sont intéressés à ce problème et notamment ceux de [2, 19, 83].

Les problèmes de plus court chemin dans un graphe dépendant du temps présenté ici ont été largement étudiés dans la littérature et sont résolus avec des algorithmes à fixation d'étiquette de type Dijkstra [19, 21, 83] ou à corrections d'étiquettes [23].

On présente ici les algorithmes basés sur des extensions de l'algorithme de Dijkstra. Une étiquette est associée à chaque nœud $i \in N$ et possède deux attributs $\pi_i(t_0)$ et $pred_i$ qui représentent respectivement la valeur du plus court chemin de O à i et le nœud prédécesseur de i dans ce plus court chemin. L'algorithme 4, décrit ci-dessous, est une extension de l'algorithme de Dijkstra pour le problème TD-SPP dans le cas de réseau FIFO.

Cet algorithme permet de calculer l'ensemble des plus courts chemins vers tous les autres nœuds à partir du nœud de départ et à une date donnée. Comme l'algorithme de Dijkstra, il commence à initialiser pour les différents nœuds i les dates d'arrivée au plus tôt $\pi_i(t_0)$ ainsi

Chapitre I. Modèles et algorithmes de recherche d'itinéraire en transport multimodal

que les prédécesseurs à des valeurs infinies.

A chaque itération, l'algorithme sélectionne dans l'ensemble Q (représentant l'ensemble des nœuds à développer) le nœud i ayant la plus petite date d'arrivée soit le minimum des $\pi_i(t_0)$. A partir du sommet sélectionné i , on explore ses sommets successeurs dont les dates d'arrivées sont actualisés par $\pi_j(t_0) \leftarrow \min(\pi_j(t_0), \pi_i(t_0) + d_{i,j}(\pi_i(t_0)))$, $\forall j \in Succ(i)$.

Ce processus est répété tant qu'il existe encore des nœuds dans Q ou que la destination n'est pas atteinte. Le plus court chemin de O vers D est retrouvé en utilisant les valeurs stockées dans les attributs $pred_i$. Tous les algorithmes présentés dans la suite de ce mémoire sont des

Algorithme 4 Algorithme de plus court chemin dépendant du temps (FIFO)

ENTRÉES: $G = (N, E)$, O : Origine ; D : Destination, t_0 : date de départ ; Q : file de priorité.

- 1: $\forall i \in N, i \neq O \quad \pi_i(t_0) \leftarrow +\infty, pred_i \leftarrow \emptyset$
 - 2: $\pi_O(t_0) = t_0, pred_O \leftarrow O, Q = \{O\}$
 - 3: **tant que** ($Q \neq \emptyset$) **faire**
 - 4: Choisir $i \in Q$ tel que $\pi_i(t_0)$ soit minimal
 - 5: $Q \leftarrow Q - \{i\}$
 - 6: **pour tout** $(i, j), j \in Succ(i)$ **faire**
 - 7: **si** $\pi_j(t_0) > \pi_i(t_0) + d_{i,j}(\pi_i(t_0))$ **alors**
 - 8: $\pi_j(t_0) \leftarrow \pi_i(t_0) + d_{i,j}(\pi_i(t_0))$
 - 9: $pred_j \leftarrow i$
 - 10: **si** $j \notin Q$ **alors**
 - 11: $Q \leftarrow Q \cup \{j\}$
 - 12: **finsi**
 - 13: **finsi**
 - 14: **fin pour**
 - 15: **fin tantque**
-

extensions de l'algorithme 4.

Différentes améliorations de l'algorithme de Disjktra ont été adaptées avec succès dans le cas de réseau dépendant du temps comme A^* ou le calcul de landmarks. Les approches bidirectionnelles sont un peu plus délicates à mettre en œuvre. En effet, ne sachant pas à quelle heure on va arriver à destination, la phase de recherche en arrière ne peut fonctionner de la même manière que celle de recherche en avant. Nannicini et al. [77] considèrent à la fois un réseau dépendant du temps dans lequel les conditions réelles du trafic permettent d'actualiser la fonction de coût (temps de trajet). Ils proposent une extension des méthodes bi-directionnelles en considérant que la phase de recherche arrière est non dépendant du temps (les temps de trajet sont estimés par des bornes inférieures). Ils ont également proposés une extension des mécanismes de prétraitement avec utilisation des landmarks.

4.4 Optimisation Combinatoire Multi-Objectifs : MOCO

L'optimisation combinatoire est une discipline combinant diverses techniques des mathématiques discrètes et de l'informatique afin de résoudre des problèmes d'optimisation dont la structure sous-jacente est discrète. Un problème d'optimisation combinatoire est un problème qui consiste à maximiser (ou minimiser) une certaine fonction sur un ensemble fini d'éléments. Lorsque le problème nécessite la satisfaction de plusieurs objectifs, il s'agit d'un problème d'optimisation combinatoire multi-objectif (MOCO), appelé aussi problème d'optimisation multi-critère. Et parmi les problèmes MOCO, le plus court chemin multi-objectif (MOSPP) constitue l'un des plus étudiés [34].

En ce qui nous concerne, la multimodalité introduit des objectifs multiples puisque à l'optimisation des coûts habituels des chemins viennent s'ajouter des coûts spécifiques liés à la multiplicité des modes. C'est pourquoi nous donnons un bref aperçu des approches permettant de traiter les problèmes MOCO.

4.4.1 Définitions

Un problème d'Optimisation Combinatoire Multi-Objectif peut être défini par :

$$MOCO = \begin{cases} \min F(x) = (f_1(x), f_2(x), \dots, f_K(x)) & K \geq 2 \\ \text{s.c } x \in C \end{cases} \quad (I.3)$$

où $x = (x_1, x_2, \dots, x_n)$ sont les variables de décision et C est l'ensemble des contraintes.

Dans un problème d'optimisation multi-objectif, on ne cherche pas à optimiser une fonction mais un vecteur de fonctions F . Par exemple, dans un problème de transport multimodal, l'utilisateur souhaite en général l'optimisation de plusieurs objectifs tels que le coût, le temps de parcours, le nombre de changement de modes, le temps d'attente, le confort etc. Dans un problème multi-objectif, il n'y a pas de solution optimale unique mais un ensemble de solutions potentielles car en général aucune solution n'est la meilleure vis-à-vis de tous les objectifs simultanément.

En l'absence d'informations sur les préférences des décideurs, il n'est plus possible d'affirmer l'unicité liée à l'optimalité. C'est-à-dire que la valeur d'une solution ne peut être considérée

comme la meilleure de toutes. Néanmoins, il reste possible de comparer des solutions en utilisant l'ordre partiel donné par une relation de **dominance de Pareto**, définie ci-dessous. Une solution **Pareto optimale**, ou encore appelée solution efficace, ne peut être améliorée sur un objectif sans dégrader sa qualité sur un autre objectif. Selon cette relation, la comparaison de la valeur de deux solutions indiquera soit que l'une est meilleure que l'autre, soit qu'elles sont incomparables.

Définition 1. *La dominance de Pareto*

Une solution représentée par un vecteur de variable y domine un vecteur z si et seulement si :

1. $\forall i \in [1..K], f_i(y) \leq f_i(z)$
2. $\exists k \in [1..K]$ tel que $f_k(y) < f_k(z)$

En d'autres termes, y domine z si y est au moins aussi bonne que z pour tous les objectifs et y est strictement meilleure que z pour au moins un objectif.

Définition 2. *Pareto optimalité*

Les solutions qui dominent les autres mais ne se dominent pas entre elles sont appelées les solutions optimales au sens de Pareto ou solutions non dominées.

Une solution x^* est Pareto optimale si et seulement s'il n'existe pas de solution x tel que x domine x^* .

La dominance de Pareto peut être illustrée par la figure I.11 qui représente l'ensemble des solutions réalisables dans le cas d'un problème à 2 objectifs à minimiser. Les points en rouges sont les solutions non dominées par rapport aux objectifs f_1 et f_2 .

4.4.2 Les méthodes de résolution des problèmes d'optimisation multi-objectif

Plusieurs méthodes de résolution existent pour les problèmes MOCO [20]. Nous n'avons pas la prétention de montrer dans le détail le fonctionnement de toutes les méthodes. L'ensemble de ces méthodes ont été décrites, par exemple, dans la thèse de Jozefowicz [55]. Simplement, on se contentera de citer quelques méthodes parmi les plus répandues en introduisant leurs particularités.

Méthode par agrégation

C'est une méthode de transformation du problème multi-objectif (*MOCO*) à un problème mono-objectif (*MOCO $_{\lambda}$*). C'est la méthode la plus évidente et elle revient à définir une fonction

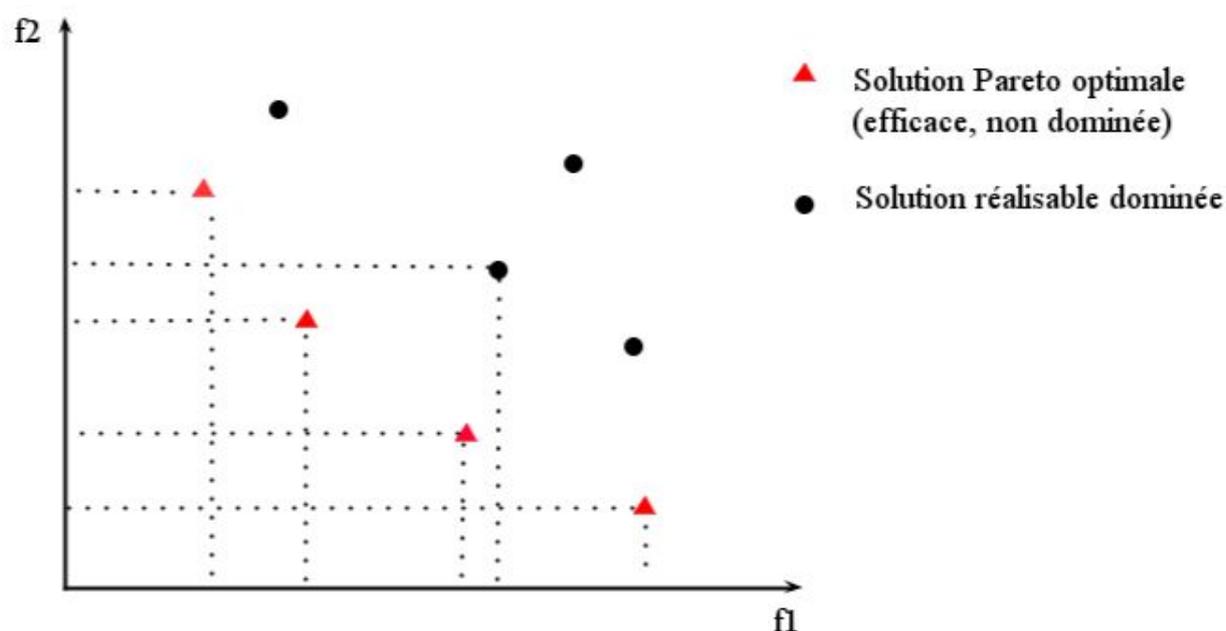


Figure I.11 – Front de Pareto.

de coût unique en introduisant une pondération des différentes fonctions objectif f_i du problème initial [53] :

$$F(x) = \sum_{i=1}^K \lambda_i f_i(x) \quad \text{ou} \quad \lambda_i \in [0..1] \quad \text{et} \quad \sum_{i=1}^K \lambda_i = 1 \quad (\text{I.4})$$

Les résultats obtenus dans la résolution du problème ($MOCO_\lambda$) dépendent fortement des paramètres choisis pour le vecteur de poids λ . Les poids λ doivent aussi être choisis en fonction des préférences associées aux objectifs, ce qui est une tâche délicate. Ainsi, une approche généralement utilisée est de résoudre le problème ($MOCO_\lambda$) avec différentes valeurs de λ .

Si les différents objectifs sont non-commensurables, on peut transformer l'équation précédente de la manière suivante :

$$F(x) = \sum_{i=1}^K c_i \lambda_i f_i(x) \quad (\text{I.5})$$

où c_i sont des constantes qui mettent à la même échelle les différents objectifs f_i . Les constantes c_i sont généralement initialisées à $\frac{1}{f_i(x^*)}$, où $f_i(x^*)$ est la solution optimale associée à la fonction objectif f_i . Dans ce cas, le vecteur est normalisé par rapport au vecteur idéal.

L'avantage de cette approche est la production d'une seule solution et ne nécessitent donc pas d'interaction avec le décideur et aussi l'utilisation de toutes les méthodes existant pour la résolution de problème mono-objectif. Un problème de cette approche est la détermination des poids, sans avoir de connaissances sur le problème traité. Cette approche a largement été utilisée dans la littérature à l'aide de différentes heuristiques et métaheuristiques : algorithmes génétiques, recherche tabou, recuit simulé, métaheuristiques hybrides etc. [73]

Méthodes ε -contraintes

Le problème consiste à optimiser une seule fonction prioritaire f_k en transformant les autres objectifs en contraintes.

$$MOCO_k(\varepsilon) = \begin{cases} \min f_k(x) \\ s.c \ x \in C \\ f_j \leq \varepsilon_j, \ j = 1, \dots, K; \ j \neq k; \ \varepsilon = (\varepsilon_1, \dots, \varepsilon_{k+1}, \dots, \varepsilon_n) \end{cases} \quad (I.6)$$

Ainsi, un problème mono-objectif (objectif f_k) sujet à des contraintes sur les autres objectifs est résolu. Différentes valeurs de ε_i peuvent être données pour pouvoir générer différentes solutions Pareto optimales. La connaissance a priori des intervalles appropriés pour les valeurs ε_i est requise pour tous les objectifs. L'approche ε -contrainte a été expérimentée en utilisant différentes heuristiques [73] : algorithmes génétiques, recherche tabou, métaheuristiques hybrides etc.

Méthodes de programmation par but

Dans cette méthode le décideur doit définir des buts qu'il désire atteindre pour chaque objectif. Ces valeurs sont introduites dans la formulation du problème, le transformant en un problème mono-objectif. Par exemple, la fonction objectif peut intégrer une norme pondérée qui minimise les déviations par rapport aux buts. Le problème peut être formulé de la manière suivante :

$$MOCO_k(z) = \begin{cases} \min (\sum_{i=1}^K \lambda_i |f_j(x) - z_j|^p)^{\frac{1}{p}} \\ s.c \ x \in C \end{cases} \quad (I.7)$$

4.5 Plus courts chemins multi-objectifs : MOSPP

4.5.1 Notations

Soit un graphe orienté valué $G = (N, E, C)$ où N est l'ensemble des n noeuds et E l'ensemble des m arcs et C une matrice de coût associée aux arcs. Chaque arc (i, j) est valué par un vecteur de K coûts : $(c_1(i, j), c_2(i, j), \dots, c_k(i, j))$. On note :

- $SP_{O,D}$ l'ensemble des chemins entre une origine O et une destination D ;
- $SP_{O,*}$ l'ensemble des chemins entre un noeuds O et les autres noeuds de N ;
- $z_p(r)$ le coût du chemin r selon l'objectif p avec $p = 1, \dots, K$;
- $z(r) = (z_1(r), z_2(r), \dots, z_K(r))$ le vecteur de coût d'un chemin r .

4.5.2 Formulation du problème

Le problème MOSPP est un des problèmes NP-difficile [89] d'optimisation multi-objectif le plus connu. Il peut se formuler comme suit :

$$\min_{r \in SP_{O,D}} (z^1(r), \dots, z^K(r)) \quad (\text{MOSPP}) \quad (\text{I.8})$$

Résoudre un problème MOSPP revient à calculer l'ensemble des solutions non dominées (l'ensemble de Pareto) de $SP_{O,D}$.

C'est l'un des problèmes MOCO les plus étudiés. En général, dans ce problème, deux types de critères peuvent être considérées, des critères de type somme, noté S-type, comme par exemple le temps de trajet et un critère de type max-min, noté M-type, comme par exemple le confort du transport. Dans le cas général on note $(\sigma - S | \mu - M)$ qui signifie que le problème comporte respectivement, σ critères du premier type et μ critères du second type.

Dans le cas d'un graphe statique, il a été démontré que dans le cas général, ce problème est NP-difficile. Il existe plusieurs algorithmes pour résoudre le MOSPP de type (S-type) [17, 46, 65, 92] pour lequel l'algorithme de Martins [66] est le plus utilisé. Le principe de cet algorithme est de mémoriser plusieurs étiquettes par noeuds (ie. les étiquettes non dominées) afin d'obtenir l'ensemble des solutions de Pareto. Pour cela, il considère deux ensembles de labels permanents (lp_i) et temporaires (lt_i) de la forme ($l_i = [z^1, \dots, z^K, j, h]$) où (z^1, \dots, z^K) est le vecteur de performance, j le noeud prédécesseur et h la position du label dans la liste des labels des noeuds j). L'algorithme commence par sélectionner le label ayant le minimum lexicographique de tous les labels temporaires, le convertit en label permanent et développe ses labels temporaires successeurs. La procédure s'arrête quand il n'y a plus de labels temporaires. Gandibleux et

al.[39] ont proposé une version révisée de l'algorithme de Martins pour les problèmes du type $(\sigma - S|1 - M)$.

Différentes applications du problème MOSPP sur des réseaux de transport statiques ou dépendants du temps peuvent être trouvées dans la littérature [32, 65–67, 75, 87]. Un état de l'art de ce problème de MOSPP est présenté dans [94].

Nous nous intéressons dans la suite de ce chapitre à un problème MOSPP dans le cas des réseaux de transport multimodaux sur lesquels il y a pour l'instant relativement peu de travaux.

5 Plus courts chemins multimodaux : MM-SPP

5.1 Plus courts chemins multimodaux viables : V-MM-SPP

Sur un réseau de transport multimodal, des contraintes d'utilisation des différents modes de transport peuvent apparaître. De telles contraintes sont appelées contraintes de viabilité et un chemin respectant ces contraintes est appelé un chemin viable. Par exemple, emprunter la voiture après un trajet en transport en commun est irréalisable.

La notion de viabilité a été introduite pour la première fois par Battista et al. [74] et il existe plusieurs approches pour modéliser ce problème. La première approche s'appuie sur la modélisation en couches du réseau de transport et met en œuvre des algorithmes pour traiter explicitement les modes associés aux nœuds et aux arcs.

On parle dans ce cas d'algorithme de plus court chemin avec contraintes d'étiquettes (label-constrained shortest path problem) [6, 7, 28, 49, 71, 90, 91, 93].

Ces algorithmes se basent d'une part sur un graphe orienté valué $G = (E, N, D)$, sur un alphabet σ et sur un langage L . Tout arc (i, j) est valué à la fois par un coût $d_{i,j}$ et par un mot $a_{i,j}$ de l'alphabet correspondant au mode associé à cet arc. Le problème V-MM-SPP consiste à déterminer un plus court chemin point à point tel que la concaténation des mots des arcs composant le chemin soit un élément du langage L .

Soit $SP_{O,D}$ un chemin $O-D$, ce chemin est formé d'un ensemble d'arcs $((O, x_1), (x_1, x_i), \dots, (x_j, D))$, la concaténation des mots de ces arcs est un mot noté $(a_{O,x_1}, a_{x_1,x_i}, \dots, a_{x_j,D})$.

Le langage L peut être représenté par un automate fini non déterministe [61, 62] $A = (S, \sigma, \delta, s_0, F)$ avec S un ensemble d'états, s_0 un état initial, F un ensemble d'états finaux, σ un alphabet, δ une fonction de transition : $\Sigma \times S \rightarrow 2^S$.

Différentes implémentations pratiques utilisant cette méthode ont été présentés dans [7, 90,

91] en se limitant à des problèmes mono-objectifs. Barrett et al. [7] Sherali et al. [91] étendent le problème au plus court chemin dépendant du temps et proposent un algorithme efficace et polynomiale dans le cas d'un graphe FIFO. Sherali et al. [90] étendent encore le problème dans une approche dépendant du temps de trajet et proposent un algorithme à fixation d'étiquettes qui est plus performant qu'un algorithme à correction d'étiquettes conçu pour ce même problème.

Cependant, les résultats expérimentaux obtenus restent limités pour des réseaux de grande taille. Face à ce constat, [25, 82], se sont intéressés à un réseau de transport spécifique comportant un graphe routier (utilisables par des véhicules et/ou des piétons) et un graphe de transport en commun (train ou avion) et considèrent que le déroulement d'un trajet débute et se termine toujours sur le graphe routier avec une utilisation d'un mode de transport en commun intermédiaire. Cette hypothèse permet de réduire les mots reconnus par le langage L. De plus, les auteurs ont développés des techniques de pré-calcul de plus courts chemins à partir de certains points spécifiques de leur réseau, appelés *access-node*, donnant accès aux modes de transport en commun. Ils considèrent également de manière séparée le graphe routier du graphe de transport en commun et peuvent ainsi utiliser les techniques connues d'accélération des algorithmes de plus courts chemins sur ces différents graphes. Les résultats expérimentaux montrent une efficacité de leur approche y compris sur des graphes de grande taille.

Une deuxième approche de la viabilité des chemins a été proposée par Lozano et Storchi [62] qui considèrent un problème de plus courts chemins multimodaux viables et bi-objectifs dans un graphe statique.

Dans ces travaux, la viabilité est représentée par un automate déterministe qui traduit l'évolution de la séquence d'utilisation des modes.

Les auteurs illustrent leur approche dans le cas d'un réseau de transport comportant 4 modes : la marche (*Ma*), le bus (*Bus*), la voiture (*Vo*) et le métro (*Me*) auquel s'ajoute le mode fictif correspondant au transfert (*Tr*). L'automate proposé dispose de 5 états traduisant leurs contraintes de viabilité sur les modes :

- le métro ne peut être utilisé qu'une seule fois ;
- la voiture est utilisée au début du trajet et une fois quittée, elle ne peut être de nouveau utilisée.

Ces 5 états sont décrits dans la figure I.12 et les transitions entre ces états sont représentées par l'automate de viabilité de la figure I.13.

Cet automate permet de modéliser l'ensemble des trajets viables multimodaux dans le cas des modes considérés. Par exemple, à partir de l'état initial s_0 , utiliser les modes *Ma* ou *Bus*

États	Description
s_0	l'état initial : aucun mode n'est encore utilisé
s_1	les modes métro et véhicule n'ont pas été utilisés
s_2	le mode véhicule n'a pas encore été quitté
s_3	le mode véhicule a été quitté et le mode métro n'a pas encore été utilisé
s_4	le mode métro n'a pas encore été quitté
s_5	le mode métro a été quitté

Figure I.12 – Description des états d'un chemin viable

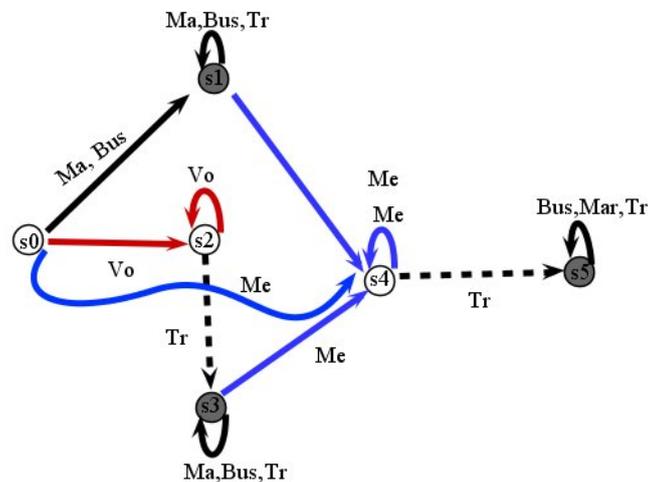


Figure I.13 – Modélisation de la viabilité par un graphe d'états (Lozano et Storchi) [62]

conduit dans l'état s_1 et utiliser le mode Vo conduite dans l'état s_2 . Les états finaux de cet automate sont les états s_1, s_3, s_5 (ce qui suppose qu'on ne peut arriver à la destination qu'en marchant ou en bus).

Les contraintes de viabilité souhaitées sont bien prises en compte par cet automate : le mode Vo ne peut être utilisé qu'en début de trajet et une fois qu'il est quitté il ne peut être de nouveau utilisé. Le mode Me ne peut être utilisé qu'une seule fois durant un trajet.

Dans l'algorithme proposé par les auteurs, qui basé sur un algorithme de Dijkstra, les étiquettes associées aux différents sommets du graphe comportent un état de l'automate de viabilité (en plus du coût). Ainsi un même sommet i va apparaître plusieurs fois avec différents états s_i . Étendre l'étiquette d'un sommet i dans un état s_i vers un sommet successeur j avec un état s_j n'est possible que si la fonction de transition de l'automate de viabilité est définie de

s_i vers s_j . L'objectif étant de déterminer le chemin de plus petit coût, les étiquettes sont triées en fonction du coût indépendamment des états de l'automate de viabilité.

5.2 Plus courts chemins multimodaux et/ou multi-objectifs

Lozano et Storchi [62] s'intéressent à la recherche du plus court chemin multimodal viable et respectant une contrainte sur le nombre maximum de transferts. Il s'agit d'un problème de plus court chemin bi-objectif où l'un des objectifs est le temps de trajet (à minimiser) et l'autre objectif est le nombre de transferts (à minimiser également). Ce problème est polynomial. Ils proposent pour cela un algorithme incrémental sur le nombre de changement de modes basé sur celui de Dijkstra [83]. Pour un nombre de transferts k compris entre 0 et K_{max} , l'algorithme détermine le plus court chemin à k transferts et mémorise dans une liste de sommets ceux conduisant à $k + 1$ transferts au moins. Cette liste de sommets en attente est ensuite exploitée lors de l'itération suivante. La validation expérimentale est cependant limitée à un graphe de petite taille (21 nœuds et 51 arcs).

Dans le cas réseaux de transport multimodaux dépendant du temps, Bielli et al [12] s'intéressent à la recherche des K-plus courts chemins viables (K-SPP). Pour cela, ils proposent un modèle simplifié de la viabilité tout en introduisant des pénalités sur les mouvements tournants. L'algorithme de résolution utilisé est aussi une extension de celui de Pallotino [83]. Les évaluations ont été effectuées sur des graphes allant jusqu'à 1000 nœuds et de 3000 arcs.

Febbraro et Sacone [35] considèrent un problème de plus court chemin multimodal où le critère à minimiser est le temps de trajet sans prendre en compte de contraintes additionnelles sur la composition des trajets. Ils proposent un algorithme adapté de celui de Dijkstra et le valident sur un graphe de petite taille (80 arcs et 66 nœuds).

Foo Heng Meng [72] s'intéresse à un problème de plus court chemin multimodal avec comme objectifs à minimiser la distance parcourue, le temps de trajet et le coût des billets. Cette application a été réalisée à Singapour. Les algorithmes utilisés pour résoudre ce problème sont des modifications de l'algorithme standard de type Dijkstra. Les détails sur le fonctionnement des algorithmes peuvent être trouvés dans [58].

Qiang Li et Carl E. Kurt [61] décrivent dans leur papier un problème multi-objectif (minimisation du temps de parcours, du nombre de transferts et du coût du voyage) avec l'intégration

dans un SIG (Système d'Informations Géographiques). L'algorithme utilisé est une modification d'un algorithme à fixation d'étiquette pour les K-Plus Courts Chemins (K-SPP).

Dans le cas des K plus courts chemins, Quijin et al. [95] résolvent un problème de recherche d'itinéraire(s) dans un réseau de transport bimodal (bus et marche) en tenant compte de l'ensemble des critères à minimiser. Ces critères sont le temps de trajet, le nombre de transfert et la distance de marche. L'algorithme utilisé est basé sur l'extension d'un algorithme à fixation d'étiquettes. Cette étude a été appliquée dans le transport public de Nottingham avec le réseau composé de 2398 arrêts et 292 lignes de bus. Le temps moyen pour calculer un seul itinéraire est de 0, 5 secondes, pour 2 itinéraires 3 secondes et pour 4 itinéraires est de 20 secondes.

Pour les problème de plus court chemin multimodaux et multi-objectifs, des extensions de l'algorithme de Martins ont été proposées [45], ainsi que pour faire face à la complexité en temps de calcul, des algorithmes génétiques [16] et un algorithme de colonie de fourmis [4]. Dans [15], l'auteur décrit un problème de recherche d'itinéraires multimodaux viables aller-retours dans des réseaux de transports urbains. La résolution de ce problème est basée sur une adaptation d'un algorithme à correction d'étiquettes.

6 Conclusion

Nous avons présenté un état de l'art sur la modélisation des réseaux de transport monomodaux et multimodaux et sur les algorithmes de calcul d'itinéraires. Le problème de calcul d'itinéraires multimodaux, qui est souvent par nature multi-objectif, peut rapidement devenir un problème NP-Difficile en fonction de la nature des objectifs considérés.

Nous allons dans le chapitre suivant définir le problème de transport multimodal que nous avons étudié : un problème bi-objectif avec contraintes sur les enchaînements de modes ou contraintes de viabilité qui est connu pour être polynomial.

Dans le contexte de mise en place d'un logiciel industriel de calcul d'itinéraires multimodaux, nous avons considéré la minimisation des deux objectifs : le temps de trajet et le nombre de transferts. Ce choix a été guidée par les préoccupations de réalisation d'un logiciel industriel et par leur intérêt pratique avéré.

Au vue de la petite taille des réseaux multimodaux testés jusqu'à présent, le développement d'une étude expérimentale permettant de comparer différents algorithmes s'avère également un point important. Enfin, obtenir un algorithme efficace pour résoudre des problèmes de calcul d'itinéraires dans un réseau de taille réaliste est le défi que nous avons voulu relever.

Chapitre II

Modélisation et méthodes de résolution proposées

1 Introduction

Dans le chapitre précédent nous avons présenté un état de l'art sur les problèmes de recherche d'itinéraires. Dans ce chapitre, nous proposons une modélisation et différentes méthodes de résolution pour le problème de transport multimodal considéré.

Nous nous situons dans un contexte où nous cherchons à évaluer et proposer des informations pertinentes afin d'aider le voyageur pour les besoins en matières d'itinéraires multimodaux. Ainsi les travaux réalisés en partenariat avec la société MobiGIS visent à mettre en place un outil d'aide à la décision pour l'optimisation et la planification d'itinéraires multimodaux. En effet il s'agit de développer des algorithmes efficaces qui seront interconnectés avec des bases de données multimodales afin de fournir des solutions optimales ou approchées à partir des besoins explicités par les utilisateurs (Figure II.1)

Ce chapitre est structuré en six parties. Une introduction est donnée dans la première partie ; la deuxième porte sur la définition du problème de recherche d'itinéraires multimodaux étudié. La troisième est dédiée à la modélisation retenue pour les réseaux de transport multimodaux. La quatrième et la cinquième parties portent sur les méthodes de résolution proposées pour résoudre notre problème dans le cas non dépendant du temps et leur amélioration en utilisant les techniques d'accélération. La dernière partie avant de conclure est consacrée à la résolution de ce problème dans le cas dépendant du temps.

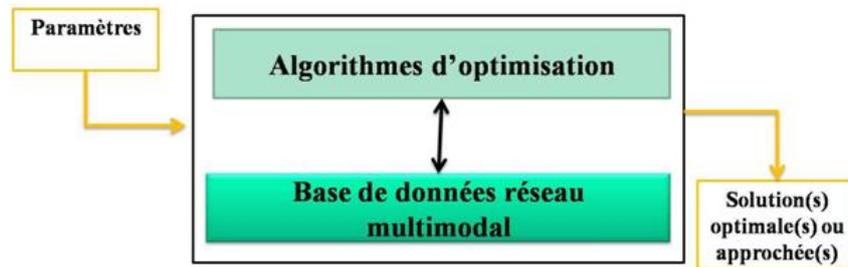


Figure II.1 – Outil d'aide à la décision de calcul d'itinéraires multimodaux.

2 Le problème étudié : définition informelle

Le problème central de ce travail est celui de la *recherche de chemins à coût minimal sur des réseaux de transport multimodaux*. La recherche de chemins de coût minimal recouvre en fait différents problèmes : calcul de trajets point à point, de trajets depuis une origine vers toute destination, de trajets depuis toute origine vers toute destination, calcul d'isochrones (zone accessible à partir d'une origine et respectant un coût donné), calcul de chemins mono ou multi-critères, calcul d'un seul meilleur chemin, ou des k meilleurs chemins, etc. Par ailleurs, la prise en compte de la multimodalité des réseaux de transport introduit un certain nombre de contraintes supplémentaires (fréquences et horaires de passage des bus et métro, vitesses de circulation fluctuantes en fonction des horaires ou des conditions de trafic, restrictions propres à chaque mode).

Le problème qui nous intéresse plus précisément est celui de la recherche du plus court chemin point à point bi-critère viable appelé **BI-MM-V-SPP : Bi-objective Multimodal Viable Shortest Path Problem**. Nous cherchons à la fois à minimiser le temps de parcours et le nombre de changements de modes (transferts modaux). Par changement de modes, on entend le passage d'un mode à un autre ou le passage entre deux lignes de bus différentes.

Dans le cas général, il s'agit d'un problème multi-objectif et dépendant du temps car les temps de trajet sont variables en fonction de l'horaire pour les modes de transport en commun. Les contraintes de ce problème d'optimisation sont les suivantes :

- le respect des contraintes de chaque usager : nombre maximal de transferts modaux, modes à exclure, etc. ;
- la viabilité des chemins c'est-à-dire des contraintes sur l'utilisation des différents modes ou la séquence d'utilisation des modes qui se doit d'être réalisable : on ne peut pas, par exemple avoir une solution entre une origine O et une destination D qui proposerait de

II.2 Le problème étudié : définition informelle

prendre sa voiture pour terminer un trajet commencé en transport commun alors que la voiture n'est disponible qu'au point origine O

Ces contraintes de viabilité peuvent provenir soit de contraintes utilisateur (limitation sur le nombre maximum de changement(s) de mode, limitation sur l'utilisation de certains modes, etc.) soit de contraintes spécifiques des modes (une fois que le mode véhicule personnel a été quitté, il ne peut plus être utilisé).

L'objectif principal de cette étude est de déterminer l'ensemble des plus courts chemins viable non dominés comme le montre la figure II.2 où apparaissent deux plus courts chemins non dominés : le premier, avec un transfert, se fait en voiture puis en transport en commun (TC), l'autre en voiture seule.

Nous considérons dans un premier temps le problème statique **BI-MM-V-SPP**, puis le problème dépendant du temps **BI-MM-V-SPP-TD**.

L'application spécifique de ce problème en relation avec la société MobiGIS est le calcul d'itinéraires multimodaux sur le réseau de transport de la ville de Toulouse.

D'autres types de réseaux sont également visés mais on ne présente ici que ce qui est lié à la ville de Toulouse. De même, d'autres applications de calcul d'itinéraires sont également considérées comme le calcul d'une origine vers plusieurs destinations ou le calcul d'isochrones multimodaux (ou accessibilité multimodale) qui consiste à délimiter les points accessibles en combinant les différents modes de transport en un temps donné (par exemple, la zone pouvant être desservie en moins de 30 minutes).

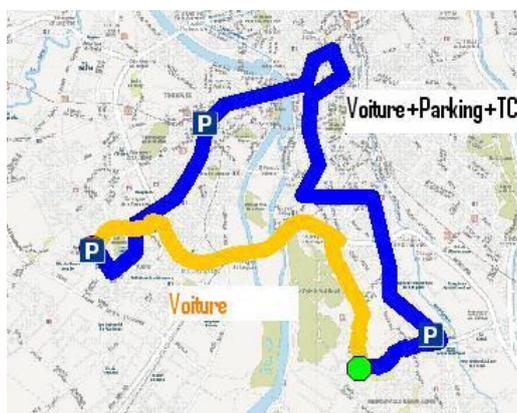


Figure II.2 – Exemple d'itinéraires multimodaux.

3 Modèles du problème : graphe multi-couche et graphe d'états

3.1 Modélisation du réseau de transport

Afin de résoudre les problèmes de recherche de plus courts chemins dans des réseaux de transport multimodaux, nous avons besoin d'une modélisation appropriée de ces réseaux. C'est cette modélisation qui conditionne les méthodes et outils utilisés pour une recherche efficace des plus courts chemins.

Pour modéliser des réseaux de transport multimodaux, nous avons choisi une approche multi-couches comme dans les travaux de [62]. Ainsi, chaque mode de transport est représenté par un graphe spécifique et des arcs dits de transferts permettant de passer d'un mode à un autre en des points donnés.

Dans l'approche proposée, la modélisation classique par les graphes est étendue par l'ajout des paramètres liés aux modes de transport disponibles dans le réseau et aux interactions entre ses modes. Cette approche a été favorisée par rapport aux hypergraphes du fait de l'utilisation simple de la représentation des données via les SIG (systèmes d'information géographique) pour la création du graphe multimodal. Nous détaillerons cette partie dans le chapitre IV.

Nous avons choisi de représenter le réseau de transport multimodal par un graphe orienté à plusieurs couches ou niveaux. Chaque mode de déplacement est représenté par un sous graphe situé sur un niveau spécifique. Dans les niveaux associés aux modes marche et véhicule particulier, les nœuds représentent les points d'intersections tandis que les arcs représentent les liaisons existant entre ces points. Un niveau est ajouté pour représenter l'ensemble des points de stationnement possibles.

Ces modes sont considérés ici indépendants du temps et peuvent être modélisés par un graphe statique présenté dans la section 3.1.1 du chapitre I.

Le niveau associé au transport en commun ou TC (train, bus, métro) est lui même divisé en plusieurs sous niveaux, chacun correspondant à un mode différent : métro, bus, train. Dans les sous niveaux, les nœuds représentent les arrêts ou stations de chaque ligne et les arcs, les dessertes entre ces arrêts.

Le réseau TC est souvent dépendant du temps. Les bus et les métros fonctionnent soit par des courses (tableaux d'horaires) soit en fréquence. Ce réseau TC est modélisé par un réseau

II.3 Modèles du problème : graphe multi-couche et graphe d'états

dynamique.

Des arcs de transferts considérés comme un "mode fictif" permettent de passer d'un niveau à un autre. On les emprunte au moins une fois dès qu'on veut changer de mode. Ces arcs représentent différentes actions :

- monter ou descendre d'un mode de TC ;
- rechercher une place de stationnement dans un parking ;
- sortir d'un parking et rejoindre la voirie ;
- marcher entre différents arrêts de TC ;
- marcher entre une place de parking et un arrêt TC ;

Le passage d'un mode "Voiture" au mode "TC" (bus par exemple) entrainera l'utilisation de deux arcs de transferts : un premier correspondant au temps nécessaire pour le stationnement du véhicule, un second au temps de déplacement depuis le point de stationnement jusqu'à l'arrêt de transport en commun.

Pour l'application étudiée avec la société MobiGIS, nous avons considéré les modes de transport suivants : Marche, Voiture, Métro et Bus. Pour pouvoir considérer les arcs de changements de modes dans le graphe du réseau de transport, nous utiliserons au besoin le mode "Transfert". Soit $M = \{Marche, Voiture, Metro, Bus\}$ l'ensemble des modes de transport considérés. Chaque mode de transport est modélisé par un graphe G_m où $m \in M$.

- Le réseau routier est modélisé par un graphe $G_R = \{N_R, E_R\}$ avec $n_R = |N_R|$ et $m_R = |E_R|$. Ce graphe peut être utilisé par les modes Marche ou Voiture. Un seul graphe est utilisé pour ces deux modes. Un arc $(i, j) \in E_R$ si et seulement si $i \in N_R$ et $j \in N_R$ et le transport est assuré par le mode $m \in M_R = \{Marche, Voiture\}$. Quel que soit l'horaire où s'effectue le trajet, le temps de trajet reste inchangé. Les temps de trajet pour ces modes dépendent uniquement de la vitesse des axes de circulation et sont supposés statiques c'est-à-dire indépendants du temps.

Si un arc du réseau routier est restreint par un mode m (par exemple l'autoroute pour la Marche), le temps de trajet est considéré infini.

- Les modes Bus et Métro du réseau du transport public $M_{TC} = \{Métro, Bus\}$ sont modélisés par ligne. Soit $L_{Métro}$ le nombre de lignes de métro et L_{Bus} le nombre de lignes de bus. Le graphe du mode Métro est composé de plusieurs couches :

$$G_{Métro} = \bigcup_{l=1}^{L_{Métro}} G_l^{Métro}.$$

Chapitre II. Modélisation et méthodes de résolution proposées

De même le graphe de bus est composé de plusieurs couches :

$$G_{Bus} = \cup_{l=1}^{L_{Bus}} G_l^{Bus}.$$

Ainsi le réseau de transport public est modélisé par un graphe $G_{TC} = G_{Métro} \cup G_{Bus}$.

Sur le réseau TC étudié de manière spécifique, il y a des tables horaires pour les lignes de bus et des fréquences de passage pour les lignes de métro. Ainsi les temps de trajet sur les arcs du graphe de transport en commun dépendent des horaires de réalisation du trajet. En effet, pour ces modes dépendant du temps, le temps de trajet l'arc (i, j) est noté $d_{ij}(t)$ puisque cette durée est fonction de la date de départ de la station i .

Pour chaque nœud i représentant une station ou un arrêt, nous associons une liste d'heures de départs planifiés notés $HD(i) = t_{d_1}(i), t_{d_2}(i), \dots, t_{d_{L(i)}}(i)$ avec $t_{d_k}(i)$ le k -ième départ possible associé au nœud i et $L(i)$ le nombre total de ces départs. Lorsqu'on s'intéresse au problème statique, les temps de trajet des arcs des modes de TC sont fixés à une valeur moyenne.

Les graphes $G_R, G_l^{Métro}, G_{l'}^{Bus}, l \in \{1, \dots, L_{Métro}\}$ et $l' \in \{1, \dots, L_{Bus}\}$ sont reliés entre eux par des arcs de transfert regroupés dans l'ensemble E_{Tr} . Un arc de transfert modélise le changement de mode et est valué par le temps de changement de mode.

Le graphe du réseau de transport multimodal $G = (N, E)$ est alors l'union des graphes G_R et G_{TC} auxquels s'ajoutent les arcs de transferts E_{Tr} , où N est l'ensemble des nœuds (arrêts, stations, intersections de rues, etc.) de cardinal $n = n_R + n_{TC}$ et E l'ensemble des arcs de cardinal $m = |E_R| + |E_{TC}| + |E_{Tr}|$.

Cette représentation permet de prendre en compte l'ensemble des opérations de transfert modal ainsi que de réaliser une optimisation simultanée du choix modal et du choix d'itinéraire, en raison de la continuité des représentations entre les différents réseaux de transport monomodaux.

Elle permet également de mettre en place assez facilement des contraintes de viabilité ou chaînage des modes, afin de proposer un itinéraire réalisable du point de vue des changements de modes.

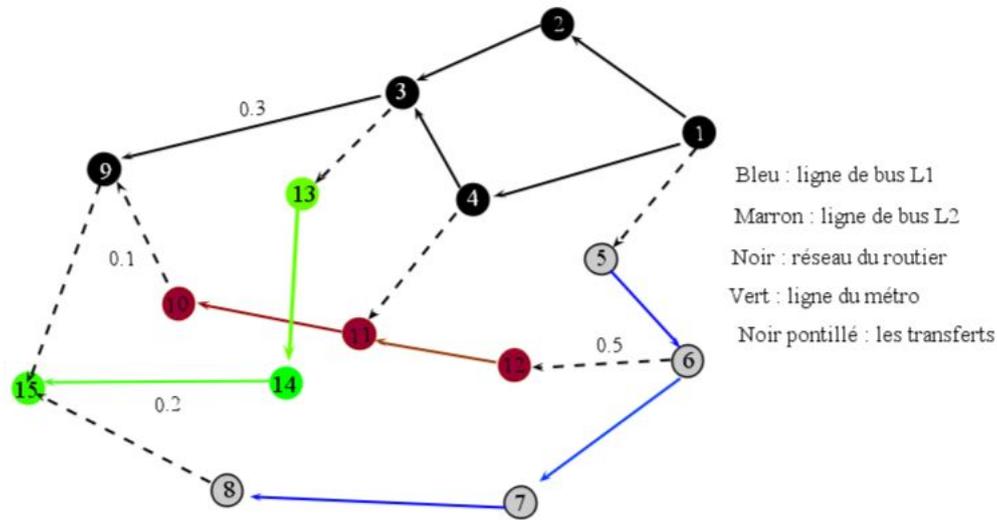


Figure II.3 – Exemple de réseau multimodal.

3.2 Modélisation de la viabilité

Un chemin est dit viable lorsqu'il respecte un ensemble de contraintes sur les modes et plus précisément sur la séquence d'utilisation. Ces contraintes sont issues soit des contraintes utilisateur soit de caractéristiques des différents modes.

Pour modéliser la viabilité d'un chemin, nous avons retenu l'approche proposée par Lozano [62]. Ainsi un chemin viable est modélisé par un automate fini $A = \{S, M, \delta, F, s_0\}$ où :

- $S = \{1, \dots, |S|\}$ est l'ensemble des états représentant l'évolution de la séquence de modes, s_0 est l'état initial;
- F est l'ensemble des états finaux;
- $\delta : M \times M \times S \rightarrow S$ la fonction de transition telle que $\delta(m, m', s)$ retourne l'état obtenu lorsqu'on passe d'un nœud du mode m et à l'état s à un nœud du mode m' . Par convention, $\delta(m_i, m_j, s) = -1$ signifie que l'évolution d'un nœud du mode m et à l'état s via le mode m' n'est pas viable (réalisable).

Les contraintes sur les modes que nous considérons sont : le métro ne peut être utilisé qu'une seule fois et une fois que le mode voiture a été quitté, il ne peut être utilisé de nouveau. Ces contraintes traduisent la viabilité d'un chemin sont représentées par l'automate de la figure II.4 où :

- $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$;
- s_0 : l'état initial ;
- F : ensemble des états finaux ;

L'état s_0 est l'état initial au nœud d'origine O , l'état s_1 signifie que la voiture n'a pas été encore prise depuis l'origine O et ne la sera plus pour le reste du trajet. Il indique de même que le métro n'a pas encore été pris.

L'état s_2 indique que la voiture a été empruntée depuis l'origine O et n'a pas été encore quittée. Il indique aussi que la voiture actuellement utilisée ne doit être quittée que dans une aire de stationnement pour atteindre la destination.

L'état s_3 signifie que la voiture ne plus peut être prise car elle a déjà été déposée à un parc de stationnement tandis que le mode métro n'a pas encore été utilisé.

Dans l'état s_4 le métro a été pris, mais pas à été quitté. L'état s_5 indique que le métro a été quitté et ne sera plus repris pour le reste du trajet.

Les états finaux F sont $\{s_1, s_3, s_5\}$ (affiché en gris dans la figure II.4) sont les états possibles à la destination D . Tout chemin viable partant de l'origine O aboutit à la destination D dans un état $s \in F$ (avec un nombre de transferts k). On suppose dans ce cas d'étude que l'on ne peut pas arriver à la destination en métro ou en voiture.

La complexité des algorithmes proposés dans les sections suivantes est fonction du nombre d'états de l'automate de la viabilité.

Étant donné les contraintes de viabilité considérées, l'automate proposé par Lozano et Storchi [62] (figure II.4) peut être réduit car les états s_1 et s_3 sont équivalents. Cela est possible du fait de la règle de dominance entre états décrite dans la section 4.2. Cet automate réduit mais équivalent sera considéré à la place de l'automate original de Lozano et Storchi [62].

3.3 Définition du problème

Compte tenu des modèles du réseau et de la viabilité d'un chemin présenté dans 3.1 et 3.2, le problème revient à trouver l'ensemble des chemins viables non dominés pour la minimisation du temps total de trajet et du nombre de transferts entre une origine O et une destination D . On considère dans la suite la variante où le temps de trajets est indépendant du temps ($d_{ij}(t) = d_{ij}$), notée **BI-MM-V-SPP**, et celle où le temps de trajet est dépendant du temps, notée **BI-MM-V-SPP-TD**.

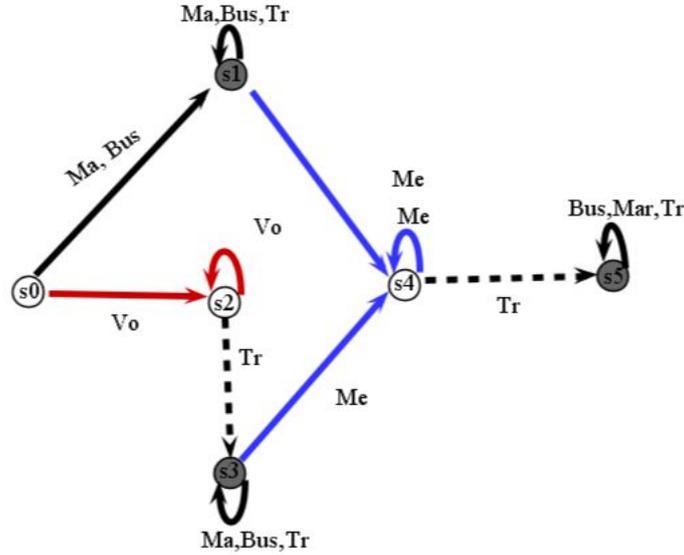


Figure II.4 – Automate de viabilité

4 Chemins partiels et règles de dominance

4.1 Étiquettes pour la représentation d'un chemin

Les algorithmes proposés pour la résolution des problèmes **BI-MM-V-SPP** ou **BI-MM-SPP-TD** sont des algorithmes d'extension et de fixation d'étiquettes (label setting algorithm) de type Dijkstra. Une étiquette (ou label) est identifiée par un triplet noté (i, s, k) représentant un chemin partiel viable de k transfert(s) partant du nœud origine, arrivant au nœud i dans l'état s noté $VP_{O_i}^{ks}$. A chaque label, on associe :

- t_{is}^k : le temps de trajet le plus court depuis l'origine au nœud i dans l'état s et avec un nombre de transfert(s) k ;
- $pred_{is}^k$: le label prédécesseur pour la reconstruction du chemin. Si $pred_{is}^k = (j, s', k')$ alors l'arc (j, i) est dans le chemin de coût minimum pour (i, s, k) , l'état du chemin au nœud j est s' , le nombre de transfert(s) utilisé est k' (avec $k' = k$ si $m_i = m_j$, $k' = k - 1$ si $m_i \neq m_j$) et $s = \delta(m_j, m_i, s')$.

Pour le cas des réseaux de transport dépendant du temps, sous l'hypothèse FIFO, il ne peut exister deux trajets de i vers j tel que l'un partant avant l'autre de i arrive après l'autre à j . Une seule valeur du temps minimal de trajet doit être stockée, et donc un seul label (i, s, k) donné. Ainsi pour un nœud i donné du réseau de transport, il y a plusieurs labels mais le nombre de labels à considérer est limité par le nombre d'états (au plus $|S|$) et par le nombre de transferts (au plus K_{\max}). Il s'ensuit que le problème à résoudre est polynomial.

4.2 Règle de dominance

Une relation de dominance permet de filtrer ou d'élaguer certains labels au profit d'autres pour ne retenir que les labels incomparables entre eux. Nous définissons dans cette section les règles de dominance suivantes : la règle de base de la dominance liée à l'optimisation bi-objectif et la règle de dominance entre états.

Définition 3 (Règle de dominance sur les deux objectifs). *Soient deux labels (i, s, k) et (i, s, k') . Si $k \leq k'$ et $t_{is}^k \leq t_{is}^{k'}$ alors (i, s, k) domine (i, s, k') . Le label (i, s, k') peut être abandonné.*

Ainsi un label d'un nœud i dans un état s domine un label pour ce même nœud i dans le même état s s'il correspond à la fois à moins de transfert et à une plus petite date d'arrivée.

Avec cette définition, un label (i, s, k) ne peut pas être dominé par un autre label (i, s', k') dès que $s \neq s'$. Nous définissons une deuxième règle de dominance qui renforce la règle de dominance initiale en s'appuyant sur les relations entre états.

Soit une relation binaire \preceq entre états, $s \preceq s'$ qui signifie que la séquence de modes correspondant à l'état s a au moins autant de possibilités d'être étendue que la séquence de modes correspondant à s' . Plus précisément $s \preceq s'$ si pour toute paire de modes de transport (m, m') , une des conditions suivantes est vérifiées :

$$\delta(m, m', s') = \delta(m, m', s) \tag{II.1}$$

$$\delta(m, m', s) = s \wedge \delta(m, m', s') = s' \tag{II.2}$$

$$\delta(m, m', s') = -1 \tag{II.3}$$

A partir de cette relation entre états on peut définir une nouvelle règle de dominance.

Définition 4 (Règle de dominance entre états). *Soient les labels (i, s, k) et (i, s', k') . Si $k \leq k'$, $t_{is}^k \leq t_{is}^{k'}$ et $s \preceq s'$ alors (i, s', k') est dominé.*

Démonstration. Nous montrons qu'un chemin arrivant à un nœud i dans l'état s a plus de possibilité d'extension qu'un chemin arrivant à i dans l'état s' .

Soient m et m' , pour chaque nœud i , si la condition II.1 est vérifiée alors tout état accessible à partir de s' par la transition de m à m' l'est aussi par la même transition. Si la condition II.2 est vérifiée, la transition de m à m' conserve l'état inchangé en s' et conserve également l'état inchangé en s . La condition II.3 garantit que toute transition non permise à s' ne l'est pas non plus à l'état s .

4.3 Simplification de l'automate de viabilité

De la relation entre états, on peut établir une condition pour fusionner les états de l'automate de la viabilité afin de réduire sa taille, qui impacte la complexité des algorithmes de calcul d'itinéraires.

Définition 5. Si $s \preceq s'$ et $s' \preceq s$ alors les états s et s' peuvent être fusionnés en un seul état.

Démonstration. Pour tout nœud $i \in V$, un chemin viable $O - i$ dans l'état s' est aussi un chemin viable $O - i$ à l'état s puisque $s \preceq s'$. Et de même un chemin viable $O - i$ à l'état s l'est aussi à l'état s' puisque $s' \preceq s$. De plus les deux chemins ont les mêmes possibilités d'extension. Par conséquent s et s' ne diffèrent pas par rapport aux séquences de modes permises et donc sont équivalents. Donc, ils peuvent être fusionnés en un seul état.

Les états s_1 et s_3 de la figure II.4 vérifient la condition décrite ci-dessus. Ainsi, à partir des contraintes de viabilité considérées et de la relation de dominance entre états que nous nous avons définie, nous pouvons réduire l'automate de viabilité. Il est remplacé par celui de la figure II.5 sera utilisé dans nos expérimentations.

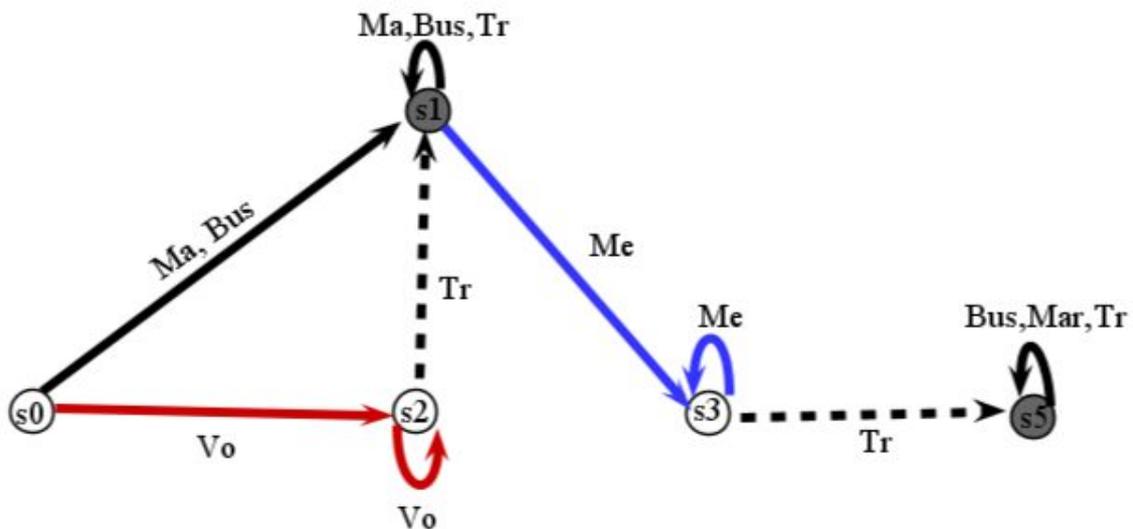


Figure II.5 – Automate de viabilité réduit par dominance

5 Algorithmes proposés pour le BI-MM-V-SPP

Nous considérons dans cette section le cas de réseaux de transport indépendant du temps bien que cela ne corresponde pas à la réalité des réseaux de transport, mais nous avons souhaité nous concentrer dans un premier temps sur la partie multimodalité du réseau.

5.1 Algorithme de Dijkstra MultiModal DIJ-MM

La méthode de résolution utilisée ici est du type ε -contrainte qui est une approche transformant le problème en un ou plusieurs problème(s) mono-objectif(s) (voir section 4.4 du chapitre I). L'algorithme est itératif sur le nombre maximum de changement(s) de modes : le plus court chemin est déterminé à l'aide de l'algorithme de Dijkstra pour chaque valeur du nombre de changement de mode K comprise entre 0 et K_{max} . De plus, pour ne conserver que les solutions non dominées, à chaque valeur de k donnée, seuls les chemins de coût inférieur au plus court chemin précédemment trouvé sont conservés.

5.1.1 Description de l'algorithme

L'algorithme présenté est une adaptation de celui de Dijkstra au cas multimodal. Il permet de calculer, à partir d'un nœud origine O , d'une date $t_0 = 0$ et d'un nœud destination D , l'ensemble des plus courts chemins multimodaux viables non dominés de 0 à K_{max} transferts. Les étapes de cet algorithme, noté **DIJ-MM** sont présentées dans l'algorithme 5. Il s'appuie sur des labels (i, s, k) (un unique label pour chaque nœud i du graphe de transport) et vise à déterminer les valeurs t_{is}^k et $pred_{is}^k$ de ces labels. Il nécessite également un automate de viabilité. Lors de l'initialisation (ligne 2), les dates d'arrivées t_{is}^k et les prédécesseurs $pred_{is}^k$ sont positionnés à des valeurs infinies. Le label $(O, s_0, 0)$ est mémorisé dans Q est sa date d'arrivée $t_{Os_0}^0$ est fixée à 0. De plus le coût du meilleur plus court chemin à 0 transfert ($CostPath_0$) est fixé à l'infini.

Les étapes suivantes (lignes 3 à 16) sont les itérations de l'algorithme de Dijkstra pour un nombre de transfert(s) K :

- sélection du label ayant la plus petite date d'arrivée (lignes 4-5)
- calcul des labels successeurs (lignes 8-13) avec prise en compte du changement d'état et de l'évolution du nombre de transferts
- actualisation éventuelle des labels successeurs en fonction non seulement de la valeur de la date d'arrivée mais aussi du nombre de transferts (devant être inférieur ou égal à k) et de la valeur du meilleur plus court chemin à $k - 1$ transfert(s) (lignes 14-16)

II.5 Algorithmes proposés pour le BI-MM-V-SPP

Lorsque la destination est atteinte, le label correspondant doit contenir un état final de l'automate et sa date d'arrivée doit être inférieure au coût du meilleur chemin précédent (ligne 21). Les lignes 23 et 24 mémorisent la solution à k transfert(s) ainsi que son coût.

Ce processus est réitéré tant que $K \leq K_{\max}$. A la fin de l'algorithme toutes les solutions non dominées sont trouvées.

Algorithme 5 Algorithme de Dijkstra MultiModal DIJ-MM

ENTRÉES: $G(N, E)$, O , D , $d_{ij}, \forall (i, j) \in E$, K_{\max} , $Costpath_k = \infty, t_0$

- 1: **pour** $K = 0 : K_{\max}$ **faire**
- 2: $Q := \{(O, s_0, 0)\}$, $t_{O, s_0}^0 := t_0$, $pred_{O, s_0}^0 := (O, 0, 0)$, $t_{is}^K := \infty, \forall i \in N \setminus \{O\}$
- 3: **répéter**
- 4: $(i, s, k) := \operatorname{argmin}\{t_{js'}^k | (j, s', k') \in Q\}$
- 5: $Q := Q \setminus \{(i, s, k)\}$
- 6: **si** $(i \neq D$ ou $s \notin F)$ **alors**
- 7: **pour** $j \in Succ(i)$ **faire**
- 8: $s' := \delta(m_i, m_j, s)$
- 9: **si** $m_i = m_j$ **alors**
- 10: $k' = k$
- 11: **sinon**
- 12: $k' = k + 1$
- 13: **finsi**
- 14: **si** $k' \leq K$ **alors**
- 15: **si** $s' \neq -1$ et $Costpath_k > t_{is}^k + d_{ij}, \forall s'' \preceq s', \forall k \leq k', t_{js''}^{k'} > t_{js'}^{k'}$ **alors**
- 16: $t_{js'}^{k'} := t_{is}^k + d_{ij}$, $pred_{js'}^{k'} := (i, s, k)$ et $Q := Q \cup \{(j, s', k')\}$
- 17: **finsi**
- 18: **finsi**
- 19: **fin pour**
- 20: **finsi**
- 21: **jusqu'à** $Q = \emptyset$ or $(i = D$ et $s \in F)$ or $t_{is}^k > Costpath_{K-1}$
- 22: **si** $i = D$ et $s \in F$ **alors**
- 23: Stocker t_{Ds}^k , $pred_{Ds}^k$
- 24: $Costpath_K = t_{is}^k$ (Plus court chemin à k transfert(s))
- 25: **finsi**
- 26: **fin pour**

5.1.2 Complexité de l'algorithme

L'algorithme de Dijkstra classique avec un tas binaire pour gérer la file de priorité a pour complexité : $O(M \log N)$. L'algorithme présenté ci dessus itère K_{\max} fois l'algorithme de Dijkstra en empêchant à chaque itération de créer un label de plus de k transfert(s). Dans ce cas la complexité (sans les règles de dominance) est celle d'un Dijkstra avec un sommet par triplet (i, s, k) c'est-à-dire $n|S|K_{\max}$ sommets et $m|S|K_{\max}$ arcs (où m est le nombre d'arcs et n le nombre de sommets du graphe modélisant le réseau de transport). La complexité avec un tas binaire est alors $O(m|S|K_{\max}^2 \log(n|S|K_{\max}))$ sans les règles de dominance. En rajoutant les règles de dominance (dominance de base en $O(1)$ et dominance entre états en $O(K_{\max}|S|)$ pour chaque nœud), la complexité de l'algorithme devient alors :

$$O(m|S|^2 K_{\max}^3 (\log(n|S|K_{\max})))$$

5.1.3 Discussion et limites de l'algorithme

L'algorithme détermine l'ensemble exact des solutions non dominées, c'est un algorithme optimal mais il présente toutefois quelques limites. En effet cet algorithme est itératif en fonction du nombre de transferts ($K = 0 \dots K_{\max}$), et la solution à k transfert(s) contient généralement des sous labels déjà explorés à l'itération précédente lors de la recherche à la solution à $K - 1$ transfert(s). Il présente ainsi une redondance importante lors de l'exploration de l'espace de recherche qui se traduit par un nombre important de nœuds explorés. Même si la complexité est polynomiale, cette redondance peut être coûteuse en temps et surtout pour des graphes de grande taille.

Pour réduire l'exploration redondante de certains labels, nous présentons dans la section suivante une évolution de cet algorithme.

5.2 Algorithme TLS (Topological Label Setting) Multimodal

C'est également un algorithme itératif sur le nombre de changement de modes, il calcule les plus courts chemins utilisant au plus k transfert(s) tout en mémorisant les chemins non dominés conduisant à $k + 1$ transfert(s) avec $k \leq K_{\max}$. Il est basé sur l'algorithme topologique de Pallottino et Scutellà [83] étendu par Lozano et al. [62] pour le problème de plus courts

chemins viables.

La structure de données Q pour trier les labels de coûts minimum est constituée de deux files de priorité Q^{now} et Q^{next} contenant respectivement les labels non-dominés ayant k et $k + 1$ transfert(s). Ces labels sont générés en fonction du nombre croissant de transferts.

En d'autres termes, cet algorithme, comme le précédent, calcule les solutions dans l'ordre croissant du nombre de transferts. Si on note $S_{K_{min}}$ (resp $S_{K_{max}}$) la solution obtenue pour la plus petite (resp la plus grande) valeur de k , les solutions sont obtenues par valeur décroissante du temps de trajet (figure II.6).

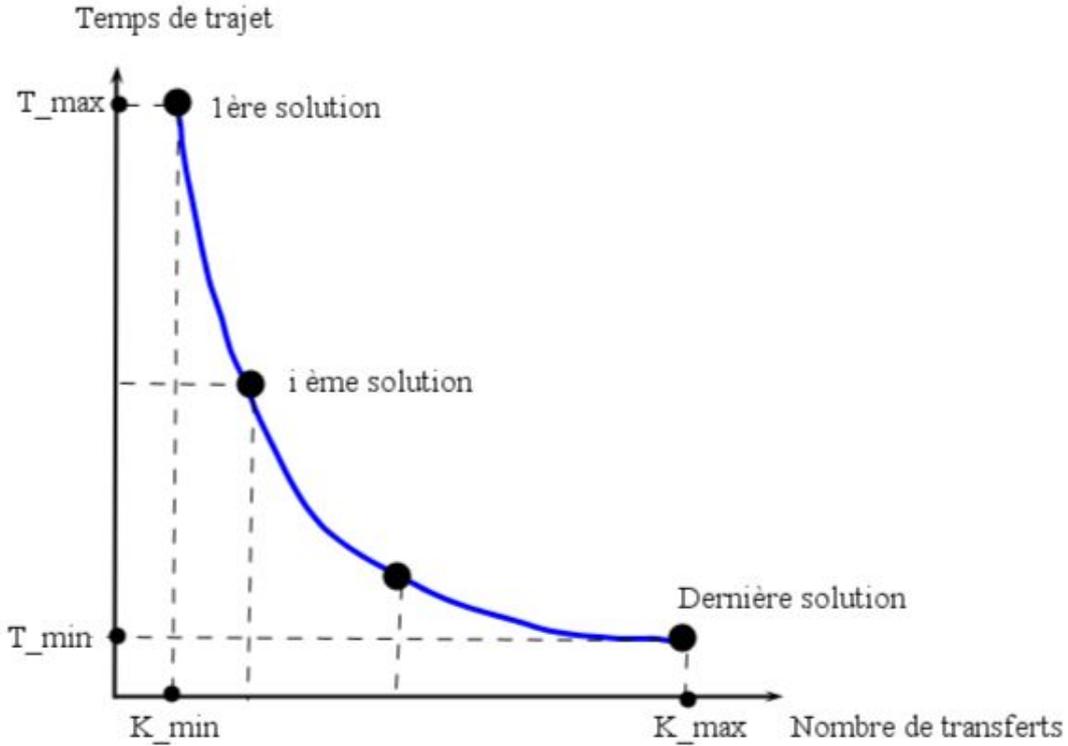


Figure II.6 – Courbe des solutions de l'algorithme TLS

5.2.1 Description de l'algorithme TLS

Au début de l'algorithme 6, les labels sont initialisés comme précédemment, Q^{now} contient le label $(O, s_0, 0)$, tandis que $Q^{next} = \emptyset$.

A chaque itération, le label de coût minimum (i, s, k) est extrait de Q^{now} et l'algorithme détermine les labels successeurs non dominés (j, s', k') avec j successeur de i et $s' := \delta(m_i, m_j, s)$,

puis l'algorithme insère ce label dans Q^{now} si $k = k'$ (cas où $m_i = m_j$) ou dans Q^{next} si $k' = k + 1$ (cas où $m_i \neq m_j$)

Dès que la destination D est dépilée de Q^{now} avec un état $s \in F$ (le chemin à k transfert(s) est trouvé) ou lorsque $Q^{now} = \emptyset$ (pas de solution à k transfert(s)), l'algorithme augmente le nombre de transferts de 1 et remplace Q^{now} par Q^{next} si Q^{next} est non vide pour chercher la solution à $k + 1$ transfert(s).

L'algorithme s'arrête lorsque Q^{now} est vide c'est-à-dire lorsqu'il n'existe plus de labels non dominés à k transfert(s), ou quand le nombre maximal K_{\max} des transferts est dépassé.

L'application de la règle de dominance de base utilisée pour décider de l'extension ou non d'un label est effectuée en $O(1)$. En effet pour un label (j, s', k') donné, issu d'une extension d'un label (i, s, k) , nous avons besoin de ne conserver que le chemin le plus court pour atteindre (j, s', k'') avec $k'' \leq k'$ transfert(s) (noté $lastlabel_{js'}$). Ainsi le label est dominé si $t_{js'}^{k'} \geq lastlabel_{js'}$ puisque le label précédemment rencontré ne peut pas avoir plus de transferts. Dans le cas contraire, c'est-à-dire si $t_{js'}^{k'} < lastlabel_{js'}$ alors le label est non dominé et l'algorithme met à jour la valeur $lastlabel_{js'}$ ($lastlabel_{js'} = t_{js'}^{k'}$).

Pour l'application de la règle de dominance entre états, un label (j, s', k') est dominé si $lastlabel_{j,s''} > t_{is''}^k + d_{ij}$ pour tout état s'' vérifiant $\forall s'' \preceq s'$. La complexité de cette règle est en $O(|S|)$.

5.2.2 Complexité de l'algorithme TLS

La complexité de l'algorithme TLS dépend de la structure de données utilisée pour mémoriser les labels. Comme pour l'algorithme de Dijkstra multi-modal, nous avons utilisé deux tas binaires. Pour chaque valeur du nombre de transfert(s) k , au plus $n \cdot |S|$ labels vont être explorés. Pour chacun de ces labels il y a deux opérations :

- (a) retrait du label (i, s, k) de Q_{now}
- (b) pour chacun des successeurs de (i, s, k) , application des règles de dominance puis ajout dans Q_{next} .

Q_{now} étant un tas binaire, l'opération (a) nécessite au pire $O(\log(n \cdot |S|))$. Pour l'opération (b), il y a $|Succ(i)|$ successeurs pour le nœud i du label (i, s, k) . Les règles de dominance sont en $O(1)$ pour la première et en $O(|S|)$ pour celle basée sur les états. L'insertion dans Q_{next} , qui est un tas binaire, nécessite au pire $O(\log(n \cdot |S|))$. Ainsi, l'opération (b) nécessite au pire $O(|Succ(i)| \cdot (|S| + \log(n \cdot |S|)))$ opérations pour chaque label (i, s, k) .

II.5 Algorithmes proposés pour le BI-MM-V-SPP

On note que le temps de calcul au pire de l'opération (b) est supérieur à celui de l'opération (a) qui est négligé par la suite.

Ainsi, la complexité de l'algorithme TLS pour chaque valeur de k est au pire en $O(m \cdot |S| \cdot (|S| + \log(n \cdot |S|)))$. D'où la complexité de l'algorithme global : $O(K_{max} \cdot m \cdot |S| \cdot (|S| + \log(n \cdot |S|)))$.

Par ailleurs, la complexité sans dominance est $O(K_{max} \cdot m \cdot |S| \cdot (\log(n \cdot |S|)))$.

Algorithme 6 Algorithme "Topological label-setting" (TLS)

ENTRÉES: $G(N, E)$, O , D , $d_{ij}, \forall (i, j) \in E$, K_{max}, t_0

- 1: $Q^{now} := \{(O, s_0, 0)\}$, $t_{O, s_0}^0 := 0$, $pred_{O, s_0}^0 := (O, 0, 0)$, $t_{is}^k := \infty, \forall i \in V \setminus \{O\}, \forall s \in S, \forall k = 0, \dots, K_{max}$
 - 2: $Q^{next} := \emptyset$, $lastlabel_{i, s} = \infty, \forall i \in V \setminus \{O\}, \forall s \in S$
 - 3: $k := 0$
 - 4: **tant que** $Q^{now} \neq \emptyset$ et $k \leq K_{max}$ **faire**
 - 5: **répéter**
 - 6: $(i, s, k) := argmin\{t_{j, s'}^k | (j, s', k) \in Q^{now}\}$ et $Q^{now} := Q^{now} \setminus \{(i, s, k)\}$
 - 7: **si** $(i \neq D$ or $s \notin F)$ et $t_{is}^k < lastlabel_{i, s}$ **alors**
 - 8: $lastlabel_{is} := t_{is}^k$
 - 9: **pour** $j \in Succ(i)$ **faire**
 - 10: $s' := \delta(m_i, m_j, s)$
 - 11: **si** $s' \neq -1$ et $\forall s'' \preceq s', lastlabel_{j, s''} > t_{is}^k + d_{ij}$ **alors**
 - 12: **si** $m_i = m_j$ **alors**
 - 13: $t_{j, s'}^k := t_{is}^k + d_{ij}$, $p_{j, s'}^k := (i, s, k)$ et $Q^{now} := Q^{now} \cup \{(j, s', k)\}$
 - 14: $lastlabel_{j, s'} := t_{j, s'}^k$
 - 15: **sinon si** $m_i \neq m_j$ et $k + 1 \leq K_{max}$ **alors**
 - 16: $t_{j, s'}^{k+1} := t_{is}^k + d_{ij}$, $pred_{j, s'}^{k+1} := (i, s, k)$ et $Q^{next} := Q^{next} \cup \{(j, s', k + 1)\}$
 - 17: **finsi**
 - 18: **finsi**
 - 19: **fin pour**
 - 20: **finsi**
 - 21: **jusqu'à** $Q^{now} = \emptyset$ ou $(i = D$ et $s \in F)$
 - 22: **si** $i = D$ et $s \in F$ **alors**
 - 23: (Plus court chemin à k transfert(s)).
 - 24: $k := k + 1$, $Q^{now} := Q^{next}$ et $Q^{next} := \emptyset$
 - 25: **finsi**
 - 26: **fin tantque**
-

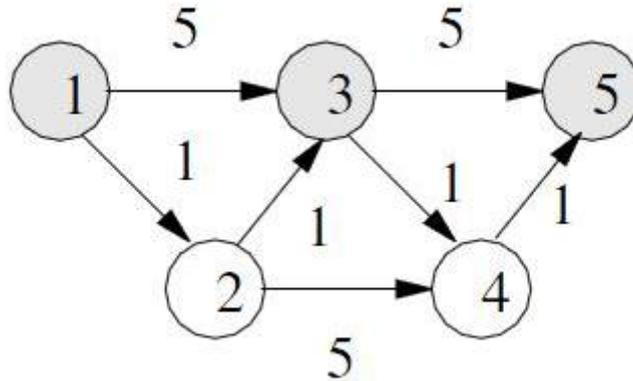


Figure II.7 – Exemple applicatif

5.2.3 Exemple illustratif

Afin d’illustrer le fonctionnement de l’algorithme TLS, on considère le problème de plus courts chemins multimodaux bi-objectif du nœud 1 au nœud 5 sur le graphe de la figure II.7. Ce graphe comporte 2 modes représentés en grisé (nœuds 1, 3 et 5) et en blanc (nœuds 2 et 4) et 4 arcs de transferts permettant de passer d’un mode à l’autre.

Pour simplifier, nous ne considérons pas d’états sur les nœuds (c’est-à-dire il n’y a pas de contraintes de viabilité).

Les principales étapes de l’algorithme **TLS** sont décrites ci dessous.

- **Itération 1** : $k = 0$
 - $i = 1, Q^{now} = \{3\}, t_3^0 = 5, Q^{next} = \{2\}, t_2^1 = 1$
 - $i = 3, lastlabel_3 = 5, Q^{now} = \{5\}, t_5^0 = 10, Q^{next} = \{2, 4\}, t_4^1 = 6$
 - $i = 5, lastlabel_5 = 10$ (Plus court chemin à 0 transfert) : $1 - > 3 - > 5$
- **Itération 2** : $k = 1$
 - $Q^{now} = \{2, 4\}, t_2^1 = 1, t_4^1 = 6, Q^{next} = \emptyset$
 - $i = 2, lastlabel_2 = 1, Q^{now} = \{4\}, Q^{next} = \{3\}, t_3^2 = 2$
 - $i = 4, lastlabel_4 = 6, Q^{now} = \emptyset, Q^{next} = \{3, 5\}, t_5^2 = 7$
 - $Q^{now} = \emptyset$: pas de plus court chemin à 1 transfert
- **Itération 3** : $k = 2$
 - $Q^{now} = \{3, 5\}, t_3^2 = 2, t_5^2 = 7$
 - $i = 3, lastlabel_3 = 2, Q^{now} = \{5\}, Q^{next} = \{4\}, t_4^3 = 3$

$i = 5$, $lastlabel_5 = 7$ (plus court chemin à 2 transferts) : $1- > 3- > 4- > 5$

– **Itération 4** : $k = 3$

$Q^{now} = \{4\}$, $t_4^3 = 3$

$i = 4$, $lastlabel_4 = 3$, $Q^{now} = \emptyset$, $Q^{next} = \{5\}$, $t_5^4 = 4$ $Q^{now} = \emptyset$: pas de plus court chemin à 3 transfert

– **Itération 5** : $k = 4$

$Q^{now} = \{5\}$, $t_5^4 = 4$

$i = 5$ $lastlabel_5 = 4$ (plus court chemin à 4 transferts) $1- > 2- > 3- > 4- > 5$

– **Itération 6** : $k = 5$

$Q^{now} = \emptyset$ pas de plus court chemin à 5 transferts

Ainsi l’algorithme a permis de déterminer 3 solutions non dominées pour le plus court chemin du nœud 1 au nœud 5 : un chemin à 0 transfert de coût 10, un chemin à 2 transferts de coût 7 et un chemin à 4 transferts de coût 4. Il stoppe car il n’y a plus de labels à traiter dans Q_{now} même si la limite du nombre de transferts posée par l’utilisateur est supérieure.

5.2.4 Discussion et limites de l’algorithme

L’algorithme TLS améliore l’algorithme de Dijkstra multimodal (algorithme 5) en réduisant la redondance dans la recherche. Par rapport à l’algorithme topologique original de Lozano et Storchi [62], TLS apporte quelques améliorations. Il s’agit comme déjà précisé de la réduction de l’automate de viabilité, de l’amélioration des règles de dominance, et une correction d’erreur sur la gestion des labels. En effet dans [62], une seule valeur $t_j^{s'}$ est utilisée pour stocker les meilleures valeurs trouvées pour toute extension (j, s') placée dans Q_{now} ou dans Q_{next} . Il s’ensuit qu’un chemin partiel à k transfert(s) pourrait ne pas être étendu parce qu’il a une valeur plus grande qu’un chemin de Q

à $k+1$ transfert(s). Nous avons corrigé cette erreur en différenciant les valeurs $t_{j,now}^{s'}$ et $t_{j,next}^{s'}$ comme dans l’algorithme original de Pallottino et Scutella [83].

Cet algorithme possède un inconvénient lié à la difficulté d’en proposer une version bidirectionnelle efficace. En effet, l’obtention des solutions par valeurs croissantes du nombre de transferts pose des problèmes lors de la connexion des recherches en avant et en arrière. On ne peut se limiter à des connexions entre chemins partiels à un même nombre de transferts. Pour pallier ce défaut, nous proposons un nouvel algorithme pouvant s’étendre plus aisément en bidirectionnel.

5.3 Algorithme Multi-labels-Multi-Heaps : MLMH

A l'inverse de l'algorithme **TLS** présenté ci-dessus (Algorithme 6) qui détermine l'ensemble des solutions non-dominées inférieures ou égales à K_{max} transferts dans l'ordre croissant du nombre de transferts, l'algorithme proposé **MLMH (Multi Labels Multi Heaps)** est un algorithme qui calcule les plus courts chemins non dominés dans l'ordre croissant du temps de trajet. Contrairement à l'algorithme **TLS** qui utilise deux files de priorité Q^{now} et Q^{next} , **MLMH** construit progressivement une liste $\mathcal{Q} = \{Q_0, Q_1, \dots\}$ de files de priorité telle que chaque file $Q_k \in \mathcal{Q}$ contient les labels des chemins à k transfert(s). Tous les labels (i, s, k) non dominés sont stockés, il est donc multi labels et, comme il utilise plusieurs de file priorité, il est donc multi-files d'où son nom en anglais multi-labels multi-heap (**MLMH**).

La particularité de cet algorithme est que le prochain label de coût minimal à étudier est recherché, non pas dans une seule file, mais dans toutes les files $Q_k \in \mathcal{Q}$ générées progressivement durant le parcours de l'algorithme. De plus, contrairement à **TLS** où les solutions non dominées sont retournées de manière croissantes sur le nombre de transferts, dans **MLMH**, elles sont décroissantes par rapport au nombre de transferts. Ainsi la première solution est minimale en temps mais maximale en nombre de transferts. Comme précédemment, on peut représenter ces solutions suivant une courbe de Pareto (figure II.8).

5.3.1 Description de l'algorithme MLMH

Le principe de l'algorithme (cf algorithme 7) est le suivant. L'algorithme part du label (i, s, k) de plus petit coût dans les files Q_k avec $k = 1 \dots |\mathcal{Q}|$, puis détermine les labels successeurs non dominés (j, s', k') et insère ces labels dans la file qui correspond au nombre de transferts permettant d'accéder à ce label. Si $k' \leq |\mathcal{Q}|$, (j, s, k') est inséré dans $Q_{k'}$ sinon on crée un nouveau élément $Q_{k'} = \{(j, s', k')\}$ de la liste \mathcal{Q} .

Lorsqu'un chemin VP_{OD}^k a été trouvé avec un nombre de transfert(s) k , l'algorithme supprime dans la liste \mathcal{Q} toutes les files Q_h telles que $h \geq k$ car les solutions possibles en utilisant ces files sont dominées par la solution trouvée avec k transferts.

L'algorithme détermine successivement des chemins dont le nombre de transferts diminue et dont le coût augmente.

L'algorithme s'arrête lorsque l'ensemble \mathcal{Q} est vide.

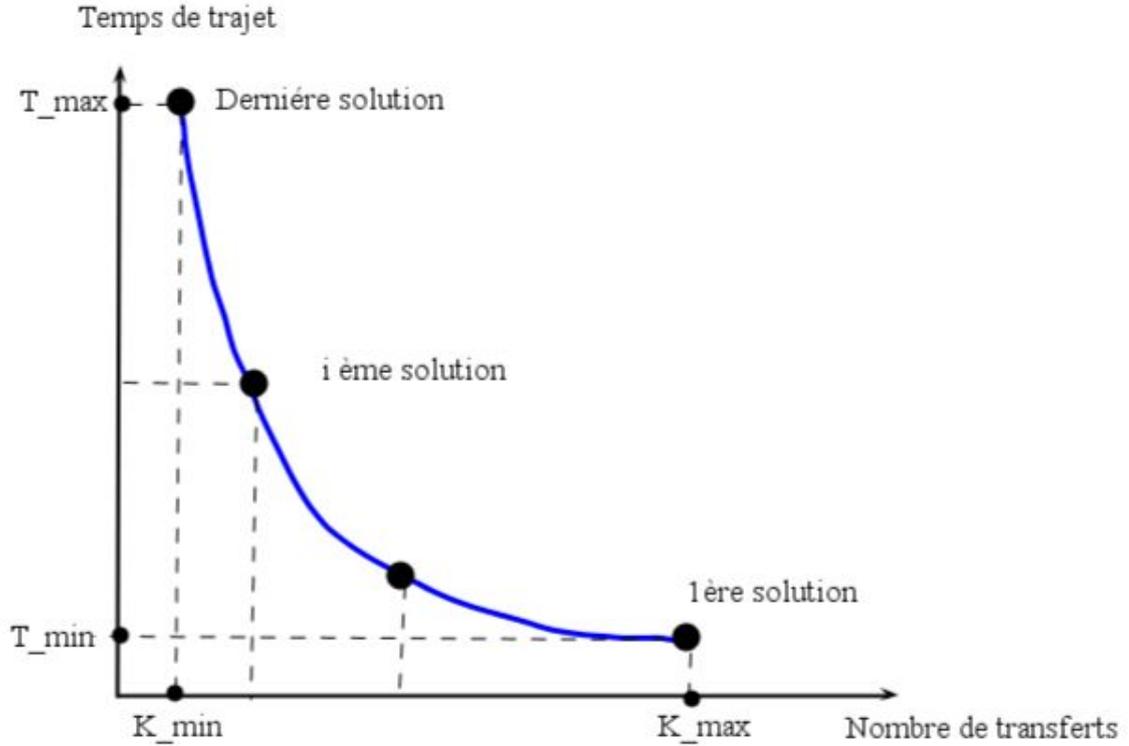


Figure II.8 – Courbe des solutions de l'algorithme MLMH

La règle de dominance de base utilisée dans le cas de **MLMH** pour l'extension ou non d'un label donné est utilisée au pire en $O(K_{\max})$. En effet pour un label (j, s', k') donné, nous avons besoin de conserver tous les labels non dominés (j, s', k'') avec $k'' \leq k'$. Ainsi le label (j, s', k') est dominé s'il existe $k'' \leq k'$ tel que $t_{js'}^{k''} \leq t_{js'}^{k'}$.

Par ailleurs, la règle de dominance entre états s'applique de la même façon dans l'algorithme **MLMH** que dans l'algorithme **TLS** (mais en $O(K_{\max}|S|)$).

5.3.2 Complexité de l'algorithme MLMH

La complexité de l'algorithme **MLMH** dépend de la structure employée pour mémoriser les labels. Nous utilisons ici des tas binaires pour chaque file $Q_k \in \mathcal{Q}$. Dans l'ensemble de ces files, $K_{\max}n|S|$ labels sont stockés, et extraits au pire. A chaque itération on effectue les opérations :

- (a) de recherche du label de coût minimum dans les K_{\max} files,
- (b) de suppression en $O(\log n|S|)$, et
- (c) de parcours et d'extension des labels successeurs. Pour chaque label successeur une règle de dominance (c1) peut être suivie par une opération d'insertion dans la file appropriée (c2). La règle de dominance de base peut être faite ici, en au plus K_{\max} opérations car tous les

labels (j, s', k'') avec $k'' \leq k'$ doivent être vérifiés (voir ci-dessus). Ainsi, l'opération (c) est de complexité au pire $O(|Succ(i)|(K_{\max} + \log(n|S|)))$.

En tenant compte des opérations (a) et (b), on obtient une complexité sans règle de dominance de :

$$O(K_{\max}|S|(nK_{\max} + n \log n|S| + mK_{\max} + m \log n|S|)) = O(K_{\max}|S|m(K_{\max} + \log n|S|)).$$

En ajoutant la règle de dominance entre états qui s'effectue en $O(K_{\max}|S|)$, La complexité devient alors :

$$O(K_{\max}|S|m(K_{\max}|S| + \log n|S|))$$

5.3.3 Exemple de fonctionnement de l'algorithme

Considérant l'exemple de la figure II.7, nous montrons ci-après l'exécution étape par étape de cet algorithme pour lequel le plus court chemin se trouve avec moins d'itérations par rapport à l'algorithme **TLS**.

– **Itération 1**

$$Q_0 = \{1\}$$

$$i = 1, k = 0, Q_0 = \{3\}, t_3^0 = 5, Q_1 = \{2\}, t_2^1 = 1$$

$$i = 2, k = 1, Q_1 = \{4\}, t_4^1 = 6, Q_2 = \{3\}, t_3^2 = 2$$

$$i = 3, k = 2, Q_2 = \{5\}, t_5^2 = 7, Q_3 = \{4\}, t_4^3 = 3$$

$$i = 4, k = 3, Q_3 = \emptyset, Q_4 = \{5\}, t_5^4 = 4$$

$$i = 5, k = 4, Q_4 = \emptyset \text{ (Plus court chemin à 4 transferts : } 1- > 2- > 3- > 4- > 5)$$

– **Itération 2**

$$i = 3, k = 0, Q_0 = \{5\}, t_5^0 = 10$$

$$i = 4, k = 1, Q_1 = \emptyset$$

$$i = 5, k = 2, Q_2 = \emptyset \text{ (Plus court chemin à 2 tranferts : } 1- > 3- > 4- > 5)$$

– **Itération 3**

$$i = 5, k = 0, Q_0 = \emptyset \text{ (Plus court chemin 0 tranfert : } 1- > 3- > 5)$$

Algorithme 7 Algorithme Multi-label/Multi-Heap (MLMH)

ENTRÉES: $G(N, E)$, O , D , $d_{ij}, \forall (i, j) \in E$, K_{\max}

- 1: $\mathcal{Q} = \{Q_0 := \{(O, s_0, 0)\}\}$, $t_{O, s_0}^0 := 0$, $pred_{O, s_0}^0 := (O, s_0, 0)$, $t_{i, s}^0 := \infty, \forall i \in V, \forall s \in S, (i, s) \neq (O, s_0)$
 - 2: $K = K_{\max}$
 - 3: **répéter**
 - 4: $(i, s, k) := \operatorname{argmin}\{t_{i', s'}^{k'} | k' = 0 \dots K \text{ et } (i', s', k') \in Q_{k'}\}$
 - 5: $Q_k := Q_k \setminus \{(i, s, k)\}$
 - 6: **si** $i = D$ et $s \in F$ **alors**
 - 7: stocker $t_{i, s}^k$ et $p_{i, s}^k$ (Plus court chemin à k transfert(s)).
 - 8: Supprimer $Q_{k'}$ pour tout $k' \geq k$.
 - 9: $K := k - 1$
 - 10: **sinon**
 - 11: **pour** $j \in \operatorname{Succ}(i)$ **faire**
 - 12: $s' := \delta(m_i, m_j, s)$
 - 13: **si** $m_i = m_j$ **alors**
 - 14: $k' = k$
 - 15: **sinon**
 - 16: $k' = k + 1$
 - 17: **fin**
 - 18: **si** $k' \leq K$ et $s' \neq -1$ et $\forall s'' \preceq s', \forall k'' \leq k', t_{j, s''}^{k''} > t_{i, s}^k + d_{ij}$ **alors**
 - 19: $t_{j, s'}^{k'} := t_{i, s}^k + d_{ij}$, $pred_{j, s'}^{k'} := (i, s, k)$ et $Q_{k'} := Q_{k'} \cup \{(j, s', k')\}$
 - 20: **fin**
 - 21: **fin pour**
 - 22: **fin**
 - 23: **jusqu'à** $K < 0$ or $\mathcal{Q} = \emptyset$
-

5.4 Algorithme bidirectionnel basé sur MLMH : FB-MLMH

Dans les algorithmes décrits précédemment, la recherche des chemins optimaux est unidirectionnelle et débute à partir du nœud origine. L'approche bidirectionnelle présentée au chapitre précédent divise la recherche en deux procédures séparées la recherche avant et la recherche arrière. Nous proposons dans cette section une adaptation de l'algorithme **MLMH** présenté en 5.3 en utilisant le principe de la recherche bidirectionnelle.

Dans cet algorithme, on considère indépendamment le nœud origine et le nœud destination comme points de départ. Ainsi on effectue simultanément deux phases de calculs d'itinéraires se déroulant chacune de la même manière que **MLMH** :

- la phase de recherche avant cherche les plus courts chemins à partir du nœud origine O dans le graphe initial G .
- la phase de recherche arrière considère le graphe inverse G^{-1} . Elle débute son exécution

à partir du nœud destination D et cherche les plus courts chemins à rebours.

L'algorithme bidirectionnel multimodal **FB-MLMH** considère deux listes de files de priorité \mathcal{FQ} et \mathcal{BQ} respectivement pour la phase de recherche avant et de recherche arrière, de telle sorte que FQ_k contienne les labels « avant » dans lesquels $ft_{i,s}^k$ est le coût du chemin $(VP_{O_i}^{ks})_f$ et BQ_k contienne les labels « arrière » avec $bt_{i,s}^k$ correspondant au coût du chemin $(VP_{D_i}^{ks})_b$.

Chaque étape de recherche avant ou arrière opère de manière identique à **MLMH**. Toutefois, il faut spécifier deux particularités. La première est liée à la modélisation des contraintes de viabilité pour la recherche arrière, la deuxième concerne la connexion des labels avants et des labels arrières.

5.4.1 Modélisation de l'automate de viabilité pour la phase de recherche arrière

Contrairement à la recherche avant, pour la phase de recherche arrière, on ne connaît pas l'état du chemin à la destination D (s_0 dans la recherche avant). Afin de respecter la viabilité des chemins dans cette phase, nous définissons ci-dessous deux types de modèle de l'automate de viabilité pour la phase de recherche arrière.

Le premier modèle utilise l'automate inverse de celui utilisé pour la phase de recherche avant. Pour cela, l'ensemble des états finaux de l'automate initial correspond à l'ensemble des états initiaux de l'automate inverse et les transitions entre états sont inversées. L'automate résultant (partie gauche figure II.9) est non-déterministe. A partir d'un état s , une même transition peut aboutir à plusieurs états distincts.

Sur cet automate, un initial initial (s_5) représentant l'état à la destination est introduit. De cet état, on peut aboutir en marchant ou en prenant le bus à l'état s_1 ou s_4 (les états finaux de l'automate de la recherche avant). La fonction de transition notée $\delta^{-1}(m_i; m_j; s)$ donne ainsi un ensemble d'états possibles. Cela signifie pour l'exemple ci-dessus que si on arrive en marchant à la destination il est possible que le métro ait été pris auparavant (état s_4) ou non (état s_1). En pratique, à chaque fois qu'une extension de label utilise une transition qui donne plusieurs états successeurs possibles (dans le chemin vers l'arrière), tous les labels correspondants sont générés. Cet indéterminisme peut produire des labels qui peuvent ne jamais parvenir à l'origine, induisant des calculs inutiles.

Le deuxième modèle que nous proposons utilise un automate déterministe. Nous avons obtenu cet automate "à la main" par des considérations de viabilité d'un chemin pris à rebours.

II.5 Algorithmes proposés pour le BI-MM-V-SPP

Cette opération peut être effectuée de manière automatique par des algorithmes dédiés mais pour un automate non déterministe à n états, on ne peut parfois pas obtenir d'automate déterministe équivalent à moins de 2^n états [50, 51].

L'automate déterministe obtenu dans notre exemple est défini par la figure II.9 (partie droite) dans lequel les états représentent :

- e_0 l'état initial à la destination ;
- e_1 marche ou bus avant d'arriver à la destination ;
- e_2 la voiture a été prise et pas encore quitté ;
- e_3 le métro a été pris et n'a pas encore été quitté ;
- e_4 marche ou bus avant de prendre le métro ;

Les états finaux de cet automate sont e_1 , e_2 et e_4 (états possibles à l'origine).

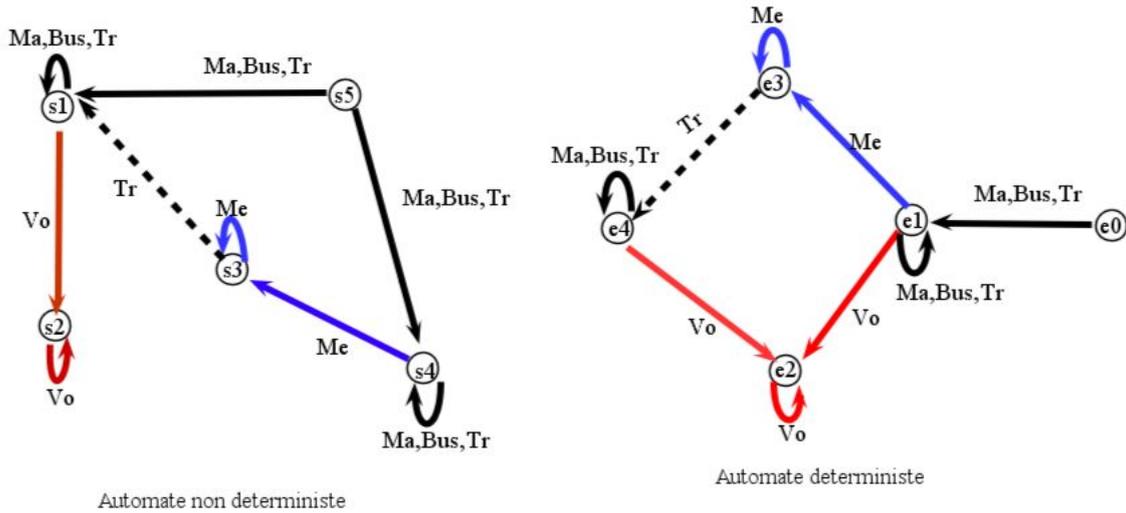


Figure II.9 – Graphe d'états non déterministe et déterministe pour la recherche arrière

Dans la suite, on notera S les états de l'automate de la recherche avant et S^B les états de l'automate de la recherche arrière. s_D correspond à l'état à la destination pour la recherche arrière, soit $s_D = e_0$ pour l'automate déterministe et $s_D = s_5$ pour l'automate non déterministe. La fonction de transition de l'automate arrière est notée $\delta^{-1}(m, m', s)$

5.4.2 Connexion des labels avant et arrière

Le deuxième point à résoudre pour concevoir l'algorithme bidirectionnel survient lors des connexions entre des labels avant et arrière pour déterminer si le chemin obtenu est optimal ou non. En effet, le chemin VP_{OD}^{ks} résultant de la connexion au nœud j des labels avant et arrière, notés respectivement $(VP_{O_j}^{hs'})_f$ et $(VP_{D_j}^{qs''})_b$, avec $k = h + q$, n'est pas toujours viable. Pour

Chapitre II. Modélisation et méthodes de résolution proposées

étudier la connexion des chemins, nous considérons deux types de connexions : une connexion obtenue en recherche avant, appelée connexion avant et une connexion obtenue en recherche arrière, appelée connexion arrière.

La connexion avant (respectivement arrière) est la construction d'un chemin de l'origine à la destination à partir du chemin $(VP_{O_j}^{hs'})_f$ (resp. $(VP_{D_j}^{qs''})_b$) obtenu lors de la phase de recherche avant (resp de la phase de recherche arrière) avec un chemin obtenu en arrière $(VP_{D_j}^{qs''})_b$ (resp un chemin d'avant $(VP_{O_j}^{hs'})_f$). Afin que ces sous chemins puissent être connectés pour former un chemin viable, il faut que leurs labels soient compatibles. En d'autres termes, si la connexion est faite dans la recherche avant, les labels (j, s', h) et (j, s'', q) des chemins connectés vérifient que $s'' \in SC_f(s')$ (où $SC_f(s')$ est l'ensemble des états compatibles à s' dans l'automate de la recherche avant). Inversement on doit aussi vérifier que $s' \in SC_b(s'')$ avec $SC_b(s'')$ l'ensemble des états compatibles à s'' dans l'automate de la recherche arrière.

Dans le cas de l'automate non déterministe $SC_f(s) = s$, c'est-à-dire que la connexion se réalise au même état car les automates utilisés pour les recherches avant et arrière ont des états identiques.

Dans le cas déterministe, les états des automates avant et arrière n'étant pas identiques, chaque état s à son propre ensemble $SC_*(s)$. Ces ensembles sont représentés dans notre cas d'étude par les tableau suivants (tableaux II.1 et II.2) :

Dès qu'une connexion à $h + q$ transferts est trouvée lors de l'extension d'un label en avant

états s	Ensemble $SC_f(s)$
s_1	$\{e_1; e_4\}$
s_2	$\{e_2\}$
s_3	$\{e_3\}$
s_4	$\{e_1\}$

Tableau II.1 – Compatibilité entre états pour la connexion en recherche avant

ou en arrière, le chemin correspondant est mémorisé s'il améliore la plus petite des connexions trouvées pour $h + q$ transfert(s).

Pour l'algorithme bidirectionnel monomodal, la condition d'optimalité du chemin résultant d'une connexion obtenue par les étapes de la recherche avant et de la recherche arrière a été définie par Nicholson [79]. Cette condition considère la connexion (i.e. le chemin) de plus petit coût $\min_{i \in N} (\pi_i^f + \pi_i^b)$, un tel chemin est le chemin optimal entre l'origine O et la destination

II.5 Algorithmes proposés pour le BI-MM-V-SPP

états s	Ensemble $SC_b(s)$
e_1	$\{s_1; s_4\}$
e_2	$\{s_2\}$
e_3	$\{s_3\}$
e_4	$\{s_1\}$

Tableau II.2 – Compatibilité entre états pour la connexion en recherche arrière

D si la condition suivante définie par l'équation II.4 est vérifiée :

$$\min_i (\pi_i^f + \pi_i^b) \leq \min_{(i') \in Q_F} \pi_{i'}^f + \min_{(i') \in Q_B} \pi_{i'}^f,$$

autrement dit, si le coût du plus petit des chemins déjà obtenus est inférieur au coût de tout autre chemin susceptible de se connecter plus tard.

Dans le cas du multimodal, avec l'utilisation du principe de l'algorithme MLMH, on recherche d'abord le plus court chemin indépendamment du nombre de transferts. Ainsi il suffit de comparer la meilleure connexion trouvée avec une borne inférieure du plus court chemin, ce qui donne :

$$\min_{(i,s^f,h) \text{ et } (i,s^b,q) \text{ connectés}} (ft_{i,s^f}^h + bt_{i,s^b}^q) \leq \min_{(i,s^f,k) \in \mathcal{FQ}} ft_{i,s^f}^k + \min_{(i,s^b,k) \in \mathcal{BQ}} bt_{i,s^b}^k, \quad (\text{II.4})$$

et indique que le plus court chemin à $h + q$ transfert(s) a été trouvé. Dans ce cas on peut, comme dans l'algorithme MLMH, supprimer toutes les files avant et arrière d'au moins $h + q$ transfert(s).

5.4.3 Description de l'algorithme

L'algorithme 8 décrit les principales étapes de la recherche bidirectionnelle basée sur l'algorithme **MLMH**.

Les étapes 1-3 effectuent l'initialisation de la structure \mathcal{FQ} avec un seule file FQ_0 contenant le label $(O, s_0, 0)$ et de la structure \mathcal{BQ} avec BQ_0 contenant soit le labels $(D, e_0, 0)$ dans le cas déterministe. Il initialise aussi les valeurs des connexions minimales dans L^k .

La boucle principale calcule le label minimum parmi les labels avant (i^f, s^f, k^f) et arrière (i^b, s^b, k^b) (étape 5-6). L'algorithme commence le parcours par le label minimal parmi les (i, s, k) .

Selon la direction de recherche, on utilise l'une des procédures respectives (*ForwardSearch* (Algorithme 9) et *backwardSearch* (Algorithme 10) décrivant les recherches en avant et en arrière. Le label (i, s, k) de coût minimal est extrait de la file XQ_k (étape 12).

Dans chaque une de ces procédures, le parcours des successeurs, l'extension et la recherche de connexion sont effectués. Pour chaque successeur (j, s', k') , la règle de dominance est appliquée et le label est actualisé puis inséré dans la file correspondante avant de vérifier l'ensemble des connexions minimales avec les labels (j, s', k'') de la recherche inverse avec $k' + k'' \leq K$, qui peuvent donner un chemin $O - D$ de moins de K transfert(s). Si une telle connexion est établie, le temps de trajet du chemin obtenu est comparé au meilleur chemin $O - D$ déjà trouvé avec $k' + k''$ transfert(s) dont la valeur est stockée dans $L^{k'+k''}$. Si ce nouveau chemin est de coût inférieur à $L^{k'+k''}$, ce dernier est mis à jour.

A la fin de la procédure, le test d'optimalité de l'équation II.4 est effectué en comparant la plus petite des connexions trouvées L^{k^*} avec une limite inférieure donnée par la somme des plus petits labels avant et arrière ($\min_{(if, sf, kf) \in \mathcal{FQ}} ft_{if, sf}^{kf} + \min_{(ib, sb, kb) \in \mathcal{BQ}} bt_{ib, sb}^{kb}$) (lignes 22 à 28). Si la condition est vérifiée alors L^{k^*} est le plus court chemin à k^* transfert(s). Ainsi toutes les files de priorités FQ_h et BQ_h avec $h \geq k^*$ peuvent être supprimées car les labels qu'elles contiennent conduiraient à des solutions dominées.

5.4.4 Exemple de fonctionnement de l'algorithme

Nous reprenons l'exemple de la figure II.7 et nous montrons ci-après l'exécution étape par étape de l'algorithme bidirectionnel proposé

– **itération 1**

– Recherche avant : $(i, h) = (1, 0)$

$$ft_1^0 = 0 \quad FQ_0 = \{1\} \quad ; \quad ft_2^1 = 1 \quad FQ_1 = \{2\} \quad ; \quad ft_3^0 = 5 \quad FQ_0 = \{3\}$$

Pas de connexion avec les labels arrière

– Recherche arrière : $(i, q) = (5, 0)$

$$bt_5^0 = 0 \quad BQ_0 = \{5\} \quad ; \quad bt_4^1 = 1 \quad BQ_1 = \{4\} \quad ; \quad bt_3^0 = 5 \quad BQ_0 = \{3\}$$

$$\text{connexion à 0 transfert : } L_0 = ft_3^0 + bt_3^0 = 10$$

$$\text{condition non vérifié } L_0 = 10 > \min\{ft^0 i\} + \min\{bt^0 i_3\} = 2$$

– **itération 2**

– Recherche avant $(i, h) = (2, 1)$

$$ft_4^1 = 6 \quad FQ_1 = \{4\} \quad ; \quad ft_3^2 = 2 \quad FQ_2 = \{3\}$$

Algorithme 8 Algorithme Bidirectionnel Multi-Labels-Multi-Heaps (FB-MLMH)

ENTRÉES: $G(V, E)$, FA , BA , O , D , $d_{ij}, \forall (i, j) \in E$, k_{\max}

{Initialisations des labels forward et backward}

1: $\mathcal{FQ} = \{FQ_0 := \{(O, s_0, 0)\}\}$, $ft_{O,s_0}^0 := 0$, $fp_{O,s_0}^0 := (0, s_0, 0)$, $ft_{i,s}^0 := \infty, \forall i \in V, \forall s \in SF$,
 $(i, s) \neq (0, s_0)$

2: $\mathcal{BQ} = \{BQ_0 := \{(D, s_D, 0)\}\}$, $bt_{O,s_D}^0 := 0$, $fp_{O,s_D}^0 := (D, s_D, 0)$. $bt_{i,s}^0 := \infty, \forall i \in V$,
 $\forall s \in SB, i \neq 0$ or $s \neq s_D$.

{Initialisation connexion minimal }

3: $K = k_{\max}$ et set $L^k = \infty, \forall k, 0 \leq k \leq k_{\max}$

4: **répéter**

5: $(i^f, s^f, k^f) := \operatorname{argmin}\{ft_{i',s'}^{k'} | (i', s', k') \in \mathcal{FQ}\}$

6: $(i^b, s^b, k^b) := \operatorname{argmin}\{bt_{i',s'}^{k'} | (i', s', k') \in \mathcal{BQ}\}$

7: **si** $ft_{i^f,s^f}^{k^f} \leq bt_{i^b,s^b}^{k^b}$ **alors**

8: $(i, s, k) := (i^f, s^f, k^f)$ and $FQ_k := FQ_k \setminus \{(i^f, s^f, k^f)\}$

9: **si** $i = D$ et $s \in F^{FA}$ **alors**

10: $L^k := ft_{i,s}^k$ et $k^* := k$

11: **sinon**

12: ForwardSearch((i, s, k) , \mathcal{FQ} , \mathcal{BQ} , k^* , L)

13: **finsi**

14: **sinon**

15: $(i, s, k) := (i^b, s^b, k^b)$ et $BQ_k := BQ_k \setminus \{(i^b, s^b, k^b)\}$

16: **si** $i = O$ et $s \in F^{BA}$ **alors**

17: $L^k := bt_{i,s}^k$ et $k^* := k$

18: **sinon**

19: BackwardSearch((i, s, k) , \mathcal{FQ} , \mathcal{BQ} , k^* , L)

20: **finsi**

21: **finsi**

{Test d'optimalité }

22: $(i^f, s^f, k^f) := \operatorname{argmin}\{ft_{i',s'}^{k'} | (i', s', k') \in \mathcal{FQ}\}$

23: $(i^b, s^b, k^b) := \operatorname{argmin}\{bt_{i',s'}^{k'} | (i', s', k') \in \mathcal{BQ}\}$

24: **si** $k^* \neq -1$ et $L^{k^*} \leq ft_{i^f,s^f}^{k^f} + bt_{i^b,s^b}^{k^b}$ **alors**

25: stocke L^{k^*} le PCC à k^* transfert(s).

26: Supprimer tout FQ_k et BQ_k avec $k \geq k^*$.

27: $K := k^* - 1$

28: **finsi**

29: **jusqu'à** $K < 0$ ou $\mathcal{FQ} = \mathcal{BQ} = \emptyset$

connexion 2 transferts : $L_2 = ft_4^1 + bt_4^1 = 7$

condition non verifié $L_2 = 7 > 2 + 1$

– Recherche arrière $(i, q) = (4, 1)$

Algorithme 9 ForwardSearch($(i, s, k), \mathcal{FQ}, \mathcal{BQ}, k^*, L$)

```

1: pour  $j \in FS(i)$  faire
2:    $s' := \delta(m_i, m_j, s)$ 
3:   si  $m_i = m_j$  alors
4:      $k' = k$ 
5:   sinon
6:      $k' = k + 1$ 
7:   finsi
8:   si  $k' \leq K$  et  $s' \neq \emptyset$  et  $\forall k'' \leq k', s'' \preceq s', ft_{j,s''}^{k''} > ft_{i,s}^{k'} + d_{ij}$  alors
9:      $ft_{j,s'}^{k'} := ft_{i,s}^{k'} + d_{ij}$ ,  $p_{j,s'}^{k'} := (i, s, k)$  et  $FQ_{k'} := FQ_{k'} \cup \{(j, s', k')\}$ 
    {Connexion}
10:    minconnection :=  $\infty$ ;  $k^* = -1$ 
11:    pour  $(j, s^b, k'') \in \mathcal{BQ}$ ,  $k' + k'' \leq K$ ,  $s^b \in CS^{FA \rightarrow BA}(s')$  faire
12:      si  $L^{k'+k''} > ft_{j,s'}^{k'} + bt_{j,s^b}^{k''}$  alors
13:         $L^{k'+k''} := ft_{j,s'}^{k'} + bt_{j,s^b}^{k''}$ 
14:      si  $L^{k'+k''} < \text{minconnection}$  alors
15:        minconnection :=  $L^{k'+k''}$ ;  $k^* := k' + k''$ 
16:      finsi
17:    finsi
18:  fin pour
19: finsi
20: fin pour

```

$$bt_3^2 = 2 \quad BQ_2 = \{3\} \quad ; \quad bt_2^1 = 6 \quad BQ_1 = \{2\}$$

$$\text{connexion à 4 transferts : } L_4 = bt_3^2 + ft_3^2 = 4$$

$$\text{condition vérifiée } L_4 = 4 \leq 2 + 2$$

Solution à 4 transferts obtenue $1- > 2- > 3- > 4- > 5$

– **itération 3**

– Recherche avant : $(i, h) = (3, 2)$

$$ft_5^2 = 7 \quad FQ_2 = \{5\} \quad ; \quad ft_4^3 = 3 \quad FQ_3 = \{4\}$$

$$\text{connexion 2 transferts : } L_2 = ft_5^2 + bt_5^0 = 7$$

$$\text{condition non vérifiée } L_2 = 7 > 2 + 3$$

– Recherche arrière : $(i, q) = (3, 2)$

$$bt_3^3 = 3 \quad BQ_3 = \{2\} \quad ; \quad bt_1^2 = 7 \quad BQ_2 = \{1\}$$

$$\text{connexion à 2 transferts : } L_2 = bt_3^3 + ft_1^0 = 7$$

$$\text{condition vérifiée } L_2 = 7 \leq 3 + 3$$

– **itération 4**

– Recherche avant : $i = (4, 3)$ pas de successeur à développer

Algorithme 10 BackwardSearch($(i, s, k), \mathcal{FQ}, \mathcal{BQ}, k^*, L$)

```

1: pour  $j \in BS(i)$  faire
2:    $s' := \delta^{-1}(m_i, m_j, s)$ 
3:   si  $m_i = m_j$  alors
4:      $k' = k$ 
5:   sinon
6:      $k' = k + 1$ 
7:   finsi
8:   si  $k' \leq K$  et  $s' \neq \emptyset$  et  $\forall k'' \leq k', s'' \preceq s', bt_{j,s''}^{k''} > bt_{i,s}^{k'} + d_{ij}$  alors
9:      $bt_{j,s'}^{k'} := bt_{i,s}^{k'} + d_{ij}$ ,  $p_{j,s'}^{k'} := (i, s, k)$  et  $BQ_{k'} := BQ_{k'} \cup \{(j, s', k')\}$ 
    {Connexion}
10:    minconnection :=  $\infty$ ;  $k^* = -1$ 
11:    pour  $(j, s^f, k'') \in \mathcal{FQ}$ ,  $k' + k'' \leq K$ ,  $s^f \in CS^{BA \rightarrow FA}(s')$  faire
12:      si  $L^{k'+k''} > bt_{j,s'}^{k'} + ft_{j,s^f}^{k''}$  alors
13:         $L^{k'+k''} := bt_{j,s'}^{k'} + ft_{j,s^f}^{k''}$ 
14:        si  $L^{k'+k''} < \text{minconnection}$  alors
15:          minconnection :=  $L^{k'+k''}$ ;  $k^* := k' + k''$ 
16:        finsi
17:      finsi
18:    fin pour
19:  finsi
20: fin pour

```

connexion 2 transferts : $L_2 = ft_5^1 + bt_5^0 = 6$

condition non vérifiée $L_2 = 7 > 2 + 3$

– Recherche arrière : $(i, q) = (2, 3)$ pas de successeur à développer

connexion 2 transferts : $L_2 = ft_5^1 + bt_5^0 = 7$

condition non vérifiée $L_2 = 7 < 5 + 5$

Solution à 2 transferts obtenue $1- > 3- > 4- > 5$ Suppression de BQ_k, FQ_k avec $k \geq 2$

– **itération 5**

– Recherche avant : $(i, h) = (3, 0)$

$ft_5^0 = 10$ $FQ_0 = \{5\}$; $ft_4^1 = 6$ $FQ_1 = \{4\}$

connexion 0 transfert : $L_0 = 10$

condition vérifiée $L_0 = 10 > 5 + 6$

Solution à 0 transfert obtenue $1- > 3- > 5$

6 Variantes A^* des différents algorithmes

Sur la base du constat que le principe A^* peut améliorer de manière significative les algorithmes exploitant le principe de l'algorithme de Dijkstra, nous proposons des adaptations des algorithmes (**TLS**, **MLMH** et **FB-MLMH**). Les algorithmes résultants sont nommés respectivement A^* **TLS**, A^* **MLMH**, A^* **FB-MLMH**.

Les deux premiers suivent respectivement les mêmes procédures que **TLS** (**MLMH**) en changeant uniquement la fonction d'évaluation f pour y introduire une fonction heuristique basée sur la distance euclidienne et la vitesse maximale. En revanche l'algorithme bidirectionnel présente quelques différences que nous détaillons ci-après.

L'algorithme A^* bidirectionnel (A^* **FB-MLMH**), à la différence des algorithmes monodirectionnels basés sur A^* considère deux fonctions heuristiques :

- $h_f(j)$ estimant la distance d'un nœud j à la destination D pour l'étape de recherche avant
- $h_b(j)$ estimant la distance de l'origine O à un nœud j pour l'étape de recherche arrière

Dans le cas de l'algorithme **FB-MLMH**, la condition d'arrêt définie par l'équation II.4 doit être modifiée en tenant compte du rajout des estimations h_f et h_b .

La valeur d'un label (j^f, s^f, k^f) obtenu lors de la recherche avant est calculée de la manière suivante : $\tilde{f}t_{j^f, s^f}^{k^f} = ft(j^f, s^f, k^f) + h_f(j^f)$ où $ft(j^f, s^f, k^f)$ est la durée réelle de O à j^f . Et, la valeur d'un label (j^b, s^b, k^b) obtenu lors de la recherche arrière est $\tilde{b}t_{j^b, s^b}^{k^b} = bt(j^b, s^b, k^b) + h_b(j^b)$ où $bt(j^b, s^b, k^b)$ est la durée réelle de D à j^b .

Lorsqu'on obtient une connexion sur un sommet j par les labels (j^f, s^f, k^f) et (j^b, s^b, k^b) on ne peut sommer les valeurs des labels $\tilde{f}t_{j^f, s^f}^{k^f}$ et $\tilde{b}t_{j^b, s^b}^{k^b}$ pour le test d'optimalité comme on le faisait sans le principe A^* car cela reviendrait à compter deux fois l'estimation. Nous remplaçons alors la condition d'arrêt par la condition plus faible suivante qui considère uniquement l'estimation la plus forte entre les labels avant et arrière.

$$\min(ft_{i, s^f}^h + bt_{i, s^b}^q) \leq \max\left(\min_{(i', s', k') \in \mathcal{F}\Omega} \tilde{f}t_{i', s'}^{k'}, \min_{(i', s', k') \in \mathcal{B}\Omega} \tilde{b}t_{i', s'}^{k'}\right) \quad (\text{II.5})$$

Avec les conditions présentées ci-dessous, l'algorithme bidirectionnel A^* fournit tous les chemins non-dominés. Toutefois, la condition d'optimalité est peu efficace. Nous proposons ensuite une autre condition qui ne garantit pas non plus d'obtenir toutes les solutions non-dominées, mais qui se révèle être plus efficace dans la pratique. Cette condition utilise deux fois la fonction

d'estimation dans les deux parties de l'inégalité.

$$\tilde{f}t_{i,sf}^h + \tilde{b}t_{i,sb}^q \leq \min_{(i',s',k') \in \mathcal{FQ}} \tilde{f}t_{i',s'}^{k'} + \min_{(i',s',k') \in \mathcal{BQ}} \tilde{b}t_{i',s'}^{k'} \quad (\text{II.6})$$

Cette condition est correct lorsqu'un sommet est plus éloigné de l'origine alors il est plus prêt de la destination.

Plusieurs conditions ont été proposées pour limiter le nombre de labels développés par l'algorithme A^* -FB-MHLM :

- Condition 1. Borne supérieure du temps de trajet

Après extension du label d'un nœud i on dispose d'un label (j, s, k) : si la valeur du temps de trajet pour ce label (j, s, k) est supérieure au temps de trajet déjà trouvé avec moins de transferts : on ne le conserve pas. C'est-à-dire si $g_f(j, s, k) > g_f(j, s', k') + h_f(j) = ft_{j,s'}^{k'}$ avec $k' \leq k$ alors on n'étend pas (j,s,k) . On a la condition symétrique en arrière $g_b(j, s, k') > g_b(j, s, k) + h_b(j) = bt_{j,s}^{k'}$ pour tout $k' \leq k$.

- Condition 2. Connexion avec un label à 0 transfert

Après extension en avant du label d'un nœud i on dispose d'un label (j, s, k) . Supposons que ce label soit connecté avec un label $(j, s', 0)$ marqué obtenu en arrière (si les états s et s' sont compatibles), alors $(j, s', 0)$ étant fixé son cout est déterminé et c'est le plus petit possible pour 0 transfert. De plus, comme on ne conserve que les labels non dominés, cela veut dire que les labels (j, s', k') avec $k' \geq 0$ sont également fixés car ils ont un coût plus faible. Donc on n'a plus besoin de poursuivre l'extension du label (j, s, k) en avant car on ne pourra pas améliorer le coût de la connexion. Ainsi, si (j, s, k) obtenu en recherche avant est connecté avec $(j, s', 0)$ fixé obtenu en recherche arrière alors (j, s, k) n'est pas étendu.

La condition symétrique est appliquée suite à l'extension d'un label arrière.

- Condition 3 : Amélioration de l'estimation à la destination ou à l'origine

On généralise la condition 2 au cas où la connexion se fait avec un label ayant plus de 0 transfert. Après extension en avant du label d'un nœud i on dispose d'un label (j, s, k) : si ce label est connecté avec un label (j, s', k') marqué obtenu en arrière (et les états s et s' sont compatibles) alors on détermine la plus grande valeur de k' telle que (j, s', k') soit marqué et on note k^* cette valeur. La valeur $g_b(j, s', k^*)$ correspond alors au plus petit temps de trajet de j vers la destination D : c'est donc une nouvelle valeur pour $h_f(j)$. Cela permet d'actualiser la condition 1 en remplaçant la valeur $h_f(j)$ par cette nouvelle valeur : Si $g_f(j, s, k) + g_b(j, s', k^*) \geq L^l$ pour

tout $l \leq k + k^*$ alors on n'étend pas le label (j, s, k) La condition symétrique suite à l'extension d'un label arrière peut également être appliquée.

7 Algorithmes proposés pour le problème BI-MM-V-TD-SPP

Les algorithmes présentés précédemment sont des algorithmes s'appliquant uniquement sur des réseaux de transport statiques. Dans le cas de réseaux dépendants du temps, on cherche les plus courts chemins non dominés de l'origine O à la destination D à partir une date de départ t_0 . Les modes *Bus* et *Métro* sont généralement dépendant du temps c'est-à-dire que pour chaque arrêt ou station i de ces modes on dispose d'une liste de dates de départs $HD(i)$.

De plus, comme nous avons supposé que le réseau considéré est FIFO, on ne conserve donc que l'étiquette dont la date d'arrivée est la plus petite. Ainsi, les algorithmes dans le cas dépendant du temps conservent la même structure que ceux présentés dans le cas statique avec une étape de plus qui est la recherche d'horaires dans les tables $HD(i, j)$. Les algorithmes **DIJ-MM**, **TLS** et **MLMMH** peuvent être adaptés dans le cas dépendant du temps en **DIJ-MM-TD**, **TLS-TD** et **MLMMH-TD** uniquement en intégrant une procédure de recherche d'horaires.

Pour déterminer les durées des trajet en fonction des horaire, nous procédons de la manière suivante. Considérons un départ d'un nœud i vers un nœud j à la date t_i , la durée de trajet $d_{ij}(t)$ de l'arc (i, j) est égale à $t_j = d_{ij}(t_i)$ donnée par l'équation I.1 (chapitre I où t_j est la date d'arrivée à j). Pour déterminer t_j , il faut effectuer une recherche dans la table $HD(i, j)$. Ainsi, la recherche de cette valeur a une complexité au pire de $|HD(i)|$ pour un sommet i .

Pour réduire la complexité due à la recherche des horaires, nous avons créé un tableau d'horaires par minute pour chaque table $HD(i, j)$ noté $HD'(i, j)$. Plus précisément, on crée pour chaque arc (i, j) tel que $m_i = m_j \in M_{TD}$ un vecteur de taille $24 * 60$ (nombre de minutes dans une journée). Ainsi pour obtenir la date d'arrivée t_j au nœud j , on accède directement au vecteur $HD'(i, j)$ à la position $d_{ij}(t_i)$. Cet accès en $O(1)$ fait que les variantes dépendant du temps des algorithmes ont la même complexité au pire que dans le cas statique.

Cependant pour l'algorithme bidirectionnel, dans le cas dépendant du temps, on ne connaît pas la date d'arrivée t_D au nœud de destination D . Pour traiter ce problème, nous avons suivi la démarche proposée par Nannicini et al. [41]. La recherche arrière se base sur le graphe inverse

de $G = (N, E, D)$, noté $G^{-1} = G(N, E^{-1}, \lambda)$ où λ est une fonction fournissant une valuation pour les arcs en sens inverse.

Ainsi, chaque arc (j, i) de G^{-1} avec $m_i = m_j \in M_{TD}$ est estimé avec la fonction λ qui est une borne inférieure de la durée réelle du temps de trajet correspondant à cet arc. La fonction que nous avons utilisée fait correspondre à chaque arc (i, j) appartenant à un mode dépendant du temps une estimation basée sur la distance euclidienne et sur la vitesse maximale de trajet.

Lors du déroulement de l'algorithme bidirectionnel dans le cas dépendant du temps (FB-MLMH-TD), une fois une solution approchée trouvée suite à une connexion sur un nœud j , on peut reconstruire le chemin de j jusqu'à la destination D avec les valeurs exactes issues des tables horaires en utilisant la date du $g_f(j, s, k)$ pour trouver la valeur réelle de ce chemin dans le graphe G .

De même la façon que dans le cas statique, on peut utiliser dans le cas dépendant les conditions d'amélioration proposées pour l'algorithme bidirectionnel basées sur les bornes décrites à la section 6.

8 Conclusion

Nous nous sommes centrés dans ce chapitre sur la résolution d'un problème de plus courts chemins multimodaux viables bi-objectifs particulier. Le problème étudié est polynomial, il consiste à déterminer l'ensemble des solutions non dominées en termes de temps de trajet et de nombre de changements de modes entre une origine et une destination.

Nous avons, tout d'abord, présenté une modélisation par couches du réseau de transport multimodal où chaque mode de transport est modélisé par un ou plusieurs graphes et des arcs de transferts permettant de passer d'un mode à un autre.

De plus, nous avons proposé une modélisation de la viabilité des chemins par un automate fini.

Nous avons proposé plusieurs méthodes de résolution pour résoudre le problème de plus courts chemins multimodaux viables bi-objectif considéré en nous limitant dans un premier temps à des réseaux de transport statiques avant d'étendre ces méthodes au cas dépendant du temps.

Nous avons de plus défini des règles de dominance dans le but d'améliorer l'efficacité de nos méthodes. Nous avons également développé des variantes A^* de tous les algorithmes proposés

Chapitre II. Modélisation et méthodes de résolution proposées

et une variante bidirectionnelle pour l'un d'entre eux.

Pour le déroulement de l'algorithme bidirectionnel, un automate de viabilité utilisé dans la recherche arrière a dû être introduit ainsi qu'une nouvelle condition d'optimalité.

Nous allons dans le chapitre suivant décrire l'ensemble des résultats expérimentaux obtenus pour ces algorithmes en nous appuyant sur une partie du réseau de transport multimodal de la ville de Toulouse.

Chapitre III

Résultats expérimentaux

1 Introduction

Ce chapitre est consacré à l'évaluation expérimentale des algorithmes décrits dans le chapitre précédent pour résolution du problème **BI-MM-V-SPP (Bi-objective Multimodal Viable Shortest Path Problem)**. Nous proposons dans ce chapitre, entièrement dédié aux expérimentations numériques, de décrire les différents scénarios de tests et de comparer les algorithmes sur ces scénarios.

Nous décrivons tout d'abord le contexte général d'expérimentation. Ensuite, nous présentons les résultats numériques qui sont regroupés en deux parties suivant les types de réseaux de test : les réseaux statiques et ceux dépendant du temps.

La première partie concerne la comparaison des algorithmes pour un réseau de transport statique et également de l'apport des différentes règles de dominances établies dans le chapitre précédent, ainsi que de l'utilisation de A^*

La deuxième partie donne les résultats des tests pour le réseau de transport multimodal dépendant du temps, en utilisant uniquement la meilleure règle de dominance déduite des résultats du cas statique. L'évaluation des variantes intégrant A^* est également effectuée.

2 Contexte général d'expérimentation

Nous définissons dans cette partie l'ensemble des conditions des tests. Il s'agit de donner la description de l'environnement de test et des réseaux de transport considérés.

Comme dit précédemment, nous avons considéré dans nos expériences les modes suivants :

$M = \{Piéton, Voiture, Métro, Bus\}$. Une des raisons est que nous ne disposons pas des données pour d'autres modes (vélo, train, etc.). Nous imposons aux modes retenus les contraintes de viabilité modélisées par le graphe d'états de la figure II.4.

La voiture ne peut être quittée que sur des nœuds de stationnement, correspondant à des parkings (le stationnement sur la voirie n'est pas autorisé).

Le nombre de changements de mode autorisés K_{\max} est un des paramètres importants des algorithmes. Dans l'ensemble des scénarios testés, cette borne est fixée à 5 transferts ($K_{\max} = 5$).

L'utilisateur peut exclure certains modes qu'il ne souhaite pas utiliser pour son trajet. Dans nos scénarii de tests, nous conservons l'ensemble des modes.

3 Réseaux de transport de tests

Nous avons considéré deux types de réseaux de transport à partir de données réelles fournies par Tisséo (l'organisme d'exploitation des transports de la ville de Toulouse) pour les données de transport en commun et une base de données commerciale pour les données de la voirie.

Le premier réseau est un graphe non dépendant du temps (statique). Les caractéristiques de ce graphe concernant le nombre de nœuds, le nombre d'arcs des différents modes sont données par le tableau suivant (Tableau III.1).

Puisque le graphe est statique, les valuations des arcs dépendant du temps sont transformées

Modes	nœuds	Arcs
Bus	6 170	6 646
Metro	75	72
Voirie	56 774	146 280
Transfert	-	6 370
Parking	29	-
Total	63 048	159 368

Tableau III.1 – Caractéristiques du réseau statique

en constantes en utilisant des vitesses moyennes des véhicules concernés.

A partir de ce graphe, nous avons généré aléatoirement 100 couples origines-destinations répartis dans les zones périphériques de Toulouse pour avoir des trajets longs afin de voir mieux les différences entre les algorithmes.

Le second réseau est un graphe dynamique où le bus et le métro ont des arcs dont la durée est fonction du temps. Ces modes fonctionnent généralement en notion de courses et de mis-

sions (voir Annexe C). Pour ce réseau, les lignes de bus sont réparties en missions. Une mission correspond donc à une suite ordonnée d'arrêts de l'itinéraire et est composée d'un ensemble de courses. Une course décrit le déplacement d'un véhicule de transport public sur un itinéraire de la ligne. La course précise l'horaire de passage à chaque arrêt d'une ligne de bus, sous la forme d'une table horaire associée à chaque arrêt. Le nombre total d'horaires du réseau est de 129975. Le métro fonctionne en fréquence de $3min$ pour la journée de service.

Le tableau ci dessous (Tableau III.2) donne les caractéristiques de ce graphe.

Modes	nœuds	Arcs
Bus	9 384	9 047
Metro	75	72
Voirie	56 774	146 280
Transfert	-	28 017
Parking	29	-
Total	65 262	183 416

Tableau III.2 – Données du réseau dynamique

Le réseau dépendant du temps utilisé est le même que le réseau statique. Cependant l'intégration des horaires réels des bus a conduit à éclater une ligne de bus en utilisant la notion de missions (ensemble d'itinéraires associés à une même ligne) et de courses (association des horaires aux différentes missions). Ceci a entraîné, d'une part, une augmentation du nombre de nœuds et d'arcs pour le mode Bus et, d'autre part, une augmentation du nombre d'arcs de transfert.

3.1 L'environnement des tests

La machine utilisée a un processeur *2,47 Intel Xeon W3520 GHz avec 4Go de RAM sous Linux Fedora 11*. Les algorithmes sont codés en C++. Plusieurs types de tests ont été réalisés avec les différents algorithmes :

- des tests sur le graphe statique ;
- des tests sur le graphe dépendant du temps ;

Pour le graphe statique, nous avons évalué trois utilisations des règles de dominance :

- sans dominance :
- dominance de base (dominance de Pareto)

- dominance entre états basée sur la dominance de base et les notions de préférences entre états

Chaque algorithme est testé avec l'utilisation ou non du principe de la recherche A^* . A ce sujet, nous devons souligner qu'aucune des améliorations proposées dans la section 6 du chapitre précédent n'a permis d'accélérer la résolution. Nous n'avons donc pas utilisé ces conditions dans les expérimentations présentées.

Pour l'algorithme bidirectionnel multimodal, nous avons considéré deux types de tests qui sont fonction de l'automate de viabilité de la recherche en arrière :

- le cas où cet automate est déterministe (**FB-MLMH**)
- le cas où il est non déterministe (**FB-MLMH-ND**)

3.2 Métriques de performance

Pour évaluer les performances à la fois qualitative et quantitative des résultats de nos différentes méthodes de résolution, nous considérons différentes métriques de performance pour mesurer et comparer les algorithmes de résolution.

Les métriques suivantes sont utilisées pour évaluer les algorithmes :

- le temps de calcul (temps Cpu) : c'est le critère usuel en termes de mesure de l'efficacité (rapidité) d'un algorithme.
- le nombre de labels empilés dans la file de priorité ;
- le nombre de labels dépilés de la file de priorité ;
- le nombre de labels visités ;

4 Résultats des expérimentations dans le cas non dépendant du temps

Tous les algorithmes parviennent à résoudre optimalement toutes les instances. Les trajets calculés ont des durées minimales, moyennes et maximales en minutes respectivement de 120, 179 et 225 minutes. Le nombre de solutions non dominées est en moyenne de 5. Le nombre moyen de transferts est de 4. Ces résultats montrent ainsi tout d'abord qu'un nombre significatif de solutions de compromis peut être trouvé lorsqu'on considère simultanément un objectif de temps de trajet et un objectif lié à la multimodalité.

Les résultats des expérimentations sont présentés d'abord dans le cas sans dominance, puis

III.4 Résultats des expérimentations dans le cas non dépendant du temps

avec la règle de dominance de base et enfin la dominance entre états. Ces comparaisons sont effectuées en utilisant ou non le principe de la recherche A^* . L'ensemble des résultats est fournis en Annexe B.

4.1 Résultats des expérimentations sans A^*

Les tableaux III.3, III.4 et III.5 répertorient pour chaque type de dominance les résultats obtenus pour les différents métriques.

Chaque tableau présente les valeurs moyennes, sur les 100 trajets O-D testés, des temps de calcul, du nombre de labels dépilés, empilés et visités pour chaque algorithme.

	DIJ-MM	TLS	MLMH	FB-MLMH	FB-MLMH-ND
Temps cpu (ms)	4 582	2 603	2 644	2 245	2 273
# labels dépilés	1 468 170	555 900	555 899	386 391	409 516
# labels empilés	1 607 760	629 159	61 0193	430 994	456 657
# labels visités	3 783 750	1 428 760	1 428 760	1 016 180	1 076 450

Tableau III.3 – Résultats des algorithmes (sans règle de dominance)

	DIJ-MM	TLS	MLMH	FB-MLMH	FB-MLMH-ND
temps cpu (ms)	4 112	1 654	1703	1 600	1 654
# labels dépilés	1 325 430	376 339	377 119	283 721	306 942
# labels empilés	1 450 250	420 374	411 903	313 798	339 536
# labels visités	3 412 480	960 483	962 706	738 354	798 842

Tableau III.4 – Résultats des algorithmes (avec la règle de dominance de base)

	DIJ-MM	TLS	MLMH	FB-MLMH	FB-MLMH-ND
temps cpu (ms)	4 205	1587	1525	1 396	1 559
# labels dépilés	1 325 430	357 838	337 687	255 464	278 561
# labels empilés	1 450 250	399 459	368 021	281 770	307 381
# labels visités	3 412 480	912 819	857 944	660 971	721 159

Tableau III.5 – Résultats des algorithmes (avec la règle de dominance entre états)

4.1.1 Analyse des résultats en termes de temps de calcul et de nombre de labels

Les résultats obtenus montrent que l'extension directe de l'algorithme de Dijkstra au multimodal (**DIJ-MM**) n'est pas efficace par rapport aux autres méthodes proposées : les améliorations en termes de temps de calcul varient de 42,3% à 66,8% (selon les algorithmes considérés et selon l'emploi des règles de dominance). En termes de nombre de labels (dépilés, empilés ou visités), les variations sont de même nature.

La comparaison des deux autres algorithmes mono-directionnels TLS et MLMH montre qu'en l'absence de règle de dominance ou avec la règle de dominance simple, l'algorithme TLS est plus performant que l'algorithme MLMH (hausse du temps de calcul respective de 1,58% et de 2,96%) en revanche lorsqu'on emploie la règle de dominance basée sur les états, l'algorithme MLMH s'avère plus performant que l'algorithme TLS (diminution du temps de calcul de 3,91%). Les variations en termes de nombre de labels empilés ne sont pas similaires à celles mesurées en temps de calcul : sans dominance ou avec la dominance simple il y a plus de labels empilés avec l'algorithme TLS. Pour les autres nombres de labels, les résultats obtenus en temps de calcul sont confirmés.

Par ailleurs, nos expérimentations montrent que l'algorithme bidirectionnel FB-MLMH (avec l'automate de viabilité en recherche arrière déterministe) réalise les meilleures performances par rapport à tous les autres algorithmes sur l'ensemble des métriques.

Si on considère plus particulièrement le temps de calcul, l'algorithme FB-MLMH est le plus rapide mais l'amélioration obtenue varie en fonction des règles de dominance utilisées. Sans utiliser de règle de dominance, il améliore les temps de calcul de l'algorithme TLS de 2603 ms à 2245 ms, soit une diminution de 13,75%. Dans le cas de la dominance simple, l'amélioration est de 3,26%. Elle est et de 12,04% dans le cas de la dominance basée sur les états. De même, l'algorithme FB-MLMH améliore aussi l'algorithme MLMH dont il est dérivé. Ces améliorations sont respectivement de 15,09%, 6,05% et 8,46% dans les cas sans dominance, avec dominance de base et avec dominance basée sur les états.

L'utilisation d'un automate déterministe pour le phase de recherche arrière (FB-MLMH) améliore les performances de l'algorithme FB-MLMH-ND utilisant un automate non déterministe, de respectivement de 1,23%, 3,26% et 10,46% en temps de calcul selon la règle de dominance utilisée. Cette amélioration varie de 5 à 8% en termes de nombre de labels. L'impact des règles de dominance est analysé plus finement dans la section suivante.

4.1.2 Analyse des résultats en termes de règles de dominance

Les figures III.1 à III.4 permettent de visualiser l'influence des règles de dominance sur les différents algorithmes excepté DIJ-MM qui n'a pas été représenté en raison de ses mauvaises performances.

Elles décrivent respectivement les variations des temps de calcul, des nombres de labels empilés, dépilés et visités pour les différents algorithmes en fonction de l'utilisation des règles de dominance.

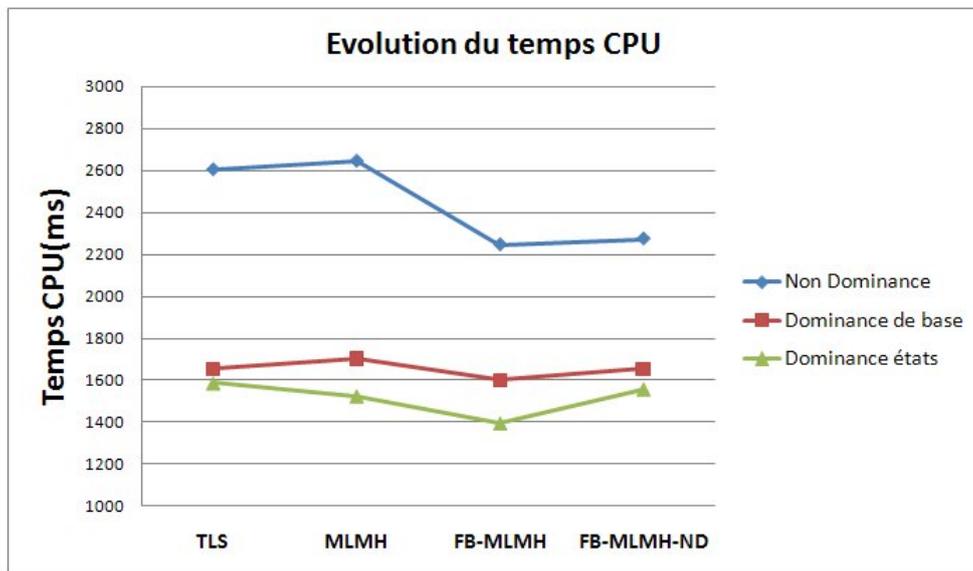


Figure III.1 – Comparaison des méthodes et des règles de dominances en termes de temps CPU

Ces figures mettent en évidence que l'utilisation des règles de dominance a un impact sur les différents algorithmes et améliorent leur performance en termes de temps de calcul et de nombre de labels

La règle de dominance avec états donnent les meilleurs résultats dans les différentes métriques (sauf pour DIJ-MM dans lequel la règle de dominance basée sur les états s'avère moins performante que la règle de dominance simple). Par rapport à l'absence de règle de dominance, la dominance simple apporte un gain en termes de temps de calcul de 36,46% pour TLS, 35,59% pour MLMH, 28,73% pour FB-MLMH et 27,23% pour FB-MLMH-ND. Ainsi cette règle de dominance s'avère plus performante pour les algorithmes mono-directionnels et la plus performante pour TLS. La même comparaison entre l'absence de dominance et la dominance basée sur les états donne des améliorations en temps de calcul de 39,03% pour TLS, 42,32% pour

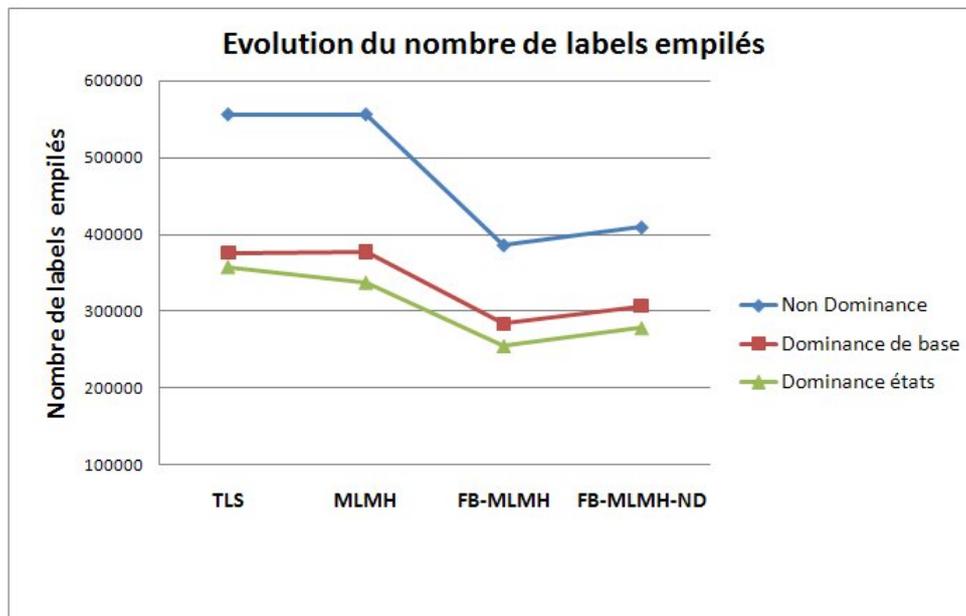


Figure III.2 – Comparaison des méthodes et règles de dominances en termes de nombre de nœuds empilés

MLMH, 37,82% pour FB-MLMH et 31,41% pour FB-MLMH-ND. De nouveau, l’apport de la règle de dominance est plus important pour les algorithmes mono-directionnels mais le gain de performance est le plus important pour l’algorithme MLMH. Si on considère l’apport de la règle de dominance basée sur les états par rapport à la règle de dominance simple, on constate qu’en termes de temps de calcul elle améliore TLS de 4,05%, MLMH de 10,45%, FB-MLMH de 12,75% et FB-MLMH-ND de 5,74%. Cette comparaison montre que la règle de dominance basée sur les états contribue à améliorer de manière plus efficace les algorithmes MLMH et FB-MLMH ; l’apport le plus significatif concerne cette fois l’algorithme bidirectionnel.

Globalement, sur l’ensemble des expérimentations menées (sans dominance, avec dominance simple et avec dominance basée sur les états) on peut dire que l’algorithme bidirectionnel (FB-MLMH) donne les meilleures performances pour trouver l’ensemble des solutions non dominées, il est suivi par FB-MLMH-ND puis par MLMH et TLS qui s’avèrent assez proches en moyenne. Cependant si on considère les résultats obtenus avec la dominance basée sur les états qui est la plus efficace : l’algorithme FB-MLMH est le plus performant suivi de MLMH puis de FB-MLMH-ND et enfin de TLS. En utilisant la règle de dominance simple uniquement le classement de ces algorithmes est modifié : le plus efficaces reste FB-MLMH suivi par FB-MLMH-ND puis TLS (à égalité en termes de temps de calcul mais en termes de nombres de labels l’algorithme

III.4 Résultats des expérimentations dans le cas non dépendant du temps

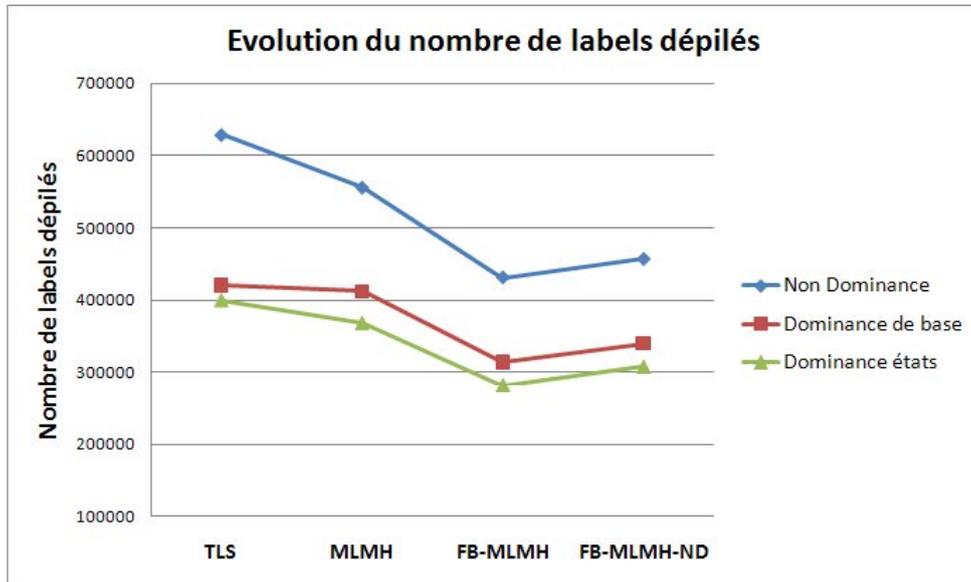


Figure III.3 – Comparaison des méthodes et des règles de dominances en termes de nombre de nœuds dépilés

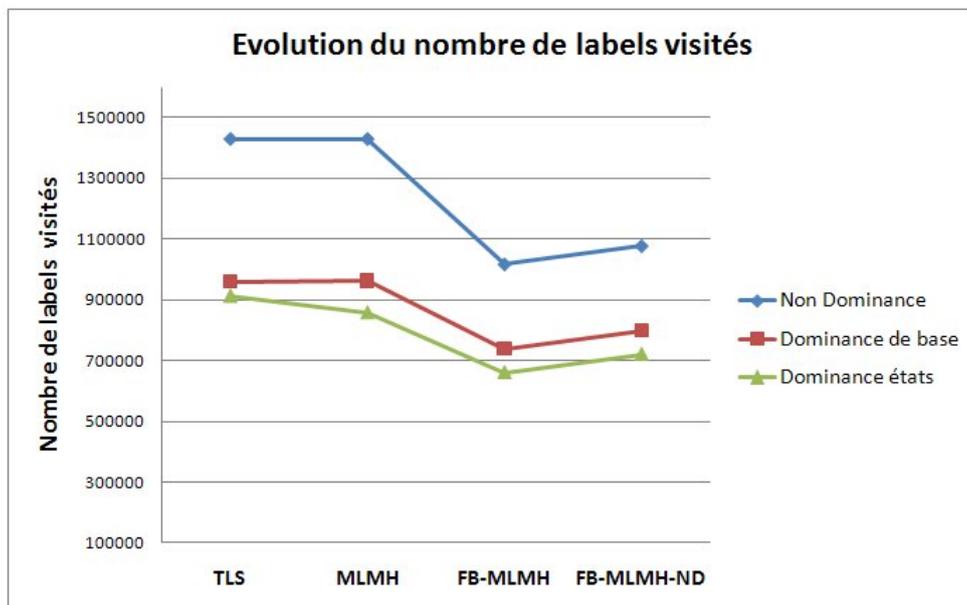


Figure III.4 – Comparaison des méthodes et des règles de dominances en termes de nombre de nœuds visités

bidirectionnel est plus performant que TLS) et en dernière position par l'algorithme MLMH.

4.2 Résultats des expérimentations avec A^*

Les résultats des tests effectués pour évaluer l'effet de la recherche A^* sur les performances des algorithmes sont décrits par les tableaux III.6, III.7 et III.8.

	A^* -DIJ-MM	A^* -TLS	A^* -MLMH	A^* -FB-MLMH-D	A^* -FB-MLMH-ND
temps Cpu (ms)	4 353	2 496	2 560	1 944	2 026
# labels dépilés	1 411 690	541 619	541 618	338 490	366 260
# labels empilés	1 550 930	614 485	595 728	379 994	415 358
# labels visités	3 636 570	1 394 650	1 394 650	894 320	966 229

Tableau III.6 – Résultats des algorithmes A^* (sans règle de dominance)

	A^* -DIJ-MM	A^* -TLS	A^* -MLMH	A^* -FB-MLMH-D	A^* -FB-MLMH-ND
temps Cpu (ms)	4 131	1 659	1 693	1 455	1 492
# labels dépilés	1 281 550	369 146	370 041	259 671	276 089
# labels empilés	1 407 030	413 389	405 046	288 897	310 959
# labels visités	3 298 560	943 528	946 037	678 385	719 805

Tableau III.7 – Résultats des algorithmes A^* (avec la règle de dominance de base)

	A^* -DIJ-MM	A^* -TLS	A^* -MLMH	A^* -FB-MLMH-D	A^* -FB-MLMH-ND
temps Cpu (ms)	4 030	1 559	1 494	1 337	1 326
# labels dépilés	1 281 550	351 998	332 027	235 079	251 082
# labels empilés	1 407 030	393 947	362 668	260 762	282 515
# labels visités	3 298 560	899 175	844 700	610 145	650 729

Tableau III.8 – Résultats des algorithmes A^* (avec la règle de dominance entre états)

4.2.1 Analyse des résultats en termes de temps de calcul et de nombre de labels explorés

De même que le cas sans A^* , les résultats obtenus montrent que l'algorithme DIJ-MM n'est pas performant par rapport aux autres et que l'algorithme A^* FB-MLMH donnent globalement

III.4 Résultats des expérimentations dans le cas non dépendant du temps

les meilleurs résultats en terme de performances. Il améliore les temps de calculs de l'algorithme TLS respectivement de 22,12% sans règle de dominance, de 12,3% avec la règle de dominance de base et de 14,24% avec la règle de dominance basée sur les états. Ces résultats se retrouvent en termes de labels explorés (dépilés, empilés ou visités).

Cette amélioration peut être aussi observée par rapport à l'algorithme MLMH (amélioration respective de 24,06%, 14,06% et 10,51%). En revanche la comparaison est plus serrée entre l'algorithme A^* FB-MLMH et A^* FB-MLMH-ND. On observe en effet une amélioration de l'algorithme basé sur un automate déterministe en temps de calcul de 4,05% sans dominance et de 2,48% avec la dominance simple mais légère dégradation en temps de calcul de 0,83% avec la dominance complète. En termes de nombre de labels explorés A^* FB-MLMH reste toujours meilleur que A^* FB-MLMH-ND.

Comme dans le cas sans A^* , l'algorithme TLS est meilleur en temps de calcul que l'algorithme MLMH sans règle de dominance ou avec dominance simple (dégradation respective de MLMH de 2,56% et de 2,05%) mais avec utilisation de la règle de dominance basée sur les états MLMH améliore TLS de 4,17%. Ce résultat est identique pour le nombre de labels explorés sauf pour le nombre de labels empilés pour lequel MLMH est toujours meilleur que TLS.

4.2.2 Analyse des résultats en termes de dominance

Les figures III.5, III.6, III.7 et III.8 comparent les différents algorithmes en fonction des règles de dominance utilisées. Par rapport à l'absence de règle de dominance, la dominance simple apporte un gain en termes de temps de calcul et de nombre de labels explorés pour tous les algorithmes (en temps de calcul ce gain est de de 33,53% pour TLS, 33,87% pour MLMH, 25,15% pour FB-MLMH et 26,36% pour FB-MLMH-ND. Ainsi cette règle de dominance s'avère plus performante pour les algorithmes mono-directionnels et la plus performante pour MLMH. Cependant les gains observés sont un peu moins élevés que dans les variantes sans A^* .

La même comparaison entre l'absence de dominance et la dominance basée sur les états donne des améliorations en temps de calcul de 37,54% pour TLS, 41,64% pour MLMH, 31,22% pour FB-MLMH et 34,55% pour FB-MLMH-ND. De nouveau, l'apport de la règle de dominance est plus important pour les algorithmes mono-directionnels. De plus, comme précédemment, le gain de performance est le plus important pour l'algorithme MLMH.

Par rapport à la même comparaison effectuée sur les variantes sans A^* , les gains de performances sont moins importants sauf pour l'algorithme FB-MLMH-ND.

Si on considère l'apport de la règle de dominance basée sur les états par rapport à la règle de dominance simple, on constate qu'en termes de temps de calcul elle améliore TLS de 6,03%,

Chapitre III. Résultats expérimentaux

MLMH de 11,75%, FB-MLMH de 8,11% et FB-MLMH-ND de 11,13%. Cette comparaison montre que la règle de dominance basée sur les états contribue à améliorer de manière plus efficace les algorithmes MLMH et FB-MLMH-NB ; de plus pour tous les algorithmes sauf FB-MLMH, le gain est plus important pour les variantes utilisant A^* que pour celles ne l'utilisant pas.

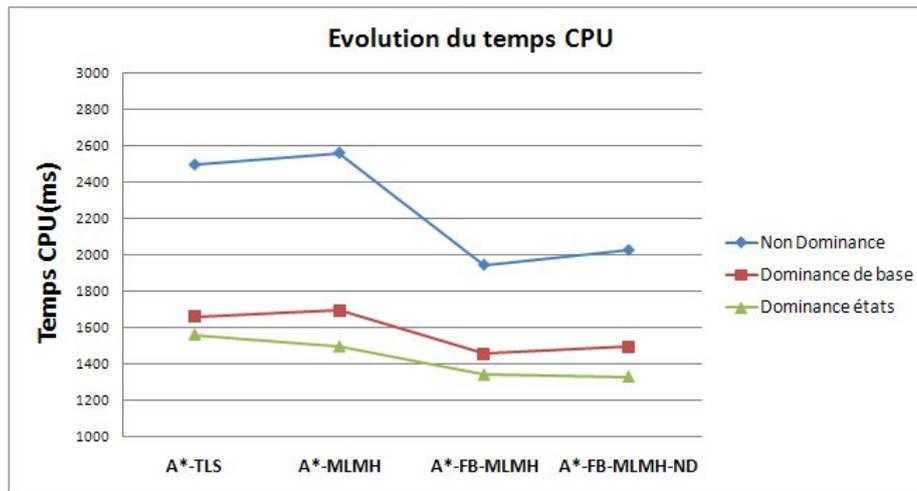


Figure III.5 – Comparaison des méthodes et des règles de dominances en termes de temps CPU

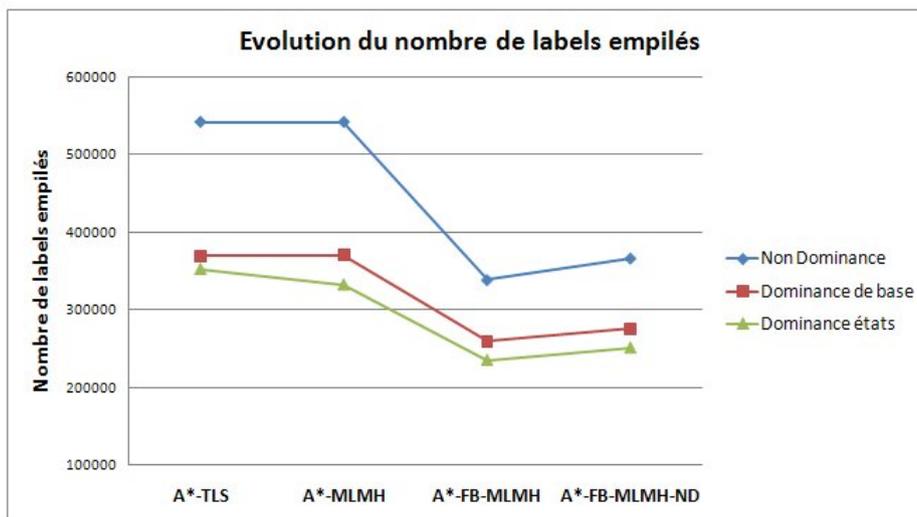


Figure III.6 – Comparaison des méthodes et des règles de dominances en termes de nombre de nœuds empilés

III.4 Résultats des expérimentations dans le cas non dépendant du temps

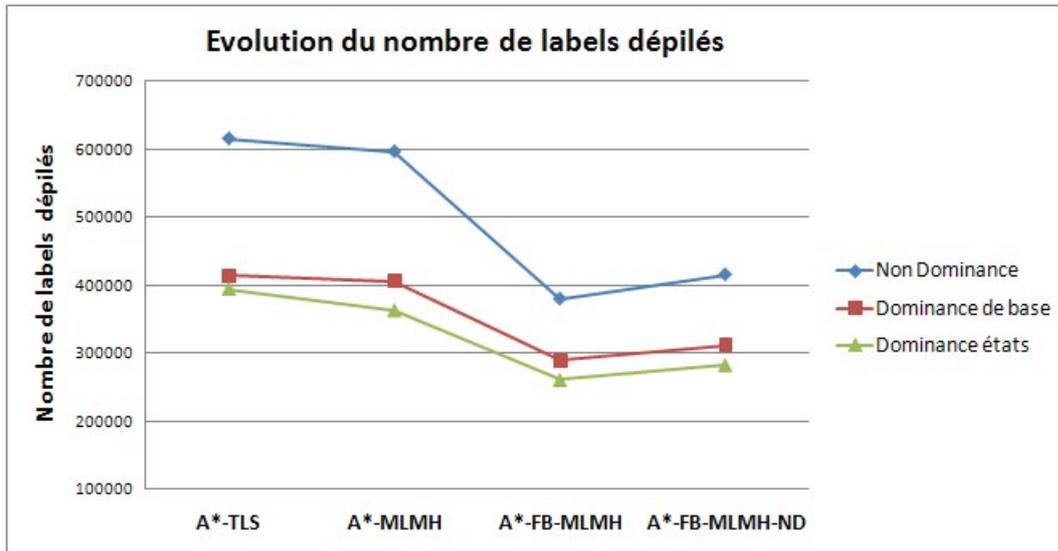


Figure III.7 – Comparaison des méthodes et des règles de dominances en termes de nombre de nœuds dépilés

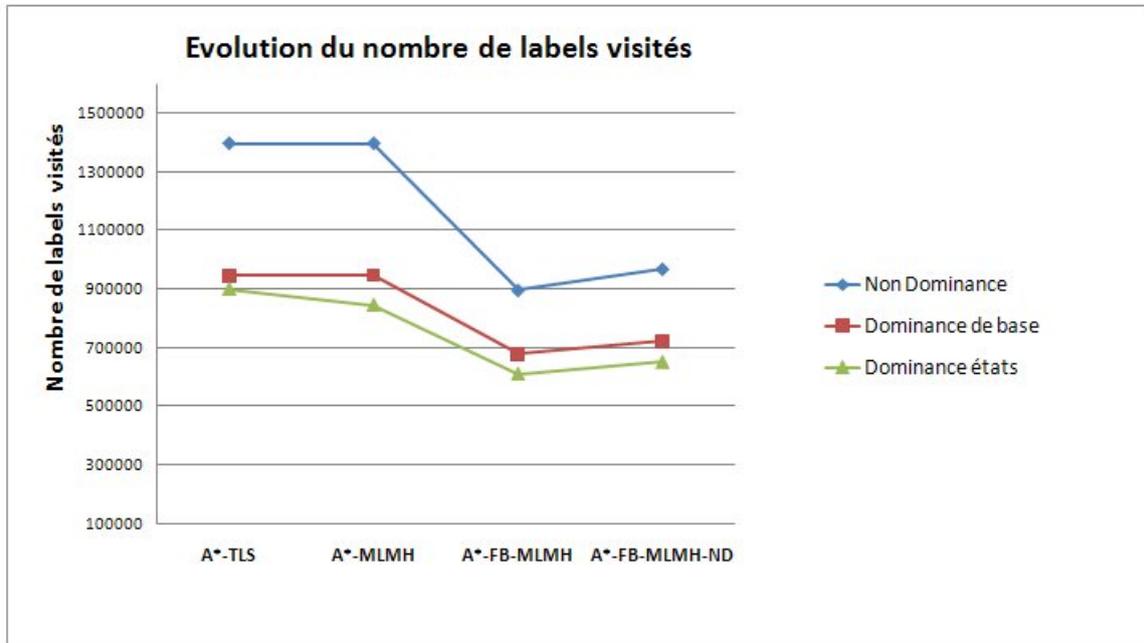


Figure III.8 – Comparaison des méthodes et des règles de dominances en termes de nombre de nœuds visités

Comparaison des variantes avec et sans A^*

Sur les figures (III.9 à III.11) on compare les performances relatives des algorithmes avec et sans A^* . Dans ce cas encore nous n'avons pas représenté l'algorithme DIJ-MM sur ces courbes car il présente des performances très éloignées des autres algorithmes.

Sans règles de dominance, les algorithmes avec A^* obtiennent des gains de performance allant de 3,8% à 13,41% (pour FB-MLMH) en termes de temps de calcul et ses résultats sont de même nature pour le nombre de labels explorés.

En utilisant la règle de dominance simple, l'algorithme A^* -TLS s'avère très légèrement moins rapide (diminution de 0,30% en temps de calcul) et les gains vont jusqu'à 9,79% pour l'algorithme A^* -FB-MLMH-ND. Avec la règle de dominance basée sur les états, les gains vont de 1,76% à 14,95% pour l'algorithme A^* -FB-MLMH-ND.

Avec utilisation de règles de dominance, l'utilisation du principe A^* s'avère plus intéressante pour l'algorithme FB-MLMH-ND puis pour FB-MLMH puis pour MLMH et enfin pour TLS.

En résumé, ces résultats sont proches des résultats établis dans le cas sans A^* : la règle de

III.4 Résultats des expérimentations dans le cas non dépendant du temps

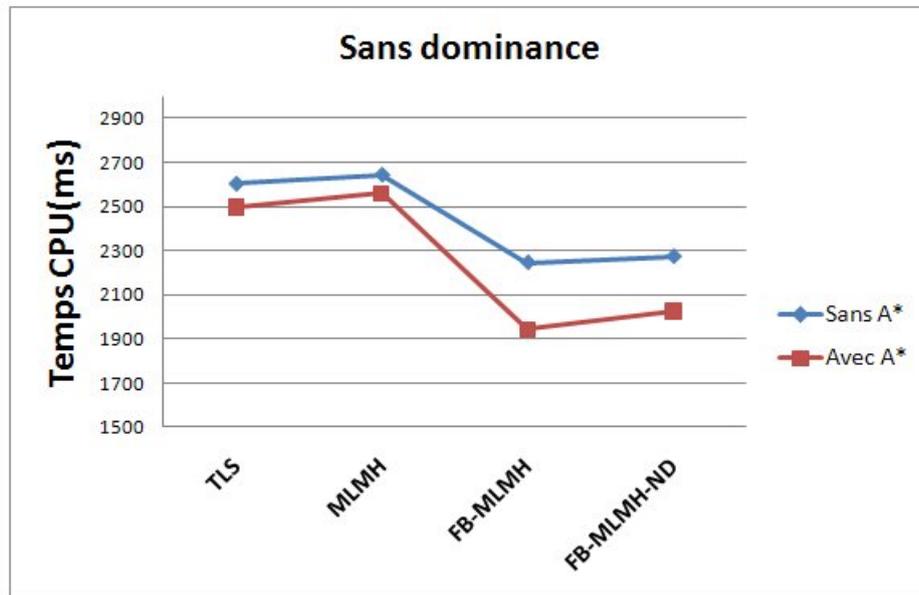


Figure III.9 – Comparaison des variantes avec ou sans A^* en termes de temps CPU (sans dominance)

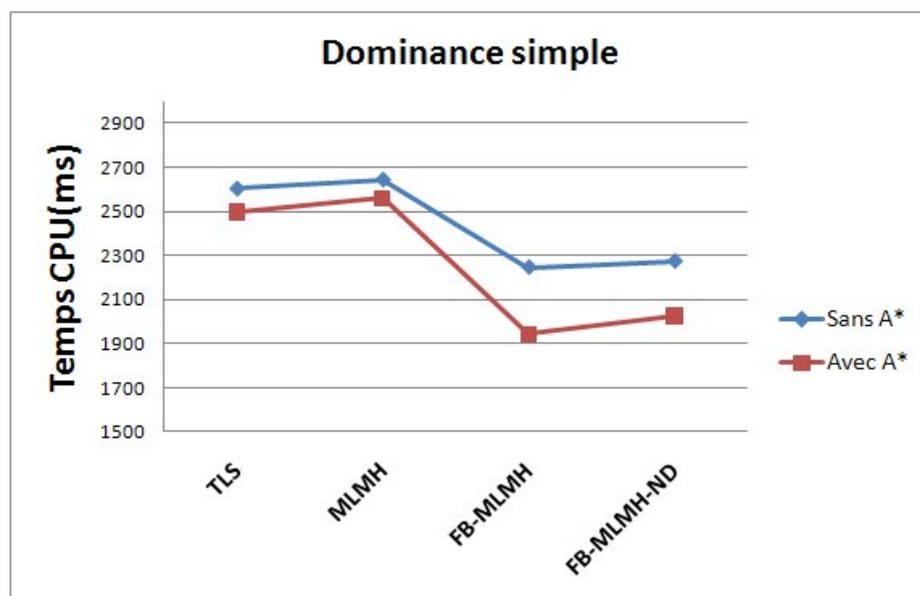


Figure III.10 – Comparaison des variantes avec ou sans A^* en termes de temps CPU (Dominance de base)

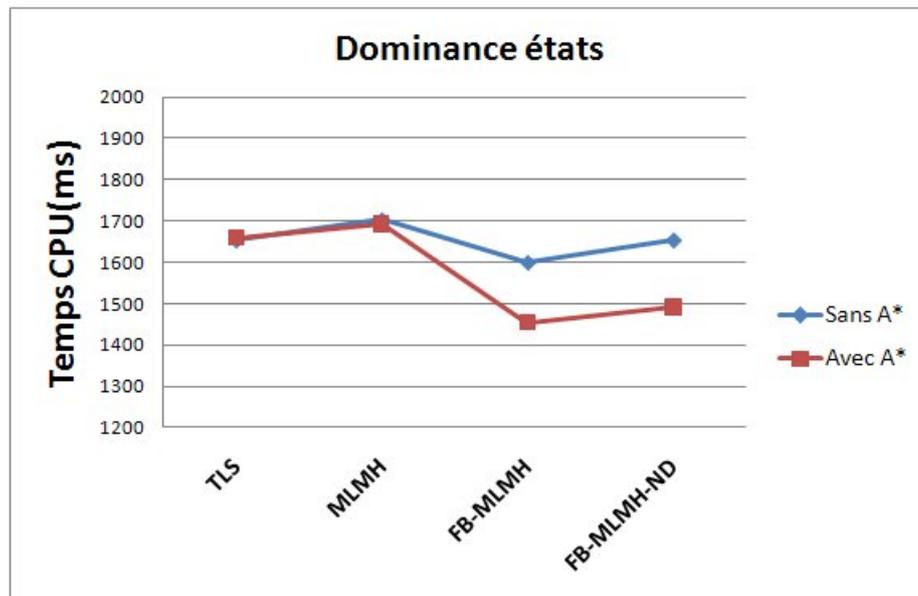


Figure III.11 – Comparaison des variantes avec ou sans A^* en termes de temps CPU (Dominance états)

dominance basée sur les états fournit les meilleures performances par rapport aux autres règles de dominance, l'algorithme MLMH est plus efficace que l'algorithme TLS avec la règle de dominance basée sur les états. Cependant l'écart entre l'algorithme A^* -FB-MLMH et l'algorithme A^* -FB-MLMH-ND s'est resserré : il s'avère légèrement plus rapide avec la règle de dominance basée sur les états même s'il parcourt plus de labels. De plus, en utilisant le principe A^* , les deux algorithmes bidirectionnels sont plus efficaces que l'algorithme mono-directionnel MLMH ce qui n'est pas le cas sans l'utilisation de A^* .

4.3 Conclusion

Nous constatons que les méthodes bidirectionnelles sont efficaces (FB-MLMH, A^* -FB-MLMH et A^* -FB-MLMH-ND) pour résoudre notre problème. Elles le sont d'autant plus dans le cas de l'utilisation de la règle de dominances basées sur les états et du principe A^* . Par ailleurs, cette règle de dominance basée sur les états améliore les algorithmes testés (sauf l'extension de Dijkstra). Le principe de A^* améliore les performances de l'ensemble des algorithmes mais cette amélioration est beaucoup plus importante pour les méthodes bidirectionnelles.

Les deux méthodes monodirectionnelles TLS et MLMH sont très proches sur l'ensemble des

III.5 Résultats des expérimentations dépendant du temps

résultats : avec utilisation de la dominance basée sur les états, l'algorithme proposé MLMH s'avère plus performant.

En conclusion, sur le réseau non dépendant du temps testé, l'algorithme FB-MLMH que nous proposons parvient à obtenir toutes les solutions non dominées en environ 1,3s en moyenne et améliore ainsi les performances de l'algorithme TLS, directement adapté de l'algorithme de Lozano et Storchi [62]. L'impact positif des règles de dominance est également vérifié.

Dans la partie suivante, nous nous intéressons aux résultats des expérimentations dans le cas où le réseau de transport multimodal est dépendant du temps.

5 Résultats des expérimentations dépendant du temps

Tous les algorithmes testés parviennent également à résoudre optimalement toutes les instances. Sur les 100 instances testés sur ce réseau, le temps de trajets moyen minimum, moyen et maximum sont respectivement de 208, 455 et 1010 minutes. Les nombres de solutions non dominées sont en moyenne au nombre de 5 comme pour le cas statique. Elles contiennent en moyenne 4 transferts.

5.1 Résultats sans A^*

Les résultats des expérimentations sont obtenus en comparant les différentes méthodes entre elles uniquement dans le cas de l'utilisation de la règle de dominance basée sur les états.

Le tableau (III.9) donne les résultats obtenus en moyenne sur les 100 trajets O-D pour chaque métrique et pour chaque type d'algorithme sans l'utilisation de la recherche A^* . Les horaires de départ de chaque trajet correspondent à un départ à 8 heure en semaine, afin de pouvoir utiliser au mieux les réseaux du transport en commun.

	DIJ-MM	TLS	MLMH	FB-MLMH	FB-MLMH-ND
temps Cpu (ms)	5 284	1 765	1 696	1 523	1 581
# labels dépilés	1 458 080	344 125	345 976	248 035	271 532
# labels empilés	1 584 860	386 538	378 042	275 912	301 892
# labels visités	3 902 600	911 676	916 213	672 985	737 073

Tableau III.9 – Résultats des algorithmes (dominance basée sur les états)

Les courbes des figures III.12, III.13 , III.14 et III.15 présentent les comparaisons des algorithmes pour chaque type de métrique.

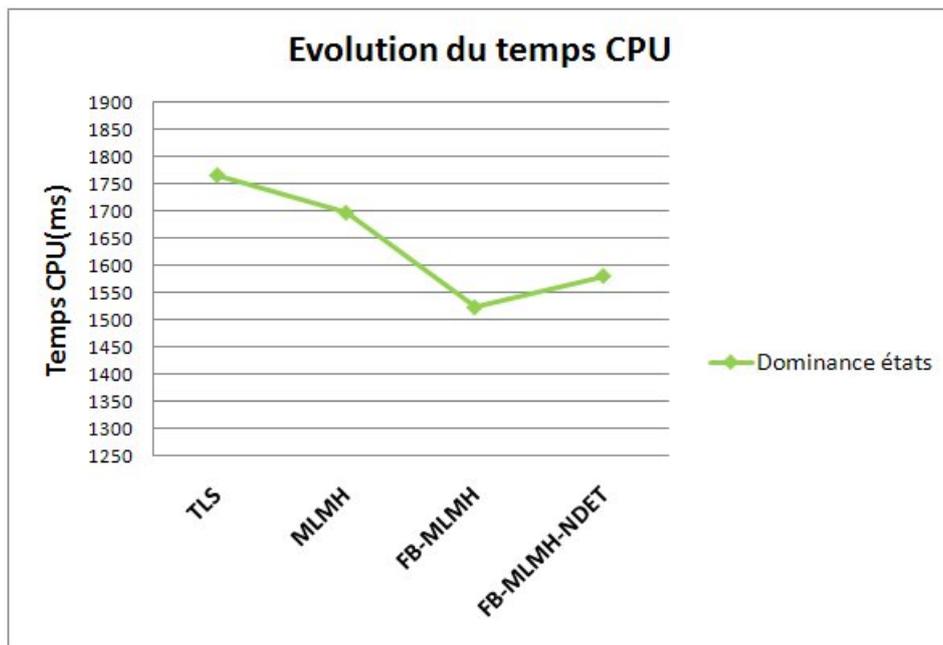


Figure III.12 – Comparaison des méthodes en termes de temps CPU

5.1.1 Analyse des résultats

D'après les figures III.12, III.13, III.14 et III.15, nous remarquons dans le cas dépendant du temps, l'algorithme **FB-MLMH** donne encore les meilleures performances sur l'ensemble des métriques.

- Pour le temps de calcul, nous constatons que l'algorithme FB-MLMH est meilleur que les autres algorithmes. L'écart est respectivement de 13,71%, 10,2%, et 3,67% par rapport aux algorithmes TLS, MLMH et FB-MLMHND.
- Pour les nombres de labels explorés (empilés, dépilés ou visités), les variations sont plus significatives mais vont dans le même sens (entre 26 et 29% d'amélioration sur le nombre de labels empilés par rapport à l'algorithme TLS ou MLMH, entre 8,5 et 8,7% d'amélioration sur le nombre de labels visités par rapport à FB-MLMH-ND).

On remarque également que les algorithmes TLS et MLMH ont des performances similaires : MLMH est légèrement moins bon que TLS sur le nombre de labels dépilés et

III.5 Résultats des expérimentations dépendant du temps

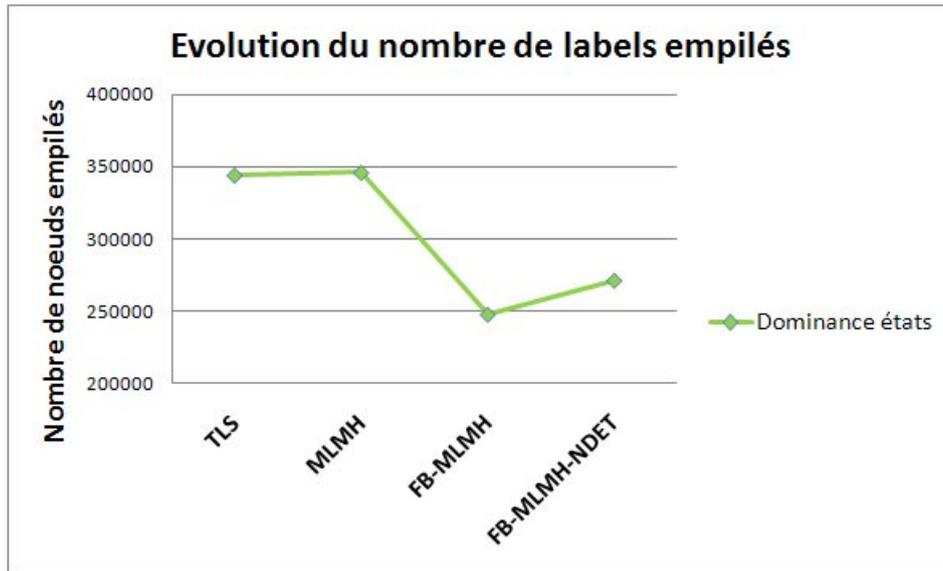


Figure III.13 – Comparaison des méthodes en termes de nombre de nœuds empilés

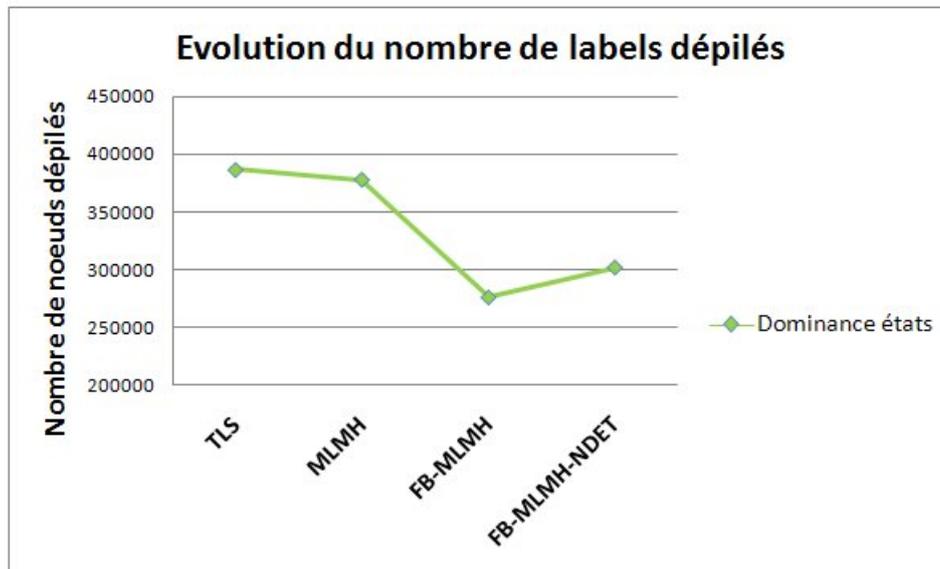


Figure III.14 – Comparaison des méthodes en termes de nombre de nœuds dépilés

visités (d'environ 0,5%) et légèrement meilleur que TLS sur le nombre de labels empilés (d'environ 2,2%) et sur le temps de calcul (d'environ 3,9%).

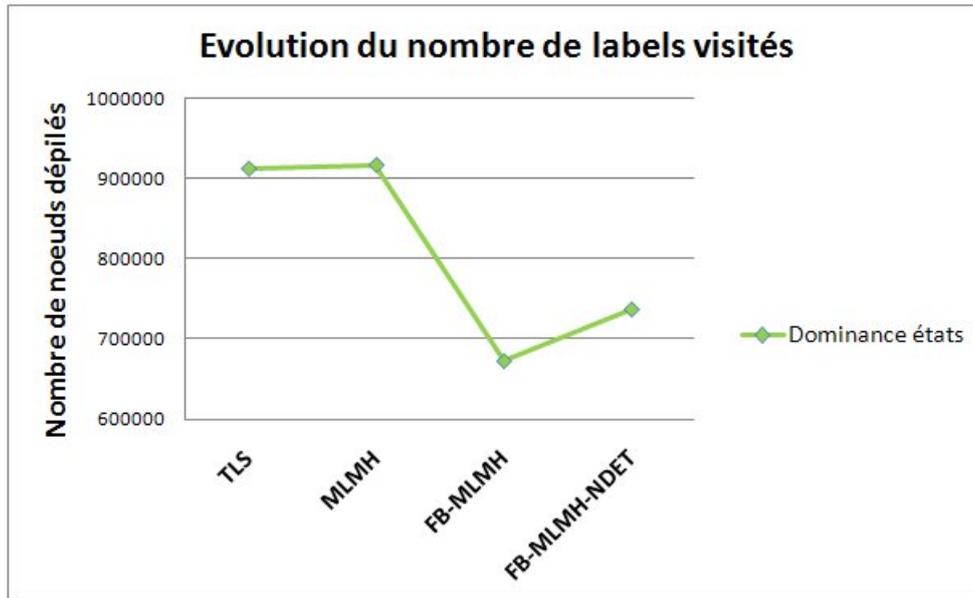


Figure III.15 – Comparaison des méthodes en termes de nombre de nœuds visités

En résumé, sans utiliser le principe de A^* , l’algorithme bidirectionnel FB-MLMH donne, comme dans le cas non dépendant du temps, les meilleures performances. Le deuxième algorithme le plus performant est FB-MLMH-ND qui est également bidirectionnel.

5.2 Résultats des expérimentations avec A^*

Dans ces expérimentations, nous cherchons à évaluer le principe de A^* pour les différents algorithmes. Les résultats sont présentés dans le tableau III.10. Les figures III.16, III.17,

	A^* -DIJ-MM	A^* -TLS	A^* -MLMH	A^* -FB-MLMH-D	A^* -FB-MLMH-ND
temps Cpu (ms)	5 094	1 714	1 671	1 366	1 437
# labels dépilés	1 414 320	339 288	341 616	233 812	237 393
# labels empilés	1 541 620	382254	374 116	261 236	269 807
# labels visités	3 781 350	900 099	905 769	635 906	643 260

Tableau III.10 – Résultats des algorithmes A^* (dominance basée sur les états)

III.18 et III.19 montrent que, dans le cas dépendant du temps, l’algorithme FB-MLMH donne les meilleures performances en termes de temps de calcul et de nombre de labels. Ces temps de calcul varient en fonction des différents algorithmes. L’algorithme FB-MLMH améliore les

III.5 Résultats des expérimentations dépendant du temps

temps de calcul respectivement de 20,3%, 18,25%, et 4,94% par rapport aux algorithmes TLS, MLMH et FBMLMH-ND. Ces résultats sont de même nature sur les nombres de labels.

Dans le cas dépendant du temps avec le principe A^* , comme précédemment, les algorithmes bidirectionnels ont des résultats assez similaires : l'écart entre A^* -FB-MLMH et A^* -FB-MLMH-ND est en faveur de l'algorithme basé sur un automate déterministe mais reste assez faible (entre 1 et 3% sur le nombre de labels mais presque 5% sur le temps de calcul).

Avec le principe A^* , les algorithmes TLS et MLMH restent toujours proches en termes de labels explorés (MLMH est moins bon d'un peu plus de 0,6% pour le nombre de labels dépilés et visités mais meilleur d'un peu plus de 2% sur le nombre de labels empilés). En revanche MLMH est meilleur que TLS en temps de calcul (de 2,5%). La comparaison de ces deux algorithmes reste stable avec l'utilisation ou non du principe A^* .

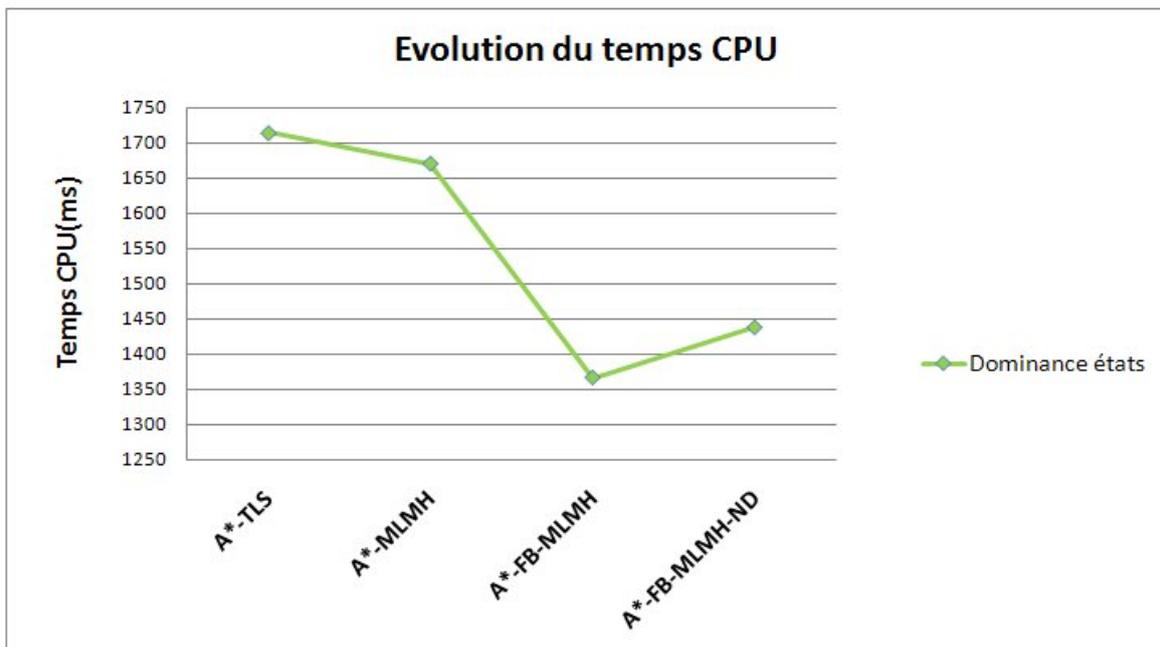


Figure III.16 – Comparaison des méthodes en termes de temps CPU

La comparaison des variantes des algorithmes avec ou sans A^* (figure III.20), montre un apport de A^* de 1,47% (pour MLMH) à 10,31% (pour FB-MLMH) sur le temps de calcul. Pour le nombre de labels (dépilés, empilés et visités) cet apport est compris entre environ 1% pour MLMH et va jusqu'à plus de 12% pour FB-MLMH-ND. Comme dans le cas statique,

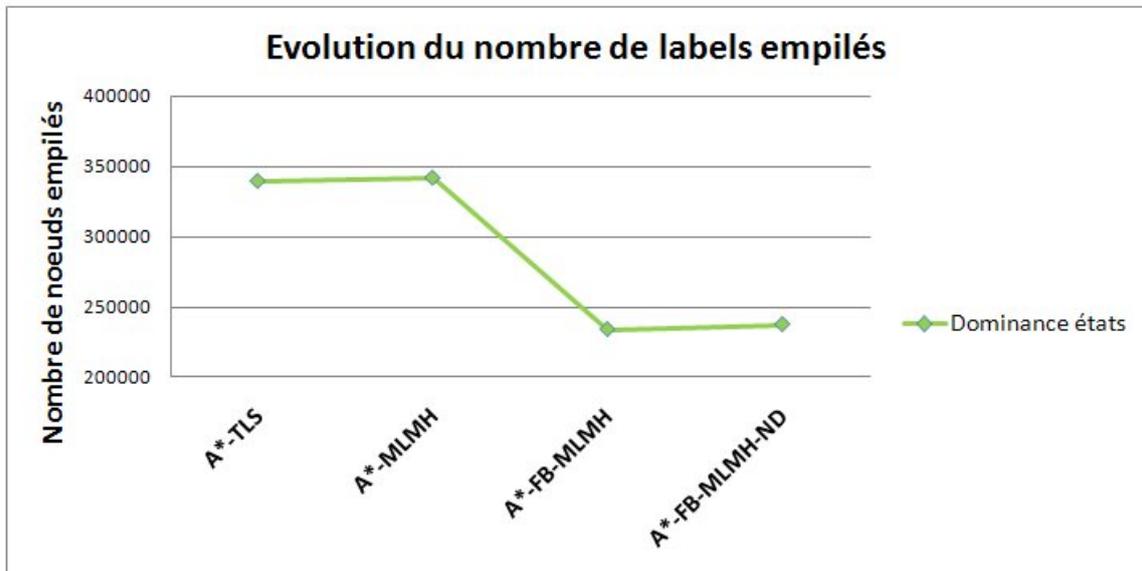


Figure III.17 – Comparaison des méthodes en termes de nombre de nœuds empilés

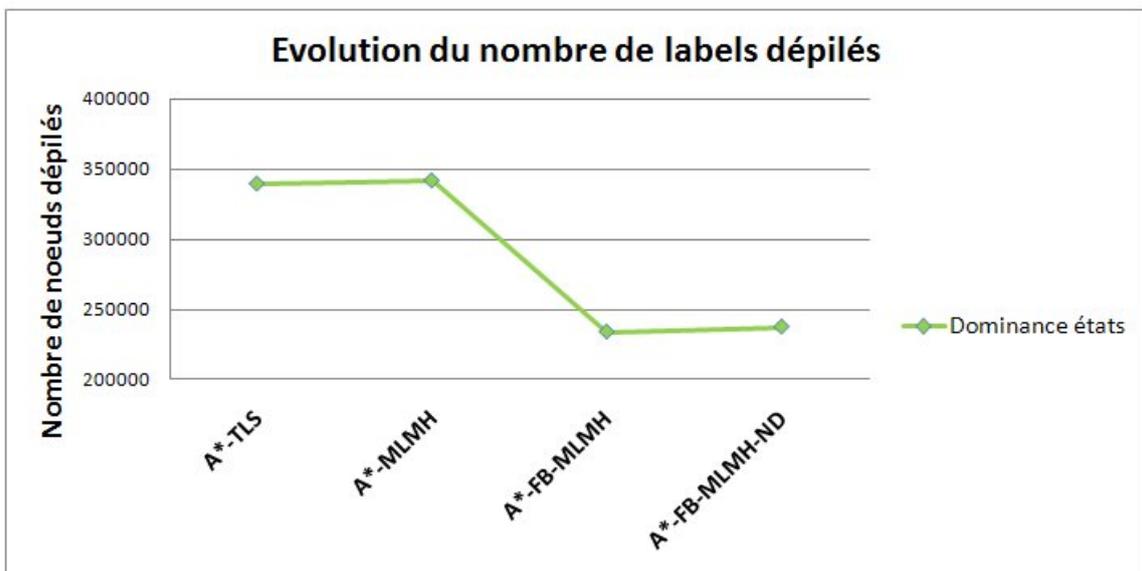


Figure III.18 – Comparaison des méthodes en termes de nombre de nœuds dépilés

l'algorithme bidirectionnel basé sur un automate non déterministe tire globalement le plus profit du principe de A^* .

En conclusion, les résultats des tests sur un réseau dépendant du temps, montrent que, comme

III.5 Résultats des expérimentations dépendant du temps

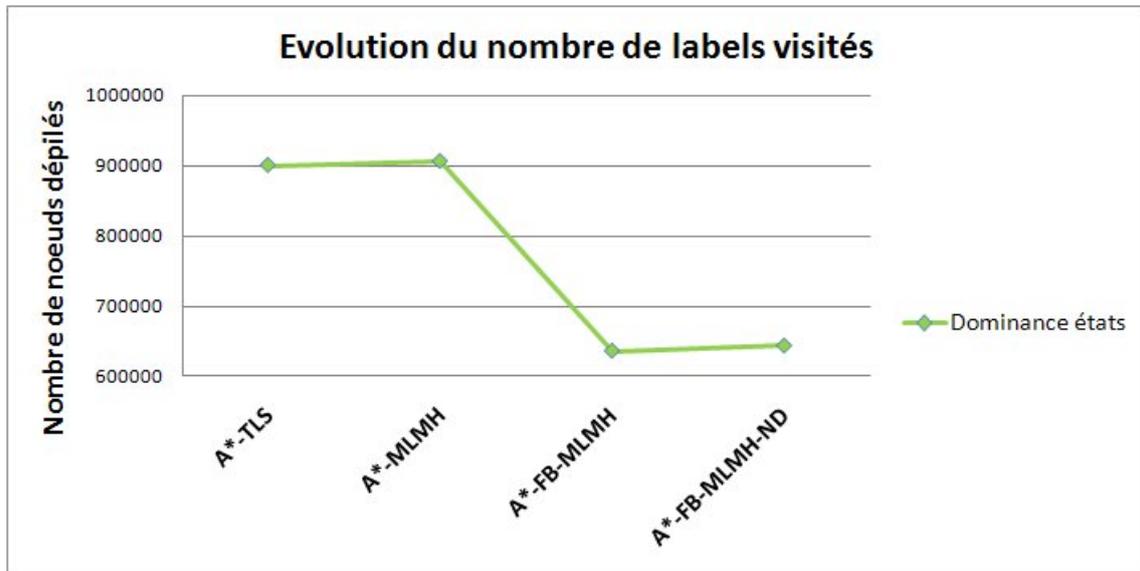


Figure III.19 – Comparaison des méthodes en termes de nombre de nœuds visités

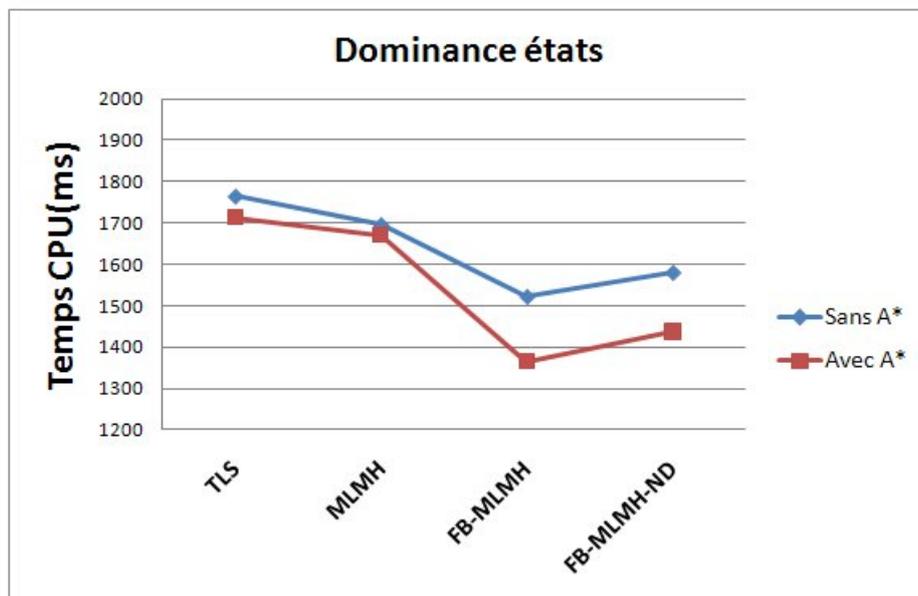


Figure III.20 – Comparaison des variantes avec ou sans A^* dépendant du temps en termes de temps CPU (Dominance états)

dans le cas de réseau statique, , l'algorithme FB-MLMH réalise les meilleures performances sur l'ensemble des métriques suivi par ce même algorithme basé sur un automate de viabilité non déterministe. En revanche, l'ajout du principe A^* , ne donne pas la supériorité à l'algorithme FB-MLMH-ND alors que c'était le cas dans le cas statique.

5.3 Conclusion

Les résultats de tests effectués dans le cas de réseau dépendant du temps montrent que :

- les algorithmes bidirectionnels sont plus performants que les algorithmes mono-directionnels. L'algorithme FB-MLMH basé sur un automate de viabilité déterministe réalise les meilleures performances sur l'ensemble des tests ;
- L'accélération A^* contribue à la diminution des temps de calcul et du nombre de labels explorés. L'apport de A^* est plus important plus les algorithmes bidirectionnels que pour ceux monodirectionnels

Ainsi, dans le cas dépendant du temps, l'algorithme A^* -FB-MLMH obtient les meilleures performances et obtient l'ensemble des solutions non dominées en moins de 1,4s, ce qui est très comparable au cas non dépendant du temps, malgré l'augmentation de la taille du graphe, et peut donc être considéré comme très satisfaisant. L'amélioration du temps de calcul par rapport à l'algorithme TLS (qui obtient environ 1,8s en moyenne), lui-même étant une amélioration de l'algorithme de Lozano et Storchi [62] est plus importante que dans le cas dépendant du temps.

6 Conclusion

Dans ce chapitre, nous avons présenté et analysé les résultats expérimentaux obtenus par différents algorithmes proposés pour le problème de recherches d'itinéraires multimodaux. Les tests ont été effectués sur deux types de réseaux sont issus des données de transport de la ville de Toulouse. Le premier réseau représente un cas statique (non dépendant du temps) et le deuxième représente un réseau dépendant du temps.

Nous avons montré que les algorithmes bidirectionnels proposés réalisent de bonnes performances. Ces performances sont meilleures pour les algorithmes bidirectionnels avec utilisation de la règle de dominance entre états mais aussi dans le cas de la recherche avec le principe A^* . En règle générale, c'est l'algorithme bidirectionnel basé sur un automate de viabilité déterministe qui est le plus performant. Cependant dans le cas statique avec A^* , c'est la variante bidirectionnelle utilisant un automate non déterministe qui s'avère plus efficace.

Nous avons également montré dans ces expérimentations que la règle de dominance basée sur les états est plus forte que celle basée uniquement sur le temps de trajet et le nombre de changement de modes. Elle contribue à améliorer plus fortement les performances des algorithmes MLMH et FB-MLMH que celles de l'algorithme TLS.

Globalement, que ce soit pour le cas statique ou pour le cas dépendant du temps, nous avons amélioré les performances de l'algorithme initial proposé par Lozano et Storchi [62].

Pour chaque instance du problème, l'ensemble des solutions exactes non dominées a été trouvé par tous les algorithmes. Les temps de calcul restent raisonnables (un peu plus d'une seconde pour les plus rapides) mais le passage à des réseaux de très grande taille nécessite de s'intéresser à d'autres techniques d'accélération pour la recherche d'itinéraires telles que celles déjà existantes dans le cas de réseaux de transport mono-modal.

Ces algorithmes sont en cours d'intégration dans le logiciel industriel dont le prototype actuel utilise l'algorithme **MLMH** à travers lequel les résultats des algorithmes pourront être exploités et les itinéraires multimodaux affichés.

Nous présentons à la suite de ce chapitre l'industrialisation de ces travaux de recherches avec le développement du logiciel.

Chapitre IV

Travaux d'Industrialisation et logiciel développé

1 Introduction

Les travaux de recherche ont été menés en collaboration avec la société MobiGIS spécialisée dans le domaine des Systèmes d'Information Géographique (SIG) en particulier dans le secteur de la mobilité et du transport des personnes.

L'un des objectifs de la société MobiGIS est de mettre en œuvre des modèles de données qui prennent en compte potentiellement tous les modes de déplacements significatifs et de mettre au point des solutions logicielles d'analyse multimodales qui exploitent le modèle de données. C'est ainsi qu'un logiciel exploitant les modèles et algorithmes de recherche d'itinéraires présentés dans les chapitres précédents a été développé. Il permet d'ores et déjà d'être utilisé dans le but d'organiser et de planifier des réseaux de transport à l'échelle d'une aire urbaine, péri-urbaine ou inter-urbaine.

Dans ce chapitre dédié à l'industrialisation de nos travaux nous présentons ce logiciel industriel et des cas d'utilisation pour l'analyse de la mobilité. En premier lieu, les objectifs visés par le logiciel sont exposés, on exposera ensuite ses fonctionnalités. Les deux parties suivantes sont consacrées à la conception et au paramétrage du logiciel. Nous présentons ensuite la modélisation dans une base de données géo-spatiale de réseaux de transport multimodaux. Enfin, la dernière partie de ce chapitre présente des résultats obtenus par le logiciel.

Les expérimentations ont été menées sur la zone du Grand Toulouse avec des données de transports publics réels mises à disposition par Tisséo pour ce projet de recherche et pour le projet de recherche Potimart de mise en place d'un SIG transport dans le monde Open Source (www.potimart.org). Les cartes présentées dans ce chapitre sont toutes situées sur la zone du

Grand Toulouse.

2 Objectifs du logiciel

Le logiciel fonctionne dans l'environnement SIG ArcGIS© pourvu de l'extension NetworkAnalyst qui permet de modéliser des graphes et offre des fonctions évoluées de parcours de graphe. Ce travail effectué dans l'univers des SIG permet également de travailler dans un environnement cartographique et de produire des rapports cartographique ou cartes.

L'objectif premier du logiciel est de permettre une analyse de la mobilité sur des réseaux de transport multimodaux. En cela, il permet de répondre aux problématiques suivantes :

- Planification de futurs réseaux de transport ;
- Évaluation des différents scénarios de transport ;
- Amélioration de la couverture du réseau (couvrir les carences de dessertes) ;
- Amélioration du service : vitesse commerciale, régularité etc. ;
- Diminution des impacts environnementaux d'émission CO2.

Pour l'utilisateur, les apports essentiels sont les suivants :

- la visualisation des offres de transport : accès routiers, lignes et arrêts de transport en commun, parkings, points d'intérêt (hôpitaux, écoles, administrations, musées,...) ;
- la possibilité de mener des études précises grâce à des fonctions dédiées telles que :
 - l'accessibilité (ou isochrone) : quels sont, pour un service ou une offre commerciale donnée, les bassins d'usagers couverts et dans quelles conditions ?
 - la mobilité : quelles sont, pour un déplacement ou un ensemble de déplacements donnés (privés ou professionnels) les différentes alternatives possibles, leurs coûts, leurs impacts environnementaux ?
 - l'intermodalité : comment connecter les différentes offres de transport entre elles afin d'assurer la transition d'un mode vers l'autre ?
 - la multimodalité : comment organiser un déplacement en combinant plusieurs modes de transport choisis parmi l'offre disponible (Voiture Particulière (VP), Transport en Commun (TC)(métro, bus), vélo, marche à pied, etc.) ?

Le logiciel est destiné à différents types d'utilisateurs :

- les gestionnaires et exploitants de réseaux de transport pour optimiser l'utilisation de leurs infrastructures ;
- les collectivités territoriales afin de répondre à la demande croissante du transport ;

- les entreprises du secteur privé ou public par la mise en place de Plans de Déplacement Entreprise (PDE) (ou pour les administrations (PDA)) qui sont des mesures visant à optimiser les déplacements liés aux activités professionnelles en favorisant l’usage des modes de transport alternatifs à la voiture individuelle ;
- les bureaux d’études ou agences d’urbanisme pour proposer des solutions mieux adaptées au besoins des utilisateurs (calculateurs d’itinéraires multimodaux) avec l’utilisation des SIGs ;
- les passagers pour sécuriser leur voyage et expliciter les choix modaux.

Le modèle de données est parfaitement ouvert, il respecte des normes mise en œuvre par l’Open GIS Consortium (OGC) et des normes de données transport telles que Trident. Les données peuvent ainsi être exploitées par d’autres outils que ceux d’ESRI comme, par exemple, des outils Open Source. Cet aspect est parfaitement différenciateur d’outils de modélisation propriétaires comme Emme¹, TransCAD², etc.

Par ailleurs, ce logiciel se positionne en aval des outils de modélisation pure et en amont des Systèmes d’Aide à l’Exploitation (SAE) ou des Systèmes d’Information Voyageur (SIV). Les données et les résultats obtenus par ces systèmes peuvent être exploités par le logiciel développé pour l’analyse des réseaux (vitesses moyenne de circulation, indicateur de validation des tickets dans les transports en commun, etc.).

La figure IV.1 illustre sa position centrale.

3 Principales fonctionnalités

3.1 Fonction de modélisation de l’offre de transport dans une base de données géo-spatiale

Le logiciel offre des fonctions de modélisation pour l’ensemble des modes de transports. Le réseau routier est modélisé à partir des données sources issues des réseaux de voirie de type NavTeq/TéléAtlas/OpenStreetMap. Le modèle tient compte des temps de parcours fixes ou variables (en voiture en fonction du type de voirie) et les parcs de stationnement. Pour le TC (bus, métro, train etc.), les données sources peuvent être fournies de multiples manières. Des normes de données existent et permettent une prise en compte facilitée des données comme la norme Trident issue du modèle européen Transmodel, ou le modèle GTFS mis en place par

1. <http://www.inro.ca/fr/products/emme/>

2. <http://www.caliper.com/>

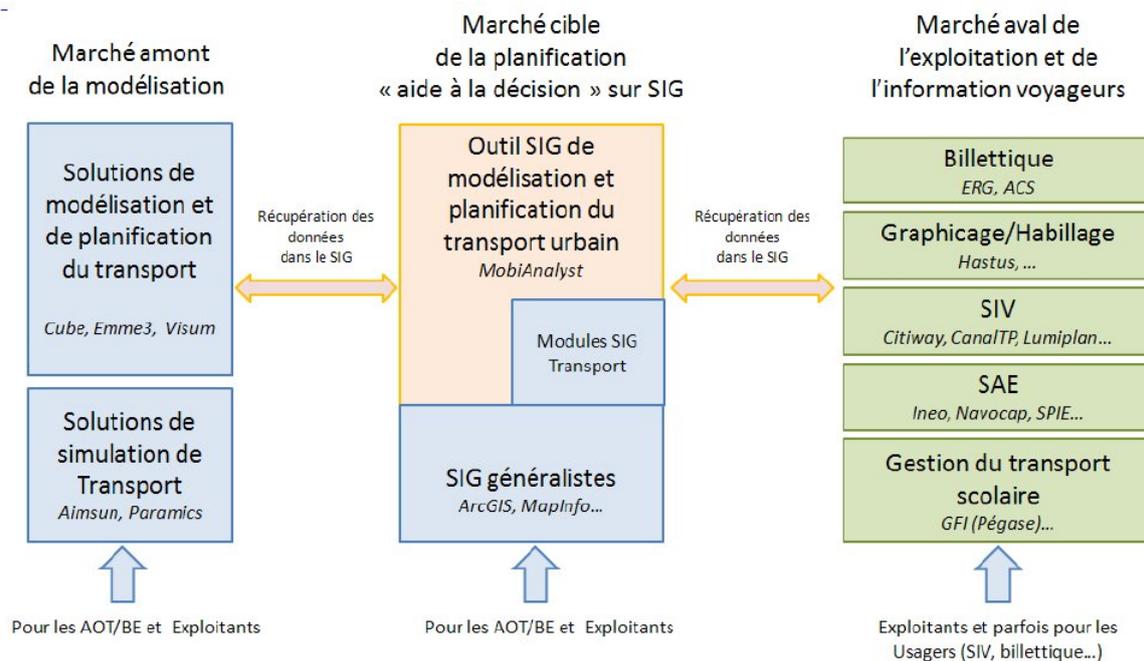


Figure IV.1 – Étude de marché du logiciel

Google.

Potentiellement, tout mode de transport peut être ajouté dans le modèle mis en œuvre : pistes cyclables, voies piétonnes, parkings, parkings relais, auto partage, covoiturage, etc.

Également, des données tierces peuvent compléter les données transport à des fins de statistiques, d'études économiques et sociales : contours administratifs, données INSEE, points d'intérêt (POI).

3.2 Fonctions d'analyses spatiales et multimodales

Les principales fonctions mises en œuvre sont : la recherche d'itinéraires, le calcul d'accessibilité, la recherche de service de proximité, le calcul de matrice Origines/Destinations.

Ces différentes fonctions sont caractérisées par les paramètres suivants :

- mode unique ou en conjugaison de modes (voiture et TC par exemple) ;
- calcul pour un jour type ou une plage horaire type (les vitesses des lignes de TC sont calculées à partir de l'offre théorique ou prédéfinies par l'utilisateur) ;
- calcul pour une date/heure particulière (23/02/2010 11h30 par exemple) avec les horaires théoriques du TC ou en vitesse moyenne ;
- calcul pour une date de départ ou pour une date d'arrivée (départ à 8h d'une origine) ou

(arrivée à 8h à une destination) ;

- restrictions ou non sur les modes de transport : restriction des sens de circulation pour les voitures, piéton sur certaines voiries par exemple ;
- intégration de contraintes utilisateurs (temps de marche maximum pour le piéton par exemple).

Ces fonctionnalités permettent de proposer des services ajoutés. En particulier dans le cas des isochrones, elles permettent d'évaluer certains indicateurs importants pour les agences d'urbanisme ou les agences immobilières :

- calcul d'indicateurs TC (nombre de dessertes par arrêt évalué pour un jour et une plage horaire) : par exemple la desserte en transport multimodal pour accéder aux commerces, aux écoles ou encore aux zones d'emploi pour un habitat privé donné ;
- croisement des recherches d'itinéraires ou accessibilités avec des données socio-économiques (INSEE, IRIS) ou d'autres données (billettique, trafic routier, etc.). Ainsi, il est possible de quantifier une implantation commerciale, en analysant sur la base des recherches d'itinéraires ou isochrone et des données INSEE, les populations (par catégorie) susceptibles d'y accéder via les réseaux de transport multimodaux ;

4 Conception du logiciel

4.1 Architecture de l'application

Une architecture 3-tiers a été mise en œuvre afin de bien séparer : (1) le modèle de données, (2) le noyau applicatif et (3) l'Interface Homme Machine (IHM).(cf. figure IV.2)

4.1.1 Composant d'accès aux données et administration du logiciel

Ce composant est un élément fondamental pour la mise en œuvre du logiciel : des données de qualité, des réseaux de transport interconnectés (ou non), des vitesses de circulation ou des horaires théoriques et des fréquences de passage pour les véhicules de TC, etc. L'ensemble des données est placé dans la base de données relationnelle spatialisée. Les données sont traitées pour permettre leur exploitation :

- les données TC (horaires, fréquences, missions, vitesses moyennes de circulation des bus) sont structurées pour le modèle utilisé dans le logiciel ;
- le réseau TC est connecté à la voirie, des attributs sont calculés et affectés aux éléments du réseau : temps de trajet piéton, temps de trajet voiture, etc. ;

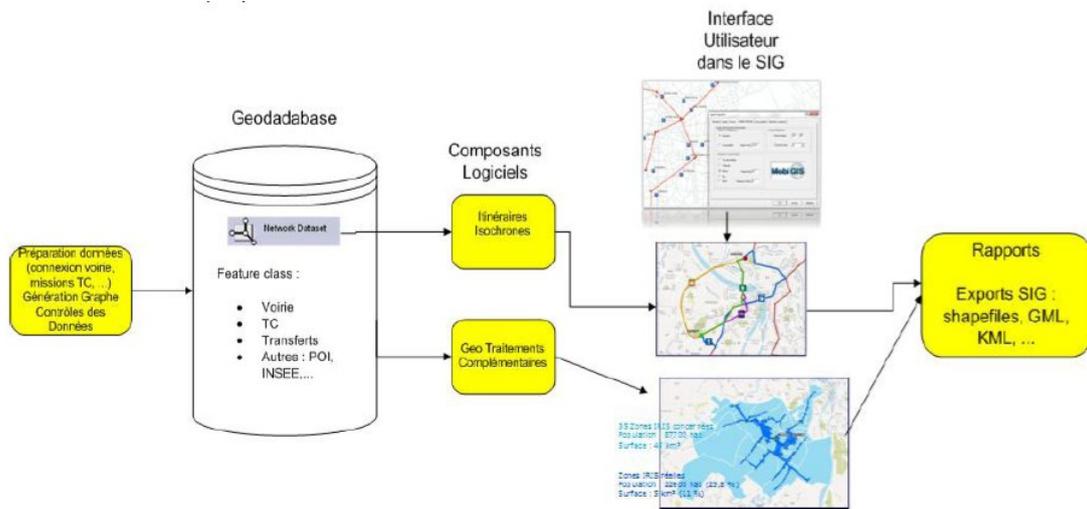


Figure IV.2 – Architecture de l'application

- le graphe généré à partir de ces données permet d'effectuer des calculs d'itinéraires ou d'accessibilité.

Des fonctions de contrôle des données sont proposées à l'utilisateur : par exemple contrôle de la topologie du réseau ou contrôle des connexions inter-modes.

4.1.2 Composants applicatifs

Les composants applicatifs constituent le cœur du logiciel. Les fonctions métiers sont assurées par ce niveau d'architecture :

- Calcul d'itinéraire monomodal et multimodal ;
- Calcul d'itinéraire sur des matrices Origine / Destination ;
- Calcul d'accessibilité (isochrones) ;
- Géo-traitements complémentaires ;
- Génération des résultats ;

Pour le moment, seul l'algorithme **MLMH** sans les règles de dominance est intégré dans le logiciel industriel. Des travaux sont en cours pour l'industrialisation d'autres variantes.

4.1.3 Interface utilisateur

L'interface utilisateur permet de saisir facilement les paramètres et d'afficher les résultats obtenus. Elle permet en effet :

- l’affichage des données sous forme de cartes ;
- la mise en forme adaptée des données et des résultats (couleurs, symboles, taille des éléments géographiques, etc.) ;
- la constitution des paramètres géographiques des traitements (points de départ / arrivée pour un calcul d’itinéraire par exemple) à l’aide de la carte et de fonctions avancées de navigation ;
- l’affichage des résultats ;
- l’interopérabilité des données et standards des SIG : import / export dans des standards de l’OGC (consortium international pour le développement et promotion de standards ouverts se basant sur les spécifications OpenGIS®) pour une exploitation dans l’environnement SIG ESRI ou pour une utilisation des résultats dans des environnements différents (SIG MapInfo, Google Earth, QGIS etc.) ou également de produire des rapports de type tableur.

Les résultats obtenus, quelle que soit leur forme, visent à être évolués, faciles à interpréter et propices à la prise de décision ou à la communication. La cartographie est en cela un moyen très efficace pour atteindre ces objectifs.

4.2 Diagramme des classes

Le diagramme des classes constitue un élément fondamental de conception de logiciel. Il définit l’ensemble des composants du système final. Il aide à structurer le travail de développement de manière efficace et permet aussi de construire le système de manière correcte en identifiant la structure des classes du système ainsi que les propriétés et les méthodes de chaque classe. Dans le cas de la conception du logiciel, le diagramme est illustré par la figure IV.3 qui identifie l’ensemble des composants de ce système. Les propriétés et les méthodes de ces classes sont détaillées dans la section suivante.

4.3 Description des classes

Notre implémentation utilise de manière intensive le polymorphisme et l’héritage de classe mais aussi les patrons de classe (template en anglais), encore appelées classes paramétrées. Chaque classe est paramétrée sur le type de données qu’elle peut contenir.

4.3.1 Solvers

C’est la classe dont héritent tous les algorithmes. Elle décrit l’ensemble des attributs et des méthodes propres aux différents algorithmes de calcul d’itinéraires ou d’accessibilité. Elle est

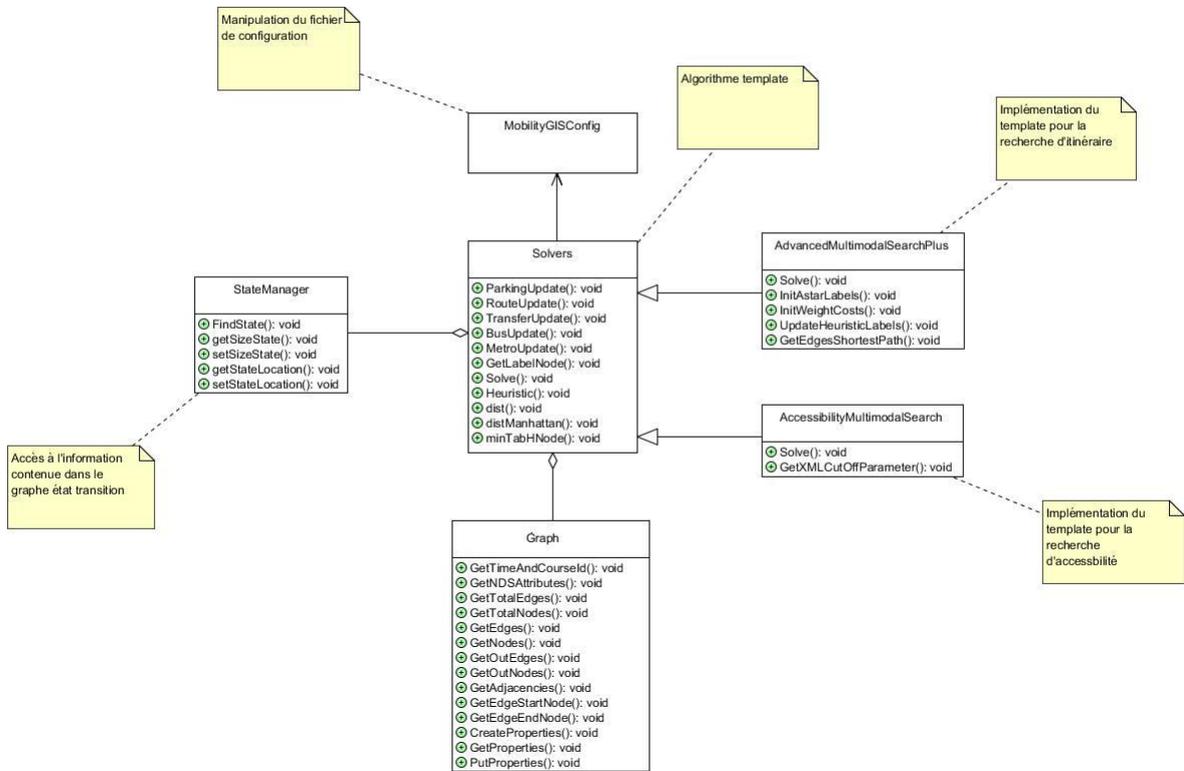


Figure IV.3 – Diagramme de classes

composé d'objets de la classe *StateManager* et de la classe *Graph*.

4.3.2 StateManager

Cette classe gère les automates permettant de représenter la viabilité des chemins.

4.3.3 Graph

La classe graphe correspond à l'utilisation de *NetworkAnalystGraph* pour représenter le réseau multimodal. Elle contient les liens vers la couche de persistance en mémoire d'ArcGIS et utilise directement l'API C++ d'ArcGIS pour implémenter l'interface de parcours de graphe. Celle-ci est ensuite utilisée par les algorithmes.

5 Paramétrages du logiciel

Une bonne interface graphique est un élément essentiel pour les systèmes de calcul d'itinéraires. Elle permet à l'utilisateur d'interagir avec le système en utilisant l'ensemble de ces fonctionnalités. L'interface graphique du logiciel est une interface d'un modèle ergonomique simple divisé en trois parties (cf. figure IV.4) :

- la première partie est réservée à la sélection des modes de transports souhaités (voiture, piéton, bus, métro, tramway, ferré). La sélection des modes est multiple permettant donc de conjuguer plusieurs modes de transport. Quand un seul mode est sélectionné, le problème devient monomodal ;
- La deuxième partie est consacrée à la définition des paramètres des algorithmes proposés. Ces paramètres sont dépendants de la fonction d'analyse utilisée (itinéraires ou accessibilité) ;
- la troisième partie est dédiée à la génération des résultats et est également liée à la fonction d'analyse sélectionnée.

6 Modélisation du réseau de transport multimodal

6.1 Modélisation des modes de transport

Avant d'utiliser les fonctions de calcul d'itinéraires, il est nécessaire de modéliser en amont le réseau de transport à partir des données disponibles.

La création du graphe logique et géométrique (graphe multimodal interconnecté), est effectuée

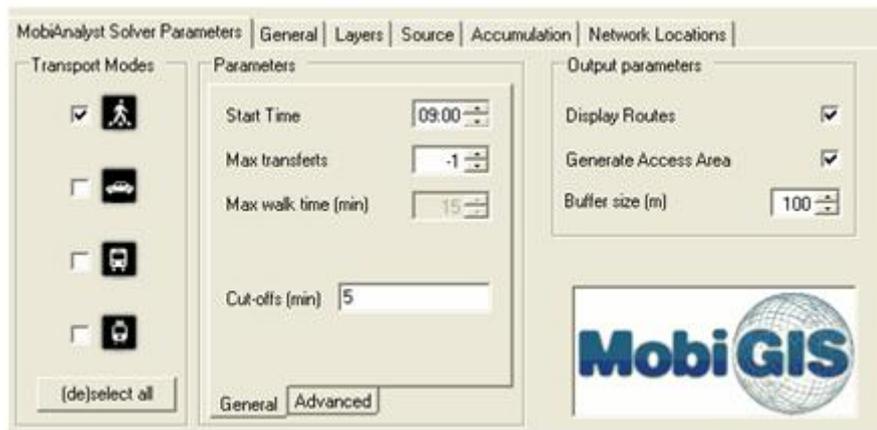


Figure IV.4 – Interface utilisateur

à l'aide de l'extension Network Analyst (NA) d'ESRI. A partir de ce logiciel, on crée une géo-database où sont stockées les entités géométriques des jeux de données qui constituent les sources des réseaux de transport (tronçons TC, tronçons de voirie, pistes cyclables, arrêts TC, arrêts projetés sur la voirie, parking relais, parking vélo, transferts, horaires TC).

Avant de créer un réseau de transport multimodal, il est nécessaire de comprendre comment ce réseau peut être modélisé. Le réseau que nous avons créé est constitué de trois ensembles d'éléments : la voirie, les éléments de bus et les éléments de métro. Il est également possible d'ajouter à ces ensembles ceux des éléments de navigation fluviale, aérienne, de tramway, etc. La voirie constitue l'élément principal du réseau, et permet d'utiliser les modes de transport individuel tels que :

- le véhicule particulier ;
- le vélo ;
- la marche à pieds.

De plus, c'est la voirie qui permet de connecter tous les autres modes de transports entre eux. En effet, pour passer du mode bus au mode métro, on passe obligatoirement par la voirie.

Le bus et le métro sont des modes de transport en communs utilisés pour se déplacer dans un environnement urbain et ce afin d'éviter d'utiliser les modes de transports personnels. Contrairement aux modes se basant sur la voirie, ces modes sont soumis à des contraintes d'horaires ou de fréquence.

Pour être efficace et se rapprocher au plus juste des conditions de transport réelles, il est donc nécessaire que le réseau créé prenne en compte les horaires et les fréquences pour les modes de transport en commun.

IV.6 Modélisation du réseau de transport multimodal

Les lignes de métro sont tracées par des tronçons à vol d'oiseaux entre les différentes stations. Le métro est également soumis à la contrainte de fréquence qui doit pouvoir varier suivant les heures de la journée.

D'autres modes de transport peuvent être ajoutés au réseau afin de le rendre le plus complet possible. La figure IV.5 présente un schéma d'un réseau de transport multimodal : elle montre

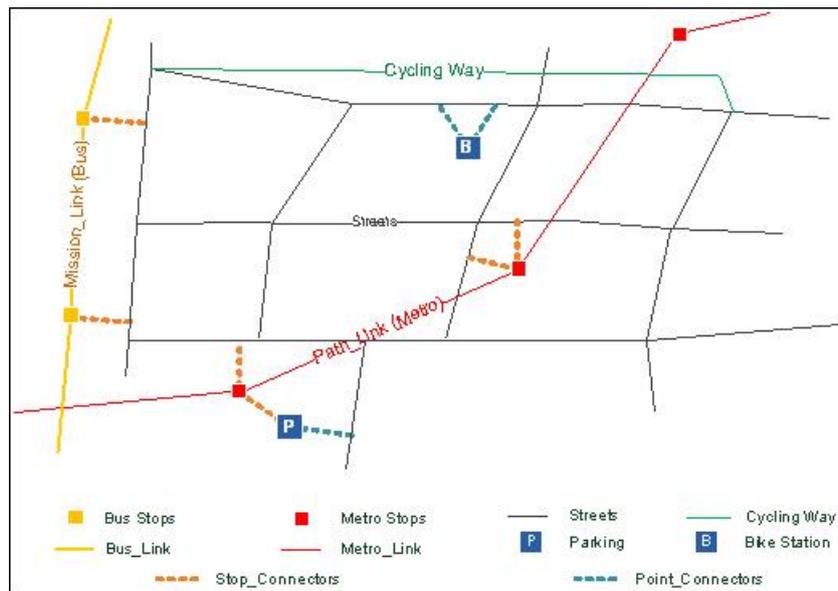


Figure IV.5 – Réseau de transport multimodal

que tous les modes de transport sont connectés entre eux par la voirie. Cependant, pour passer d'un mode utilisant la voirie à un mode de TC ou un autre, il est nécessaire d'utiliser des éléments intermédiaires que sont les transferts.

6.2 Création du réseau multimodal

6.2.1 les données du réseau TC

Les données du réseau de TC sont généralement de type GTFS (Google Transit Feed Specification) ou Trident. Pour pouvoir utiliser les données au format Trident, nous utilisons le logiciel CHOUETTE³(Création d'Horaires avec un Outil d'Échange de données TC selon le format Trident Européen) pour importer ces données dans la base de données PostgreSQL qui est un système de gestion de bases de données relationnelles et objet (SGBDRO). Dans PostgreSQL,

3. <http://www.chouette.mobi/>

Chapitre IV. Travaux d'Industrialisation et logiciel développé

il faut maintenant créer et définir la géométrie (ligne, point, etc.) des couches géographiques nécessaires à la création du réseau logique. Ces couches sont :

- la couche *StopArea* des arrêts de bus physiques du réseau TC ;
- la couche *PathLink* des tronçons qui relient les *StopArea* entre eux. Sur ces tronçons sont notamment calculées les vitesses moyennes ;
- la couche *StopAreaShifted* des arrêts de bus dupliqués et décalés en fonction des missions. Une mission est associée à un itinéraire (voir annexe C). Chaque mission possède son propre *StopAreaShifted* ;
- la couche *MissionLink* des tronçons reliant les *StopAreaShifted* entre eux selon les missions auxquelles ils appartiennent ;

Les horaires des *StopAreaShifted* pour une date donnée (issue de la table *VehiculeJourneyAtStop* de Trident) sont stockés dans une table appelée *VjatDate*.

Les données du mode métro, comme les données du bus sont souvent dans le même format de données. Les données du métro utilisés ici sont sous format de fichier. Elles sont constituées de trois couches :

- la couche *MetroStations* des stations de métro ;
- la couche *MetroPathLink* des tronçons reliant les stations entre elles ;
- la couche *MetroStopsConnector* qui relie des lignes de métros différents.

De même que le mode bus, pour relier le mode métro à la voirie, il est nécessaire de créer des stations projetées sur la voirie et les liens de connexions. Deux couches permettent de gérer la connexion de ces deux réseaux (métro et voirie) :

- la couche *MetroStopsProjected* des stations projetées ;
- la couche *MetroStopsTransfert* des tronçons connectant les stations du métro à la voirie.

Ces différentes couches, une fois exportées, peuvent être visualisées et localisées sous la forme d'une carte vectorielle (constituée par un assemblage de points, lignes et polygones) et ont une géométrie projetée en WGS 84 (World Geodetic System 1984 est le système géodésique associé au GPS qui est une référence "standard" pour la cartographie). Il est nécessaire de reprojeter ces couches en coordonnées Lambert⁴.

Ces couches, une fois importées dans un projet ArcMap⁵, ainsi que la couche de voirie qui contient l'ensemble des informations indispensables à la constitution d'un réseau routier (vitesse de circulation, sens de circulation, restrictions véhicules, nom des rues) montre qu'il manque plusieurs éléments pour connecter correctement le réseau bus au réseau Voirie (cf figure IV.6)

4. http://fr.wikipedia.org/wiki/Projection_conique_conforme_de_Lambert (projection officielle pour les cartes de France métropolitaine depuis le décret du 26 décembre 2000)

5. logiciel D'ESRI pour le traitement des données spatiales

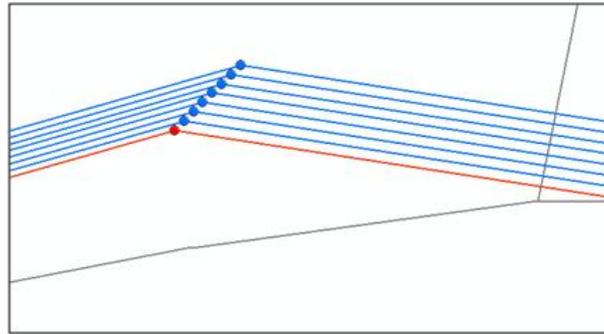


Figure IV.6 – tronçons de bus (tronçons bleus et rouges) non connectés à la voirie (en gris)

Pour cela, il faut créer d'autres couches permettant cette liaison :

- la couche *StopAreaProjected* des projections des *Stoparea* sur la voirie la plus proche ;
- la couche *StopAreaTransfert* qui regroupe les tronçons de transferts reliant les *StopArea* aux *StopAreaProjected* ;
- la couche *StopAreaShiftedTransfert* des tronçons de transfert des *StopAreaShifted* aux *StopAreaProjected*.

Une fois ces nouvelles couches ajoutées au projet ArcMap, les éléments du bus (arrêts et tronçons) sont connectés à la voirie (cf. figure IV.7).

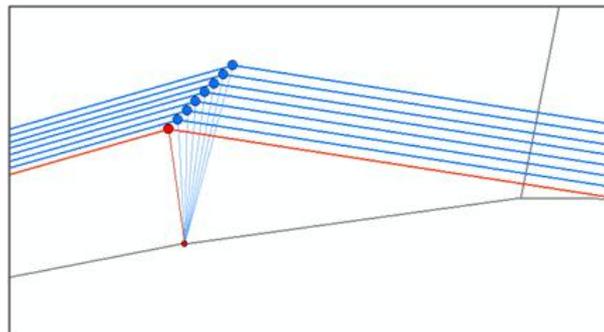


Figure IV.7 – connexion du mode bus à la voirie

6.2.2 Création des autres couches du réseau multimodal

De plus, il est nécessaire d'introduire des parking relais pour le stationnement des véhicules. Il est donc important de prendre en compte dans la modélisation l'utilisation de ces parking relais. Pour cela deux couches sont utilisées :

- la couche *ParkingRelais* pour les différents parking relais du métro ;

- la couche *TransfertParkingVoirie* pour connecter la station métro à son parking relais correspondant.

La figure IV.8 décrit cette modélisation. La station de métro est connectée à la voirie et au parking relais par un arc de transfert (station-Rue). Le parking est relié à la voirie par un transfert (Parking-Voirie).



Figure IV.8 – Représentation d'une station de métro avec parking relais

Pour les modes piéton et voiture on utilise les couches suivantes :

- la couche *Streets* qui regroupe l'ensemble des tronçons des modes piéton et voiture ;
- la couche *StreetsNodes* des nœuds de la voirie.

6.2.3 Récapitulatif des éléments du réseau multimodal

Le tableau III.1 récapitule l'ensemble des couches nécessaires à la construction des réseaux de transport multimodaux qui connectent les quatre modes que nous avons considéré dans notre application (bus, métro, piéton et voiture).

Au total, quinze couches sont nécessaires à la création de notre réseau de transport pour connecter les différents modes (voiture, piéton, bus et métro) entre eux et rendre le graphe connexe (Tableau IV.1).

Plus précisément, les trois couches suivantes permettent de connecter les modes entre eux :

- *BusStopsProjected* : connecte le bus au mode piéton
- *MetroStopsProjected* : connecte le métro au mode piéton
- *StreetsNodes* : connecte le mode voiture au mode piéton

Il n'y a pas de relation directe entre les modes bus ou métro et le mode voiture. En effet, pour passer du mode voiture au mode bus ou métro, il est obligatoire de passer par le mode piéton (par les transferts).

C'est avec l'ensemble de ces différentes couches qu'on crée la base de données spatiale (géo-database) dans laquelle on va définir la création du graphe qui modélise le réseau de transport

IV.6 Modélisation du réseau de transport multimodal

NOM	TYPE	MODES	DESCRIPTION
<i>MissionLink</i>	Line	Bus	Tronçons de missions des bus
<i>PathLink</i>	Line	Bus	Tronçons des bus
<i>StopArea</i>	Point	/	Arrêts physiques des bus
<i>StopareaShifted</i>	Point	/	Arrêts physiques des bus
<i>StopareaShiftedTransfert</i>	Line	Piéton	Tronçons de connexion Bus - Voirie
<i>StopareaProjected</i>	Point	/	Arrêts physiques des bus projetés
<i>MetroPathLink</i>	Line	Metro	Tronçons des métros
<i>MetroStations</i>	Point	/	Stations de métro
<i>MetroStopsTransferts</i>	Line	Piéton	Transferts inter-stations de métro
<i>MetroStopsConnectors</i>	Line	Piéton	Tronçons de connexion MetroStops <-> Voirie
<i>MetroStopsProjected</i>	Point	/	Stations de métro projetées
<i>Streets</i>	Line	Piéton/VP	Tronçons de voirie
<i>StreetsNodes</i>	Point	/	nœuds de voirie
<i>TransfertParkingVoirie</i>	Line	/	Transfert entre la voirie et le parking
<i>ParkingRelais</i>	Point	/	les parkings relais de métro

Tableau IV.1 – Récapitulatif des éléments constituant le réseau

multimodal. Cette étape est importante puisqu'elle permet d'établir les relations entre les différents modes de déplacement et donc de rendre le réseau connexe. Mais aussi, c'est dans cette partie que nous allons définir les attributs pour paramétrer toutes les informations nécessaires aux calculs d'itinéraires ou accessibilité (sens de circulation, horaires de départ, mode de transport, temps de parcours, restrictions particulières).

6.2.4 Le graphe de la geo-database

Le réseau logique ou NetworkDataSet correspond au graphe que doit explorer l'algorithme pour trouver une solution au problème de recherche d'itinéraires. Dans ce réseau logique, nous définissons :

- Les sources qui comprennent l'ensemble des classes d'entités issues des couches définies ci dessus (cf figure IV.9).
- Les connexions qui permettent de décrire les règles de connectivité : intégrité du réseau. Elles définissent les connexions entre les éléments logiques, la cardinalité de ces liens et les sous types de jonctions permises entre ces éléments (cf. figure IV.10).
- Les attributs qui définissent les paramètres du réseau logique (cf. figure IV.11). Ces paramètres peuvent être de type :
 - Coût : qui sert généralement à définir les impédances qui représentent le coût associé à

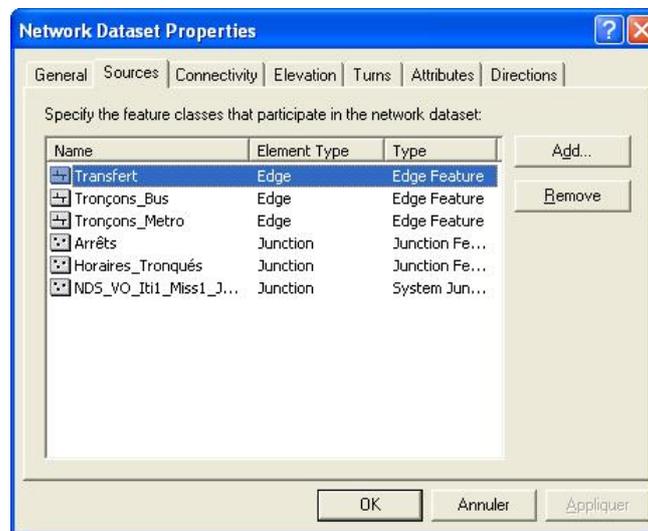


Figure IV.9 – Les sources du graphe logique multimodal (ArcGIS)

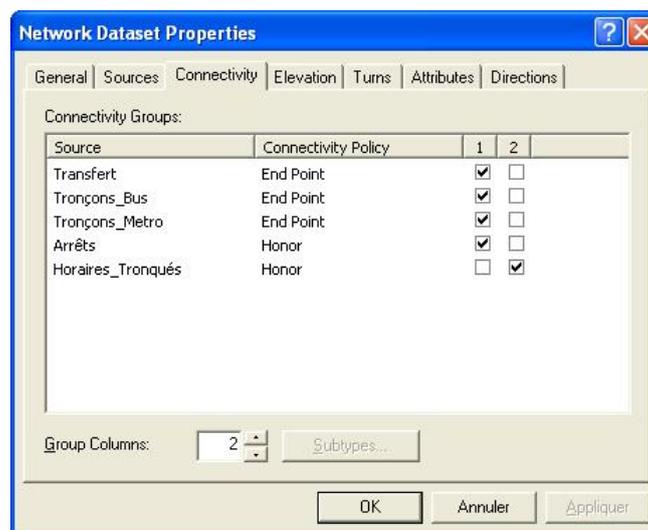


Figure IV.10 – Les connexions pour définir le réseau multimodal (ArcGIS)

la traversée d'un élément du réseau (temps de trajet par exemple). Elles sont définies lors de la création du réseau et ne peuvent être modifiées par la suite ;

- Restriction : chaque élément d'un réseau peut être actif ou inactif. Un élément inactif ne sera pas utilisé lors de l'analyse du réseau. Le statut est stocké dans un champ ajouté automatiquement à chaque classe géométrique participant au réseau lors de la création (par exemple les restrictions pour les sens de circulation) ;
- Descripteur : pour caractériser des types ou des clés pour les éléments du réseau.

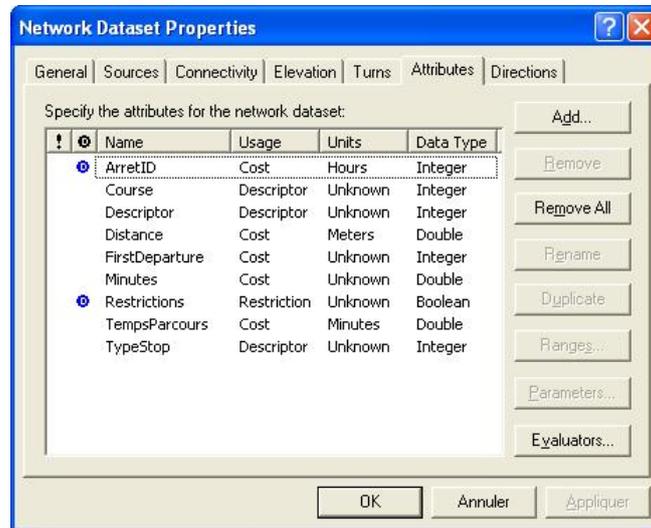


Figure IV.11 – Les attributs du graphe logique multimodal (ArcGIS)

7 Exemples d'utilisation

7.1 Visualisation d'un réseau TC

Le réseau de transport est constitué d'objets géographiques complexes stockés dans des bases de données et auxquels sont associés des attributs topologiques, géométriques et métriques. Ainsi pour l'organisation, l'exploration et la représentation d'informations géographiques d'un réseau de transport, il devient impératif de comprendre les propriétés structurelles du réseau. Dans le domaine des transports en commun, le logiciel offre des solutions de modélisation et de représentations des lignes de bus, de métro etc.

La figure IV.12 met en évidence l'ensemble des lignes de bus, du métro et des TER de la ville de Toulouse. Chaque ligne est modélisée de façon propre en fonction de ses propriétés (lignes à horaires ou ligne à fréquence). Ainsi la figure IV.13 représente une ligne de bus où les horaires de passages aux différents arrêts sont modélisés par une table. La figure IV.14 donne une vision cartographique modélisant une ligne pour les itinéraires aller et retour et les noms des arrêts de la ligne.

Le logiciel offre aussi la possibilité de mener des études d'indicateurs dans le domaine du transport enfin de comprendre les offres de transport proposées aux usagers. Ainsi la figure IV.15 montre un exemple d'étude d'analyse sur le nombre de dessertes par arrêts sur une journée. Ceci permet de connaître la fréquence dans certaines zones de l'offre de transport en commun.

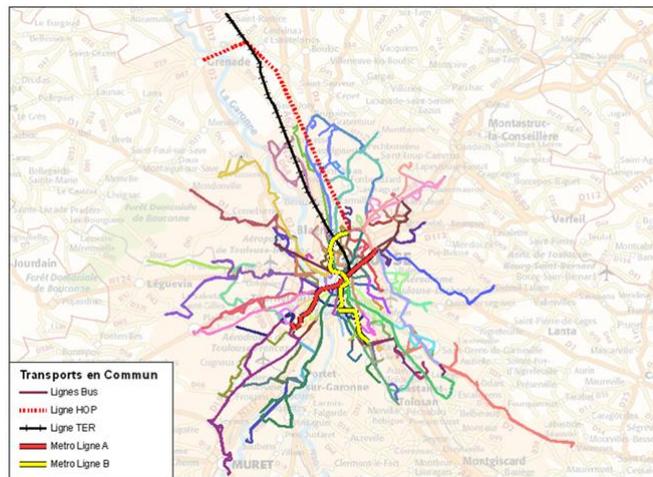


Figure IV.12 – Visualisation d'un réseau TC

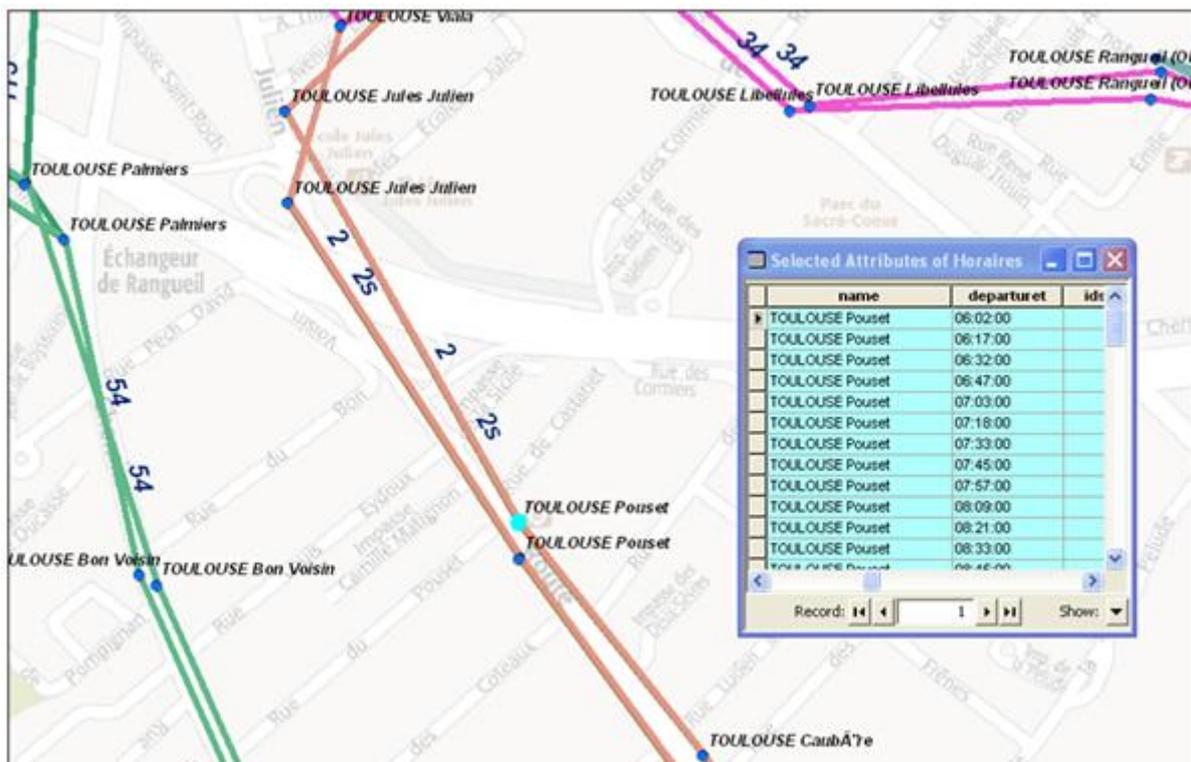


Figure IV.13 – Zoom de l'offre TC avec horaires à un arrêt

7.2 Recherche d'itinéraires et matrices OD

Les algorithmes présentés dans le chapitre II sont intégrés avec des fonctionnalités supplémentaires comme le calcul de l'émission de CO2 a posteriori.

IV.7 Exemples d'utilisation

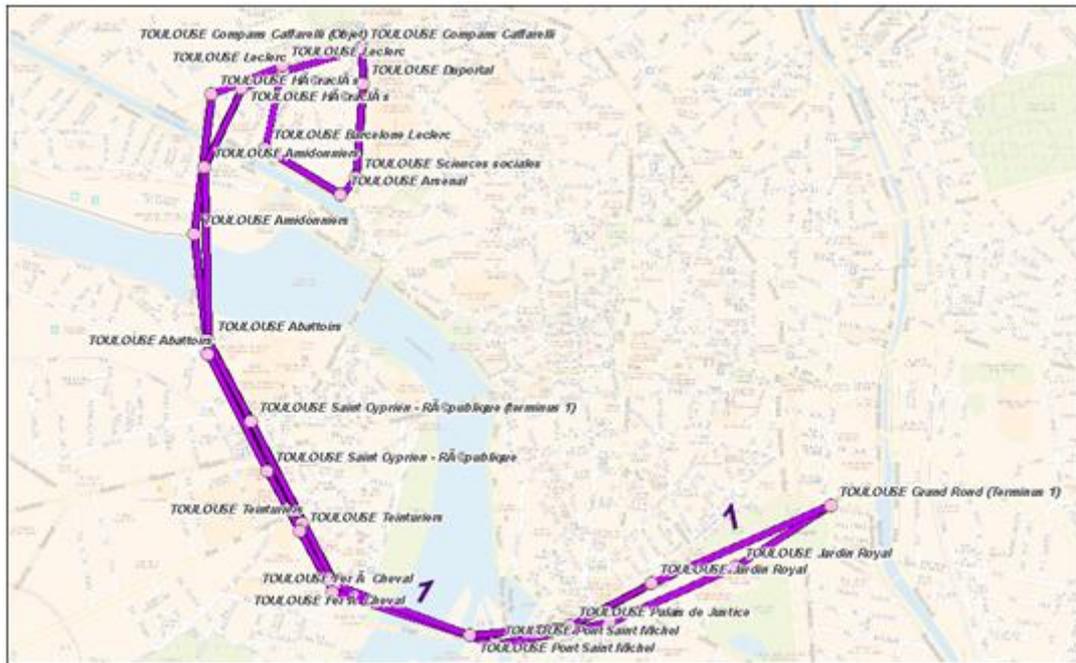


Figure IV.14 – Représentation cartographique d'une ligne de bus

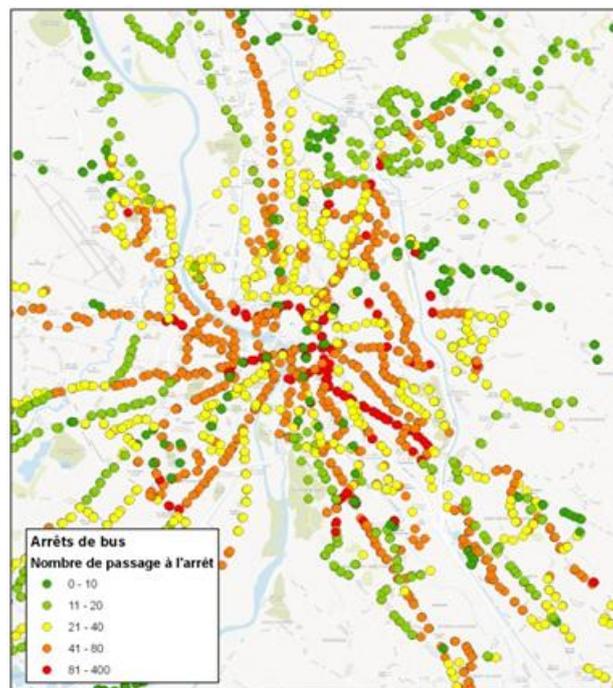


Figure IV.15 – Nombre de passages par arrêt

Chapitre IV. Travaux d'Industrialisation et logiciel développé

Les figures IV.16 et IV.17 fournissent l'ensemble des chemins multimodaux issus d'un calcul de plus court chemins entre une origine et une destination. Ces chemins sont fonctions des modes utilisés et du temps de trajets et sont calculés avec l'algorithme MLMH présenté dans le chapitre II.

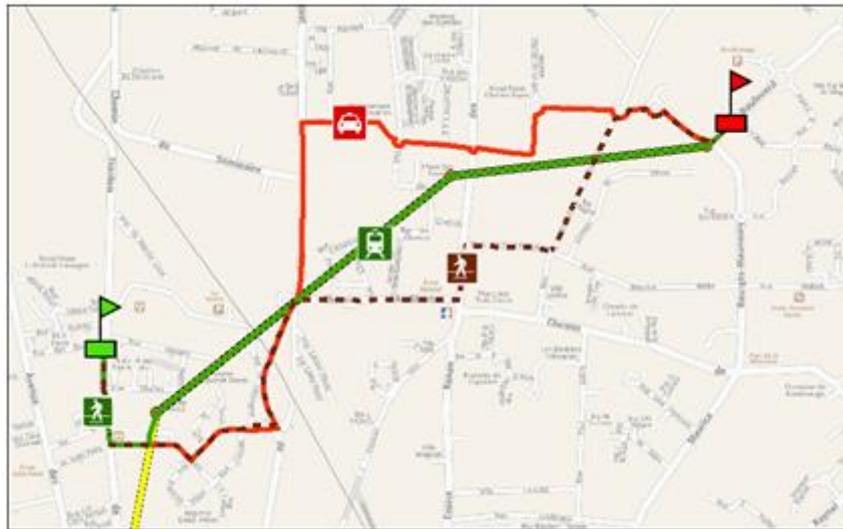


Figure IV.16 – Visualisation d'itinéraires multimodaux

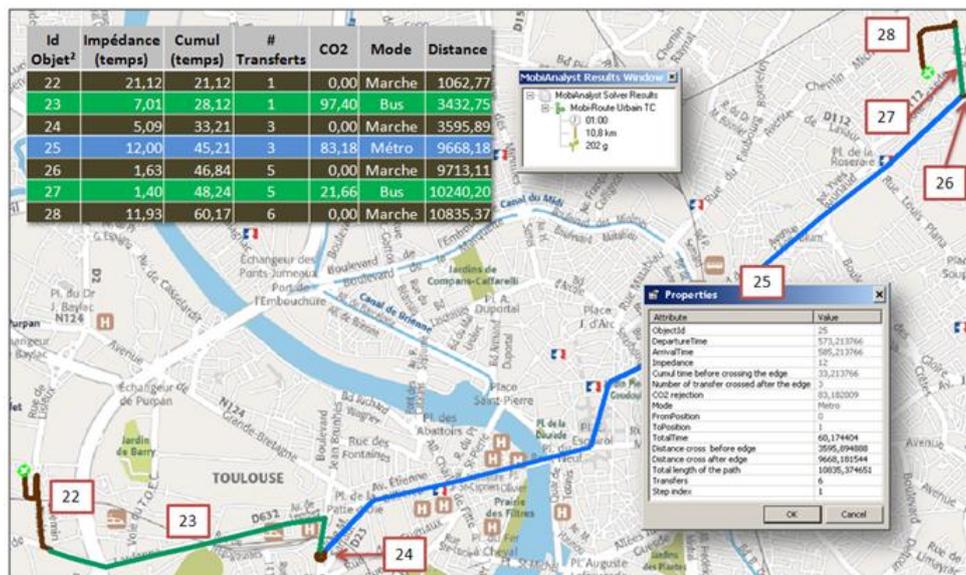


Figure IV.17 – Calcul d'itinéraire multimodal - Résultats proposés à l'utilisateur

Au delà des calculs d'itinéraires, le logiciel offre des fonctionnalités pour le calcul des matrices origines-destinations multimodales qui jouent un rôle important dans les études de trafic. Elles

permettent d'analyser les déplacements dans le temps et dans l'espace.

Les figures IV.18 et IV.19 montrent des exemples de résultats des calculs de matrices OD respectivement pour la marche et pour le transport en commun.

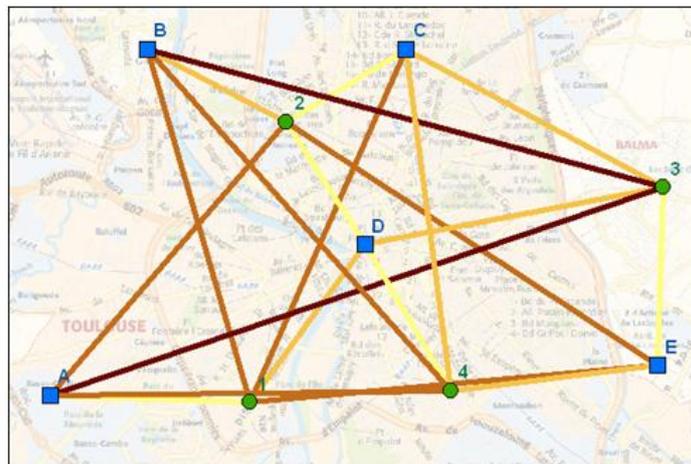


Figure IV.18 – Calcul de matrices Origines/Destinations : Impédance = temps de marche

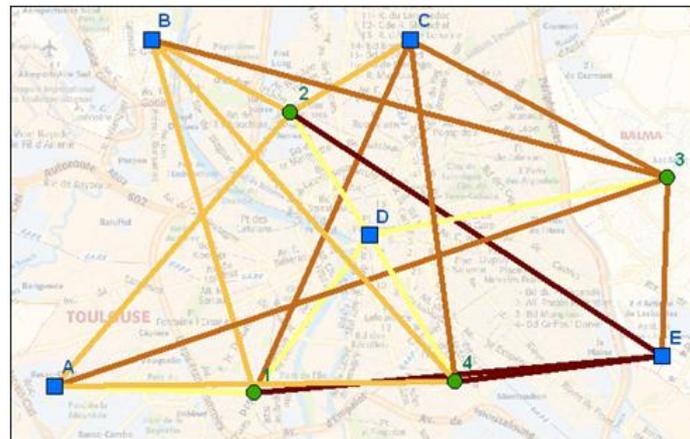


Figure IV.19 – Calcul de matrices Origines/Destinations : Impédance = temps en TC

7.3 Calcul d'accessibilité ou isochrones

Le calcul d'accessibilité ou d'isochrones consiste à déterminer l'ensemble des zones atteignables pour une durée donnée à partir d'un endroit fixé. Les figures IV.20, IV.21, IV.22 et IV.23 ci-dessous sont des exemples de calcul d'isochrones.

Chapitre IV. Travaux d'Industrialisation et logiciel développé

La figure IV.20 est l'accessibilité multi-seuils, elle détermine l'ensemble des zones accessibles à partir de la place Esquirol (centre ville de Toulouse) en combinant le transport en commun et la marche à pied suivant différents seuils de temps. La couleur définit l'importance de l'intervalle de durée, plus elle est foncée, plus l'intervalle est grand. Cette figure indique pour une personne non véhiculée l'ensemble des zones atteignables depuis *place Esquirol* en moins de 40 minutes.

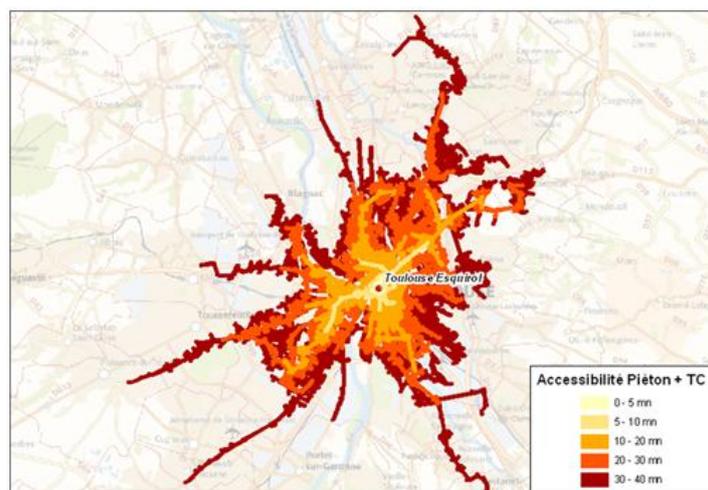


Figure IV.20 – Accessibilité multi-seuils

La figure IV.21 est une autre forme d'isochrone qui est calculée en fonction du ou des modes utilisés. C'est une isochrone mono et multimodale. Elle détermine les zones à moins de 10 minutes à partir de *Toulouse Esquirol* en fonction des modes bus, piéton+bus, piéton+bus+marche et en voiture mais aussi en fonction des horaires (heures creuses et heures de pointes). La figure IV.22 est la même isochrone montrant les différents arcs ou tronçons atteints entre 10 et 20 minutes.

La figure IV.23 est une autre forme de représentation de l'isochrone précédent en montrant les points d'intérêt (POI) situées dans la zone d'accessibilité.

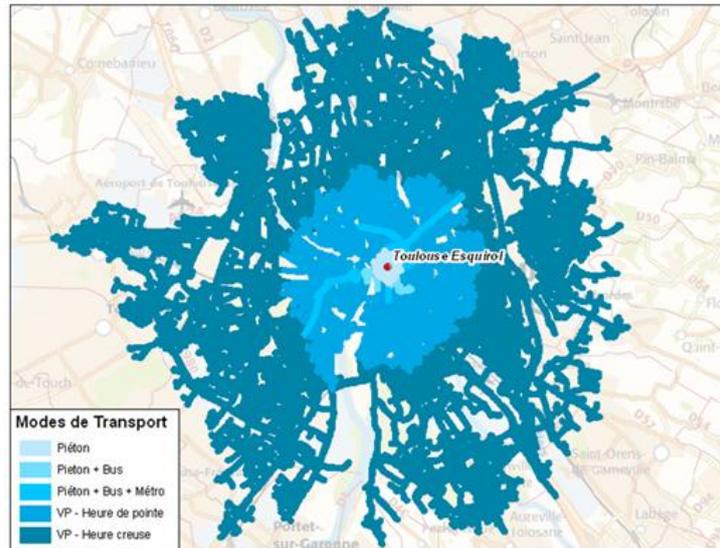


Figure IV.21 – Accessibilité en 10 minutes selon différents modes

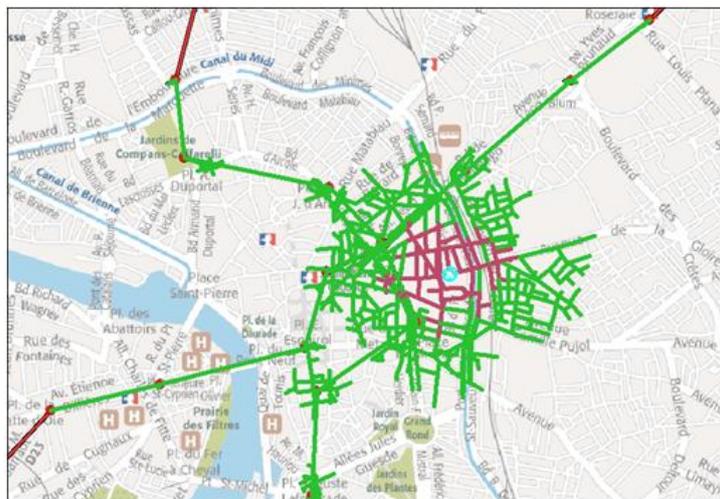


Figure IV.22 – Accessibilité piétonne 10 et 20 mn - Représentation sous forme de tronçons parcourus

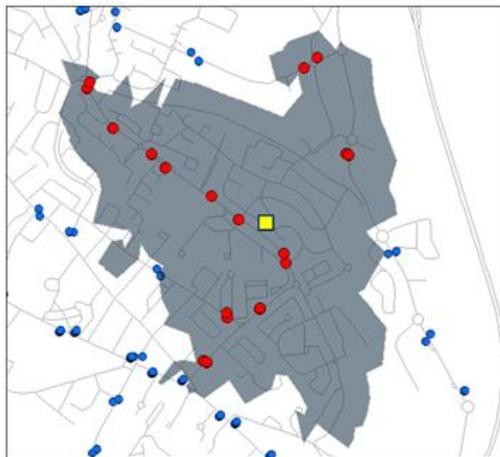


Figure IV.23 – Points d'intérêts situés dans une zone d'accessibilité

8 Conclusion

Dans ce chapitre, nous avons présenté le logiciel industriel permettant l'organisation et la planification des transports en exploitant la multimodalité des réseaux de transport. Ainsi, après une description détaillée des objectifs et fonctionnalités de cet outil, nous avons évoqué la conception et la mise en œuvre de ce dernier.

Un des algorithmes de recherche d'itinéraires multimodaux mis au point au cours de la présente thèse a été utilisé pour établir des calculs d'itinéraires et d'accessibilité.

Les travaux de recherche présentés au chapitre II (règles de dominance, approches A^* ou bidirectionnelle) offrent un potentiel d'amélioration pour ce logiciel qui devront permettre d'effectuer des traitements sur des réseaux de transports correspondant aux plus grandes agglomérations mondiales.

Conclusion générale et perspectives

Dans ce mémoire, nous nous sommes consacrés à la proposition d'algorithmes de calcul d'itinéraires en transport multimodal.

Dans un premier temps (chapitre 1), nous avons tout d'abord présenté un état de l'art sur la modélisation des réseaux de transport. Puis nous avons exposé les algorithmes de la littérature pour résoudre différents problèmes de plus courts chemins dans des réseaux de transport monomodaux et multimodaux, statiques ou dépendants du temps.

Dans le chapitre 2, nous avons présenté le problème qui nous a intéressé pendant cette thèse : celui du calcul de plus courts chemins bi-objectifs viables dans des réseaux de transport multimodaux. Les deux objectifs considérés ont été la minimisation du temps de trajet et la minimisation du nombre de changement de modes. Puis nous avons décrit la modélisation du réseau de transport par un graphe multi-couches et des contraintes de viabilité par un automate à états finis.

Pour la résolution du problème, nous avons d'abord constaté que les méthodes existantes n'avaient jamais été validées sur des réseaux de grande taille. Nous avons apporté des améliorations à l'algorithme de plus court chemin « topologique » existant puis nous avons proposé un nouvel algorithme basé sur des « labels multiples ». Nous avons notamment défini de nouvelles règles de dominance basées sur les états de l'automate de viabilité. Nous avons montré que l'algorithme multi-labels MLMH pouvait facilement être étendu sous une forme bidirectionnelle contrairement à l'algorithme topologique. Ceci nous a conduit à étendre la condition d'optimalité de l'algorithme bidirectionnel existant dans le cas de réseaux monomodaux et à proposer différentes approches pour la construction de l'automate de viabilité inverse nécessaire

Conclusion générale et perspectives

à la phase de recherche arrière. Enfin, des versions A^* de chacun de ces algorithmes ont été définies.

Le chapitre 3 a été consacré aux expérimentations et à la comparaison de ces différents algorithmes. Pour ces expérimentations, nous avons considéré deux types de réseaux de transport de la ville Toulouse. Un réseau de transport dit statique et un réseau dépendant du temps et respectant le principe de FIFO.

Les expérimentations ont montré que l'algorithme bidirectionnel proposé obtient les meilleures performances dans tous les cas. On a pu également constater que la règle de dominance basée sur les états que nous avons proposée a eu une influence significative sur la diminution du temps de calcul. Notons que dans tous nos tests, plusieurs solutions non dominées (relativement à la minimisation du temps de trajet et du nombre de transferts) ont été trouvées dans des temps de calcul satisfaisants (de l'ordre de la seconde) ce qui souligne l'intérêt de cette recherche de compromis sur un réseau de transport réel.

Dans le chapitre 4, nous avons abordé l'industrialisation des travaux de recherche réalisés. Nous avons décrit la conception et la réalisation d'un logiciel couplé à un système d'information géographique et visant à fournir des outils d'aide à l'analyse de la mobilité sur des réseaux de transport multimodaux. Un des algorithmes de calcul d'itinéraires que nous avons proposé est intégré dans plusieurs des fonctions de ce logiciel : calcul d'itinéraires point à point, accessibilité (isochrone), calcul de distancier, etc.

Plusieurs perspectives peuvent être identifiées suite aux travaux que nous présentons dans ce mémoire.

Les validations actuellement effectuées n'ont porté que sur des réseaux de taille moyenne, ce qui est déjà un apport par rapport aux approches existantes dans le cas multimodal. Toutefois, l'extension à de grands graphes nécessitera probablement la mise au point de techniques d'accélération intégrant les aspects multimodaux. Cela concerne notamment les techniques de prétraitement telles que les *landmarks* et les méthodes de contaction de réseaux.

Parmi ces techniques d'accélération, nous avons utilisé la méthode A^* pour laquelle des améliorations devront être trouvées notamment pour conjuguer les cas dépendant du temps et multimodaux. Nous avons proposé quelques pistes en ce sens qui ne se sont pas révélées suffisantes.

Une meilleure prise en compte des spécificités des réseaux dépendants du temps doit égale-

ment être envisagées par exemple pour la différenciation des modes dans le calcul des estimations de temps de trajet dans la recherche inverse des algorithmes bidirectionnels.

D'autre part, nous nous sommes limités à des réseaux de transport FIFO. L'extension de nos algorithmes dans le cas non FIFO est à étudier.

Nous avons proposé, d'un coté, une règle de dominance basée sur les états de l'automate de viabilité qui s'est révélée efficace. Cela souligne le potentiel d'exploitation de la structure de l'automate pour une réduction supplémentaire de l'espace de recherche.

Il paraît indispensable à la suite de ces travaux d'étendre, pour un réseau de transport donné, les expérimentations effectuées à d'autres automates afin d'évaluer la robustesse des algorithmes face à des contraintes de viabilité issues de préférences de l'utilisateur.

D'un autre coté, nous avons mis en évidence l'intérêt de la mise sous forme déterministe de l'automate inverse pour la diminution du nombre de labels générés dans l'algorithme bidirectionnel. Or, cette opération peut provoquer une explosion du nombre d'états pouvant annuler l'amélioration escomptée. Une des perspectives de notre travail consiste en la recherche de l'obtention d'un automate inverse déterministe avec un nombre d'états minimum.

Enfin, en termes d'industrialisation, des travaux sont à mener pour permettre l'obtention d'automates à partir de contraintes ou préférences exprimées par des utilisateurs.

Le problème bi-objectif que nous avons étudié peut être considéré comme le problème multimodal de base. D'autres objectifs liés à la multimodalité devront être étudiés tels que les couts de trajet, l'impact sur l'environnement, le confort, Cependant, le caractère polynomial du problème risque d'être perdu et sa résolution devra faire appel à d'autres méthodes d'optimisation combinatoire. Il serait toutefois intéressant de trouver d'autres problèmes polynomiaux pertinents.

Au delà des problèmes de recherche d'itinéraires, les modèles et les algorithmes multimodaux présentés dans cette thèse pourront être exploités pour des problèmes de flots ou de tournées multimodales.

Conclusion générale et perspectives

Bibliographie

- [1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows : Theory, Algorithms and Applications*. Prentice-Hall Englewood Cliffs, NJ, 1993. 17
- [2] R. Ahuja, J. Orlin, S. Pallottino, and M. Scutella. Minimum time and minimum costpath problems in street networks with periodic traffic lights. *Transportation Science*, 36(3) : 326–336, 2002. 25
- [3] ATEC/ITS. www.atec-itsfrance.net. 2
- [4] H. Ayed, Z. Habbas, and D. Khadraoui. Aco for solving a multimodal transport problems using a transfer graph model. *International Conference on Computers and Industrial Engineering (Troyes, France)*, 2009. 36
- [5] D. G. B. On the shortest route through a network. *Management Science*, 6(2) :pp. 187–190, 1960. ISSN 00251909. URL <http://www.jstor.org/stable/2627005>. 21
- [6] C. Barrett, R. Jacob, and M. Marathe. Formal-language-constrained path problems. *SIAM Journal on Computing*, 30(3) :809–837, 2000. 32
- [7] C. Barrett, K. Bisset, M. Holzer, G. Konjevod, M. Marathe, and D. Wagner. Engineering label-constrained shortest path algorithms. In *In 4th International Conference on Algorithmic Aspects in Information and management AAIM 2008 Shanghai China*, volume 5034, pages 27–37, 2008. 32, 33
- [8] G. Batz, R. Geisberger, and P. Sanders. Time dependant contraction hierarchies-basic algorithms ideas. Technical report, Faculty of Informatics, University Karlsruhe, 2008. Technical report, ITI Sanders. 24

Bibliographie

- [9] R. Bauer and D. Delling. Sharc : Fast and robust unidirectionnal routing. *ACM journal of Experimental Algorithms*, 2008. 24
- [10] D. Beauquier, J. Berstel, and P. Chrétienne, editors. *Elements d'algorithmique*. Masson, 1992. 18
- [11] R. Bellman. On a routing problem. *Quartely Applied Mathematics*, 16 :87–90, 1958. 18
- [12] M. Bielli, A. Boulmakoul, and H. Mouncif. Object modeling and path computation for multimodal travel systems. *European Journal of Operational research*, 175 :1705–1730, 2006. 9, 35
- [13] C. Boris, G. Andrew, and R. Tomasz. Shortest paths algorithms : Theory and experimental evaluation. *Mathematical Programming*, 73 :129–174, 1996. ISSN 0025-5610. 17
- [14] A. Bousquet and N. E. Faouzi. Temps de parcours multimodaux : vers une prise en compte complète de l'ensemble des modes et des temps d'interface dans le calcul de parcours urbain. *TEC*, 200, Décembre 2008. 12
- [15] A. Bousquet, S. Constans, and N. E. Faouzi. On the adaptation of a label-setting shortest path algorithm for one-way and two-way routing in multimodal urban transport networks. In *Proceedings of the International Network Optimisation Conference, Pise 2009*, 2009. 36
- [16] M. Boussejra, C. Bloch, and A. Moudni. Solution optimale pour la recherche du meilleur chemin intermodal. In **4^e conférence Francophone MOSIM*, 2003. 36
- [17] J. Brumbaugh-Smith and D. Shier. An empirical investigation of some bicriterion shortest path algorithms. *European Journal of Operational Research*, 43(2) :216–224, 1989. 31
- [18] CERTU. Déploiement national des systèmes d'information multimodale- transport direct : l'exemple anglais. Technical report, 2003. Collection du CERTU. 3
- [19] I. Chabini. A new shortest path algorithm for discrete dynamic networks. In *Proceedings of the 8th IFAC Symposium on Transport Systems*, pages 16–17, 1997. 10, 11, 12, 25
- [20] Y. Collette and P. Siarry. *Optimisation multiobjectif*. Edition EYROLLES, 2002. 28
- [21] K. Cooke and E. Hasley. The shortest route through a network with time-dependent intermodal transit time. *Journal of Mathematical Analysis and Applications*, 14 :493–498, 1966. 25

-
- [22] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms*. The MIT Press, 1992. 18
- [23] B. Dean. *Continuous-Time Dynamic Shortest Paths Algorithms*. PhD thesis, Master's thesis Massachusetts Institute of Technology, 1999. 25
- [24] D. Delling. Time-dependent sharc-routing. In *Proceedings of the 16th Annual European Symposium on Algorithms (ESA 08)*, volume 5193 of *Lecture Notes in Computer Science*, 2008. 24
- [25] D. Delling, T. Pajor, and D. Wagner. Accelerating multi-modal route planning by access-nodes. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA 09), Copenhagen (Danemark), Lecture Notes in Computer Science*, pages 587–598. Springer, 2009. 33
- [26] D. Delling, P. Sander, D. Schultes, and D. Wagner. Engineering route planning algorithms. In *algorithmics of large and complex networks. lecture notes in computer science*, pages 3–42. Springer, 2009. 24
- [27] E. Denardo and B. Fox. Shortest-route mehods 1 reaching , pruning and buckets. *Operations research*, 27 :161–186, 1976. 18
- [28] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis. Chapter 2 time constrained routing and scheduling. In C. M. M.O. Ball, T.L. Magnanti and G. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 35 – 139. Elsevier, 1995. 32
- [29] R. Dial. Algorithm 360 : Shortest path forest witch topological ordering. *Communications of the ACM*, 12 :632–633, 1969. 18
- [30] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959. 18
- [31] L. Diringuer. Sim en france : Etat de l'art et esquisse méthodologique pour l'assistance à maitrise d'ouvrage. *Master TURP*, 2005. 3
- [32] Y. Dissler, M. Muller-Hannemann, and M. Schnee. *Multicriteria Shortest Paths in Time-Dependant Train Networks*, volume 5038, pages 347–361. Springer Berlin Heidelberg, mcgeoch, catherine edition, 2008. 32

Bibliographie

- [33] J. Divoky and M. Hung. Performance of shortest paths algorithms in networks flow problems. *Management Science*, 36 (6) :661–673, 1990. 18
- [34] M. Ehrgott and X. Gandibleux. Multiple criteria optimization - state of the art annotated bibliographic surveys. *International series in Operations Research and Management Science*, Kluwer Academic Publishers, Boston, 52, 2002. 27
- [35] D. Febraro and S. Sacone. An on-line information system to balance traffic flows urban areas. In *In Proceeding of the 36th IEEE conference on decision and control*, pages 4772–4773, 1995. 35
- [36] R. Floyd. Algorithm 97 : Shortest path. *Communications of the ACM*, 5 :345, 1962. 25
- [37] L. Ford. Network flow theory. *Rand corporation report*, page 293, 1956. 18
- [38] L. Fu, D. Sun, and L. Rilett. Heuristic shortest path algorithms for transportation applications : State of art. *Computers and Operations Research*, 33 :3324–3343, 2006. 21
- [39] X. Gandibleux, F. Beugnies, and S. Randriamasy. Martins’algorithms revisited for multi-objective shortest path problem with a maxmin cost function. *4OR : A Quarterly Journal of Operational Research*, 4(1) :47–59, 2006. 32
- [40] GART. Groupement des autorités responsables de transport www.gart.org. 2
- [41] N. Giacomo, D. Daniel, L. Leo, and S. Dominik. Bidirectional a* search for time-dependant fast paths. In C. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 334–346. Springer Berlin / Heidelberg, 2008. 72
- [42] G. Giorgio and P. Stefano. Shortest path algorithms. *Annals of Operations Research*, 13 : 1–79, 1988. ISSN 0254-5330. 17
- [43] A. Goldberg, H. Kaplan, R. Werneck, and C. Harrelson. Efficient point to point shortest path. 2005. 7, 19, 20
- [44] M. Gondran and M. Minoux. *Graphes et Algorithmes*. Eyrolles, 1979. 18
- [45] T. Grabener, A. Berro, and Y. Duthen. Time dependent multiobjective best path for multimodal urban routing. In *ISCO*, 2010. 36

-
- [46] P. Hansen. Bicriterion path problems. In *In G. Fandel and T. Gal, editors, Multiple Criteria Decision Making Theory and Applications*, volume 177, pages 109–127. Springer Verlag, Berlin, 1979. of Lecture Notes in Economics and Mathematical Systems. 31
- [47] C. Harrelson and A. Goldberg. Computing the shortest path : A* meets graph theory. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005), SIAM (2005)*, 2005. 24
- [48] E. Hart, N. Nilson, and B. Raphael. A formal basis for heuristic determination of minimum costs paths. *IEEE Transaction System Science and Cybernetics*, SSC-4(2) :100–107, 1968. 19
- [49] M. Holzer. *Engineering Planar Separator and Shortest-Path Algorithms*. PhD thesis, University Karlsruhe, 2008. 32
- [50] M. Holzer and M. Kutrib. Nondeterministic finite automata—recent results on the descriptive and computational complexity. In *CIAA '08 : Proceedings of the 13th international conference on Implementation and Applications of Automata*, pages 1–16, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-70843-8. 63
- [51] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing, Reading Massachusetts, 1979. 63
- [52] M. Hung and J. Divoky. A computational study of efficient shortest path algorithms. *Computers and Operations Research*, 15 (6) :567–576, 1988. 18
- [53] C. Hwang and A. Masud. *Multiple objective decision making - methods and applications*, volume 164. Springer Verlag, Berlin, 1979. 29
- [54] N. Jing, Y. Huang, and E. Rundensteine. Hierarchical encoded path views for path query processing : An optimal model and its performance evaluation. *IEEE Transactions of Knowledge and Data Engineering*, 10 (3) :409–432, 1998. 9
- [55] N. Jozefowicz. *Modélisation et résolution approchée des problèmes de tournées multi-objectif*. PhD thesis, Thèse de doctorat Université Lille 1, 2004. 28
- [56] P. Judea. *Heuristics : intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., 1984. ISBN 0-201-05594-5. 19

Bibliographie

- [57] P. Lacomme, C. Prins, and M. Sevaux. *Algorithmes de graphes*. Eyrolles 2^e edition, 2003. 18, 20
- [58] Y. Lao. *Multi-Modal Route Planning in Public Land Transportation*. PhD thesis, Bachelor Thesis, School of Computing, National University of Singapore, 1999. 35
- [59] U. Lauther. Slow preprocessing of graphs for extremely fast shortest path calculations. *Lecture at the Workshop on Computational Integer Programming at ZIB*, 1997. 24
- [60] U. Lauther. An extremely fast exact algorithm for finding shortest paths in static networks with geographical background. *IFGI prints*, 22 :219–230, 2004. 24
- [61] Q. Li and C. E. Kurt. Gis-based itinerary planing system for multimodal and fixed route network. In *In Proceedings of the MID-Continent Transportation Center of Kansas University*. 35
- [62] A. Lozano and G. Storchi. Shortest viable path algorithm in multimodal networks. *Transportation Research Part A : Policy and Practice*, 35(3) :225–241, 2001. 7, 12, 13, 33, 34, 35, 42, 45, 46, 52, 57, 91, 98, 99
- [63] A. Lozano and G. Storchi. Shortest viable hyperpath in multimodal networks. *Transportation Research Part B*, 36 :853–874, 2002. 14
- [64] M. Mainguenaud. Modeling the network component of geographical information system. *International Journal of Geographic Information Systems*, 9 (6) :575–593, 1995. 9
- [65] E. Martins. On amulticriteria shortest path algorithm. *European Journal of Operational Research*, 16 :236–245, 1984. 31, 32
- [66] E. Martins and J. Santos. The labelling algorithm for the multiobjective shortest path problem. Technical report, Universidade de Coimbra, Portugal, 1999. 31, 32
- [67] J. C. E. Martins. A bicriterion shortest path algorithm. *European Journal of research*, 11 : 399–404, 1982. 32
- [68] J. McHugh. *Algorithmic graph theory*. Prentice Hall, 1990. 18
- [69] C. Méditerranée. Déploiement national des systèmes d’information multimodale- delfi : l’exemple allemenad. Technical report, CERTU, 2000. Collection du CERTU. 3

-
- [70] C. Méditerranée. Déploiement national des systèmes d'information multimodale - delfi : l'exemple allemenad. Technical report, 2001. Collection du CERTU. 3
- [71] A. Mendelzon and P. Wood. Finding regular simple paths in graph databases. *SIAM Journal on Computer*, 24(6) :1235–1258, 1995. 32
- [72] F. H. Meng, L. Yizhi, and L. H. Wai. A multi-critéria,multi-modal passenger route advisory system. In *In Proceedings of IES-CTR international symposium, Singapore*, 1999. 35
- [73] Meunier. *Algorithmes évolutionnaires parallèles pour l'optimisation multi-objectif des réseaux de télécommunications mobiles*. PhD thesis, Thèse de doctorat Université des sciences et technologies de Lille, 2002. 30
- [74] B. S. M.G. Battista, M. Lucertini. Path composition and multiple choice in a bimodal transportation network. In *Proceedings of the Seventh WCTR, Sydney*, 1995. 32
- [75] M. Müller-Hannemann and M. Schnee. *Finding All Attractive Train Connections by Multi-criteria Pareto Search*, volume 4359 of *Lecture Notes in Computer Science*, pages 246–263. Springer Berlin / Heidelberg, 2007. 32
- [76] E. Moore. The shortest path trough a lmaze. In *In H. U. Press (Ed.) Proceedings of the international Symposium on the theory of switching*, pages 285–293, 1959. 18
- [77] G. Nannicini, P. Baptiste, D. Kroh, and L. Liberti. Fast paths on dynamic road networks. In *ROADEF 08*, pages 285–293, 2008. 26
- [78] R. V. Nes. Hierarchical levels networks in the design of multimodal transport networks. In *Nectar Conference, Delft*, October 1999. 9
- [79] J. Nicholson. Finding the shortest route between two points in a network. *Computer Journal*, 9 :275–280, 1966. 21, 22, 64
- [80] J. Nilsson. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill Pub. Co., 1971. ISBN 0070465738. 19
- [81] M. Nonato, S. Pallotino, and B. Xuwen. Splt shortest path algorithms : review, new proposals and some experimental results. Technical Report TR-99-16, 1999. 18
- [82] T. Pajor. *Multi-Modal Route Planning*. PhD thesis, PhD dissertation, Universitat Karlsruhe (Germany), 2009. 33

Bibliographie

- [83] S. Pallotino and M. Scutella. Shortest path algorithms in transportation models : Classical and innovative aspects. *Equilibrium and Advanced Transportation Modeling*, pages 245–281, 1998. 11, 12, 25, 35, 52, 57
- [84] I. Pohl. Bi-directional search. *Machine intelligence*, 6 :127–140, 1971. 19
- [85] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis. Fast shortest path distance estimation in large networks. In *CIKM 09 : Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 867–876, 2009. 24
- [86] P. Sanders and D. Schultes. Engineering highway hierarchies. In *Proceedings of the 14th conference on Annual European Symposium*, volume 4168 of *Lecture Notes In Computer Science*, pages 804–816, 2006. 24
- [87] G. Sauvanet and E. Néron. Search for the best compromise solution on multiobjective shortest path problem. *Electronic Notes in Discrete Mathematics*, 36 :615 – 622, 2010. ISCO 2010 - International Symposium on Combinatorial Optimization. 32
- [88] T. Schulz. *Timetable Information and shortest paths*. PhD thesis, These de doctorat, Université Fridericiana zu Karlsruhe Allemagne, 2005. 10, 12
- [89] Serafini86. Fast paths on dynamic road networks. In *ROADEF 08*, pages 285–293, 2008. 31
- [90] H. Sherali and C. Jeenanunta. The approach dependant, time-dependant, label-constrained shortest path problem. *Networks*, 48(2) :57–67, 2006. 32, 33
- [91] H. Sherali, A. Hobeika, and S. Kangwalklai. The approach dependant, time-dependant, label-constrained shortest path problems with applications. *Transportation Science*, 37(3) : 278–293, 2003. 32, 33
- [92] A. Skriver and K. Andersen. A label correcting approach for solving bicriterion shortest path problems. *Computers and Operations Research*, 27(6) :507–524, 2000. 31
- [93] I. Stefan and D. Guy. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors, *Column Generation*, pages 33–65. Springer US, 2005. ISBN 978-0-387-25486-9. 32
- [94] Z. Tarapata. Selected multicriteria shortest path problems : An analysis of complexity, models and adaptation of standard algorithms. *International Journal Of Applied Mathematics and Computer Science*, 17 :269–287, 2007. 32

- [95] Q. Wu and J. Hartley. Using k-shortest paths algorithms to accommodate user preferences in the optimization of public transport travel. In *Proceeding of UKSIM 2004*, pages 113–117, 2004. 36

Bibliographie

Annexe A

A Réseaux de transports

A.1 Réseau de transport non dépendant du temps

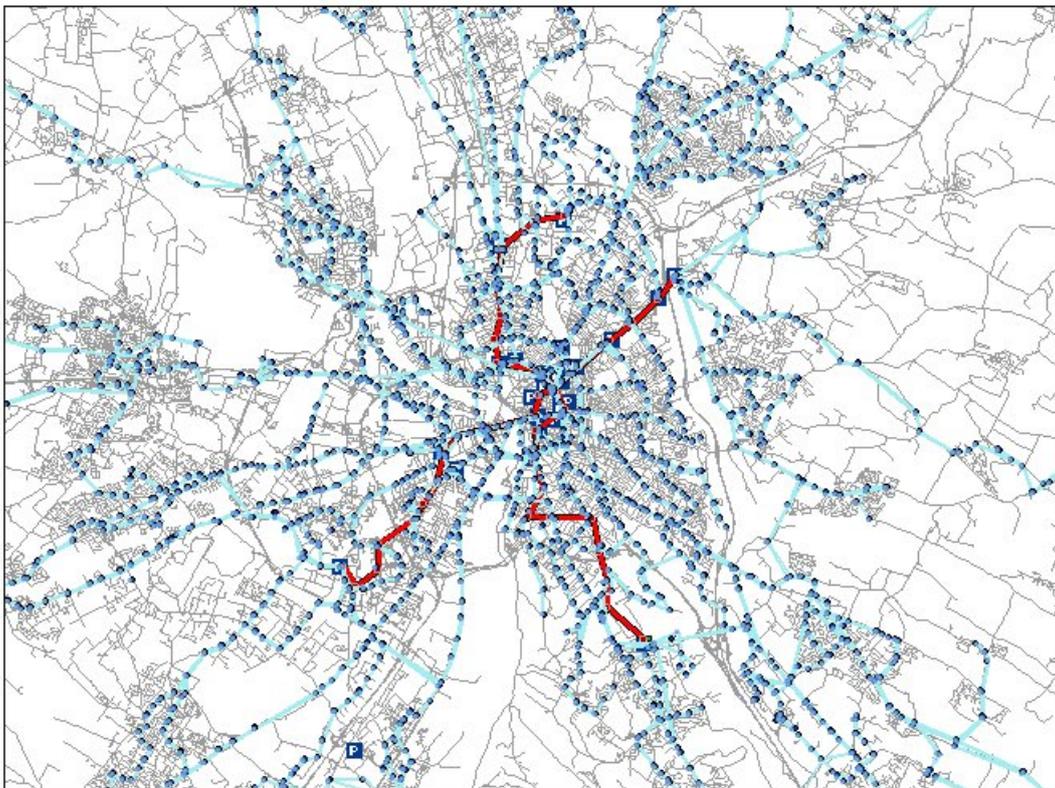


Figure A.1 – Réseau de transport multimodal avec vitesse moyenne

A.2 Réseau de transport dépendant du temps

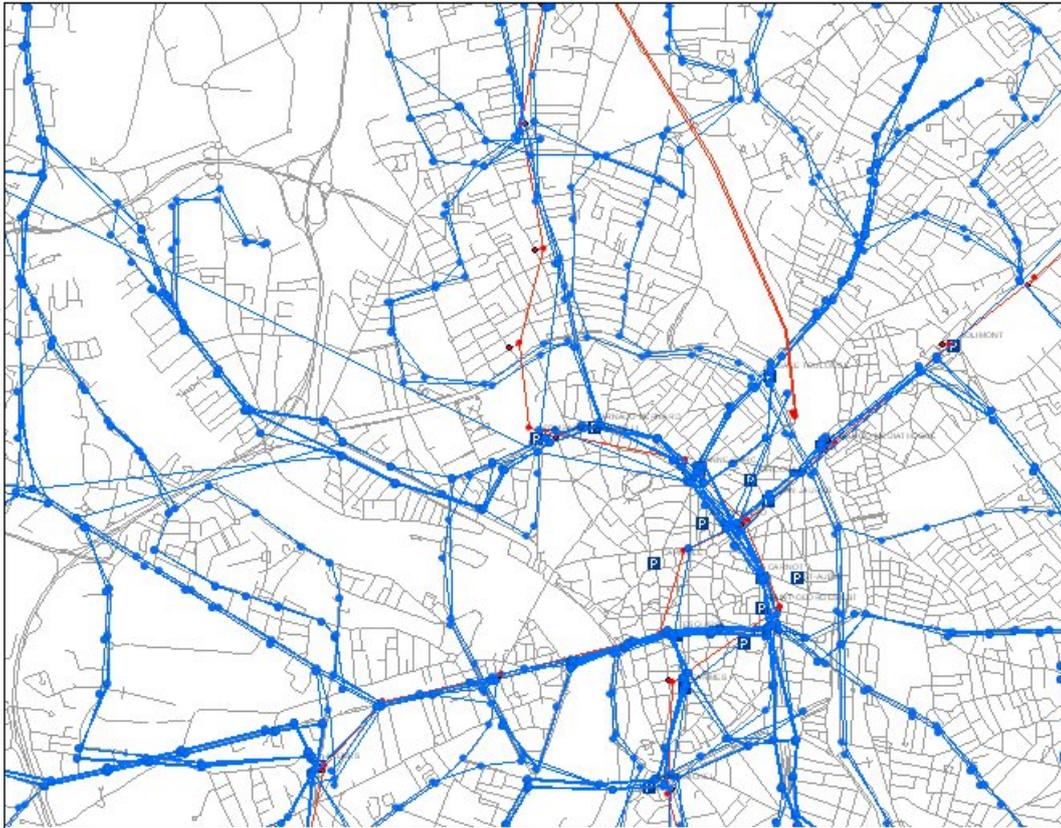


Figure A.2 – Réseau de transport multimodal avec horaires

Annexe B

B Résultats des expérimentations sur le réseau non dépendant du temps

B.1 Résultats des performances des algorithmes

B.1.1 Résultats DIJ-MM

		Min. Moy. Max.		
DIJ-MM Sans dominance	Temps Cpu (ms)	3770	4582	5200
	# labels dépilés	1203160	1468170	1657720
	# labels empilés	1319950	1607760	1812640
	# labels visités	3119950	3783750	4258770
		Min. Moy. Max.		
DIJ-MM Dominance labels	Temps Cpu (ms)	3445	4112	4553
	# labels dépilés	1114820	1325430	1456060
	# labels empilés	1221740	1450250	1590610
	# labels visités	2885580	3412480	3737110
		Min. Moy. Max.		
DIJ-MM Dominance états	Temps Cpu (ms)	3770	4582	5200
	# labels dépilés	1203160	1468170	1657720
	# labels empilés	1319950	1607760	1812640
	# labels visités	3119950	3783750	4258770

Tableau A.1 – Résultats de l’algorithme DIJ-MM

B.1.2 Résultats A*-DIJ-MM

		Min. Moy. Max.		
A*-DIJ-MM Sans dominance	Temps Cpu (ms)	3534	4353	4958
	# labels dépilés	1155690	1411690	1610750
	# labels empilés	1273560	1550930	1767060
	# labels visités	2993770	3636570	4137200
		Min. Moy. Max.		
A*-DIJ-MM Dominance labels	Temps Cpu (ms)	3427	4131	4679
	# labels dépilés	1073360	1281550	1422910
	# labels empilés	1181750	1407030	1560370
	# labels visités	2775750	3298560	3654000
		Min. Moy. Max.		
A*-DIJ-MM Dominance états	Temps Cpu (ms)	3378	4030	4465
	# labels dépilés	1073360	1281550	1422910
	# labels empilés	1181750	1407030	1560370
	# labels visités	2775750	3298560	3654000

Tableau A.2 – Résultats de l'algorithme A*-DIJ-MM

B.1.3 Résultats TLS

		Min. Moy. Max.		
TLS Sans dominance	Temps Cpu (ms)	2291	2603	2844
	# labels dépilés	494468	555900	590102
	# labels empilés	564339	629159	666843
	# labels visités	1281980	1428760	1509450
		Min. Moy. Max.		
TLS Dominance labels	Temps Cpu (ms)	1468	1654	1772
	# labels dépilés	339402	376339	400805
	# labels empilés	381582	420374	445843
	# labels visités	872522	960483	1018690
		Min. Moy. Max.		
TLS Dominance états	Temps Cpu (ms)	1473	1587	1693
	# labels dépilés	329407	357838	377671
	# labels empilés	369683	399459	421311
	# labels visités	846611	912819	959633

Tableau A.3 – Résultats de l'algorithme TLS

B Résultats des expérimentations sur le réseau non dépendant du temps

B.1.4 Résultats A*-TLS

		Min. Moy. Max.		
A*-TLS Sans dominance	Temps Cpu (ms)	2201	2496	2693
	# labels dépilés	474113	541619	576022
	# labels empilés	543027	614485	652399
	# labels visités	1232140	1394650	1475860
		Min. Moy. Max.		
A*-TLS Dominance labels	Temps Cpu (ms)	1459	1659	1851
	# labels dépilés	331121	369146	393292
	# labels empilés	373497	413389	438514
	# labels visités	852888	943528	1001100
		Min. Moy. Max.		
A*-TLS Dominance états	Temps Cpu (ms)	1418	1559	1651
	# labels dépilés	321425	351998	371860
	# labels empilés	361966	393947	415893
	# labels visités	827868	899175	946067

Tableau A.4 – Résultats

B.1.5 Résultats MLMH

		Min. Moy. Max.		
MLMH Sans dominance	Temps Cpu (ms)	2309	2644	2832
	# labels dépilés	494467	555899	590101
	# labels empilés	544899	610193	646012
	# labels visités	1281980	1428760	1509450
		Min. Moy. Max.		
MLMH Dominance labels	Temps Cpu (ms)	1504	1703	1850
	# labels dépilés	340682	377119	401446
	# labels empilés	373116	411903	437883
	# labels visités	875928	962706	1020620
		Min. Moy. Max.		
MLMH Dominance états	Temps Cpu (ms)	1333	1525	1673
	# labels dépilés	305556	337687	360240
	# labels empilés	333661	368021	392081
	# labels visités	778996	857944	911957

Tableau A.5 – Résultats de l'algorithme MLMH

B.1.6 Résultats A*-MLMH

		Min. Moy. Max.		
A*-MLMH Sans dominance	Temps Cpu (ms)	2198	2560	2753
	# labels dépilés	474112	541618	576021
	# labels empilés	524209	595728	631744
	# labels visités	1232140	1394650	1475860
		Min. Moy. Max.		
A*-MLMH Dominance labels	Temps Cpu (ms)	1466	1693	1837
	# labels dépilés	332418	370041	394187
	# labels empilés	365037	405046	430792
	# labels visités	856361	946037	1003630
		Min. Moy. Max.		
A*-MLMH Dominance états	Temps Cpu (ms)	1317	1494	1619
	# labels dépilés	299108	332027	3551520
	# labels empilés	327365	362668	387300
	# labels visités	763338	844700	900229

Tableau A.6 – Résultats de l'algorithme A*-MLMH

B.1.7 Résultats FB-MLMH

		Min. Moy. Max.		
MLMH Sans dominance	Temps Cpu (ms)	825	2245	3307
	# labels dépilés	287670	386391	443082
	# labels empilés	326607	430994	491865
	# labels visités	769016	1016180	1160870
		Min. Moy. Max.		
FB-MLMH Dominance labels	Temps Cpu (ms)	1270	1600	1981
	# labels dépilés	222319	283721	330295
	# labels empilés	249441	313798	364873
	# labels visités	586478	738354	856297
		Min. Moy. Max.		
FB-MLMH Dominance états	Temps Cpu (ms)	638	1396	1967
	# labels dépilés	202346	255464	296962
	# labels empilés	226331	281770	327417
	# labels visités	530163	660971	766086

Tableau A.7 – Résultats de l'algorithme FB-MLMH

B Résultats des expérimentations sur le réseau non dépendant du temps

B.1.8 Résultats A*-FB-MLMH

		Min. Moy. Max.		
A*-FB-MLMH Sans dominance	Temps Cpu (ms)	1257	1944	2373
	# labels dépilés	221959	338490	402644
	# labels empilés	253298	379994	449687
	# labels visités	594812	894320	1061010
		Min. Moy. Max.		
A*-FB-MLMH Dominance labels	Temps Cpu (ms)	1034	1455	1788
	# labels dépilés	183643	259671	308719
	# labels empilés	206990	288897	343044
	# labels visités	485185	678385	803849
		Min. Moy. Max.		
A*-FB-MLMH Dominance états	Temps Cpu (ms)	955	1337	1741
	# labels dépilés	168186	235079	278728
	# labels empilés	188786	260762	309091
	# labels visités	440738	610145	721725

Tableau A.8 – Résultats de l’algorithme A*-FB-MLMH

B.1.9 Résultats FB-MLMH-ND

		Min. Moy. Max.		
FB-MLMH-ND Sans dominance	Temps Cpu (ms)	1766	2273	2869
	# labels dépilés	312319	409516	494445
	# labels empilés	354294	456657	550987
	# labels visités	834428	1076450	1295760
		Min. Moy. Max.		
FB-MLMH-ND Dominance labels	Temps Cpu (ms)	1371	1654	2077
	# labels dépilés	248635	306942	382572
	# labels empilés	278843	339536	423689
	# labels visités	656112	798842	994018
		Min. Moy. Max.		
FB-MLMH-ND Dominance états	Temps Cpu (ms)	1302	1559	2017
	# labels dépilés	228399	278561	349239
	# labels empilés	255486	307381	386234
	# labels visités	599156	721159	903807

Tableau A.9 – Résultats de l’algorithme FB-MLMH-ND

B.1.10 Résultats A^* -FB-MLMH

		Min. Moy. Max.		
A^* -FB-MLMH-ND Sans dominance	Temps Cpu (ms)	1374	2026	2494
	# labels dépilés	249272	366260	444716
	# labels empilés	287359	415358	502981
	# labels visités	666728	966229	1169560
		Min. Moy. Max.		
A^* -FB-MLMH-ND Dominance labels	Temps Cpu (ms)	1087	1492	2006
	# labels dépilés	203335	276089	349344
	# labels empilés	232534	310959	394578
	# labels visités	535866	719805	910106
		Min. Moy. Max.		
A^* -FB-MLMH-ND Dominance états	Temps Cpu (ms)	954	1326	1736
	# labels dépilés	188537	251082	319813
	# labels empilés	215338	282515	360362
	# labels visités	493326	650729	829400

Tableau A.10 – Résultats de l'algorithme A^* -FB-MLMH-ND

Annexe C

C Schéma TRIDENT UML

Un itinéraire se compose de missions ; une mission est associée à un itinéraire. La relation entre eux est 1 à n. Un itinéraire ne se compose pas de points d'arrêt directement, mais de tronçons ; un tronçon peut être partagé par différents itinéraires. La relation entre eux est n à n.

- Une mission contient des courses, une course est associée à une mission unique. La relation entre eux est 1 à n. elle se compose de points d'arrêts, et un point d'arrêt peut être partagé par différentes missions. La relation entre eux est n à n.
- Une course comprend des points d'arrêt et des horaires comprenant les informations suivantes : l.heure d'arrive, l.heure de départ, la durée d'attente, le type de jour, la période de validité de ces informations, etc.
- Un point d'arrêt peut être contenu dans différentes zones d'arrêt ; chaque zone d'arrêt peut contenir plusieurs points d'arrêt (un point physique et un point logique)
- Un tronçon de correspondance relie points d'arrêt ou zones d'arrêt. Pour en savoir plus sur ses caractéristiques, voir l'entité de Correspondance dans la partie de Transmodel
- Une horaire contient des courses.

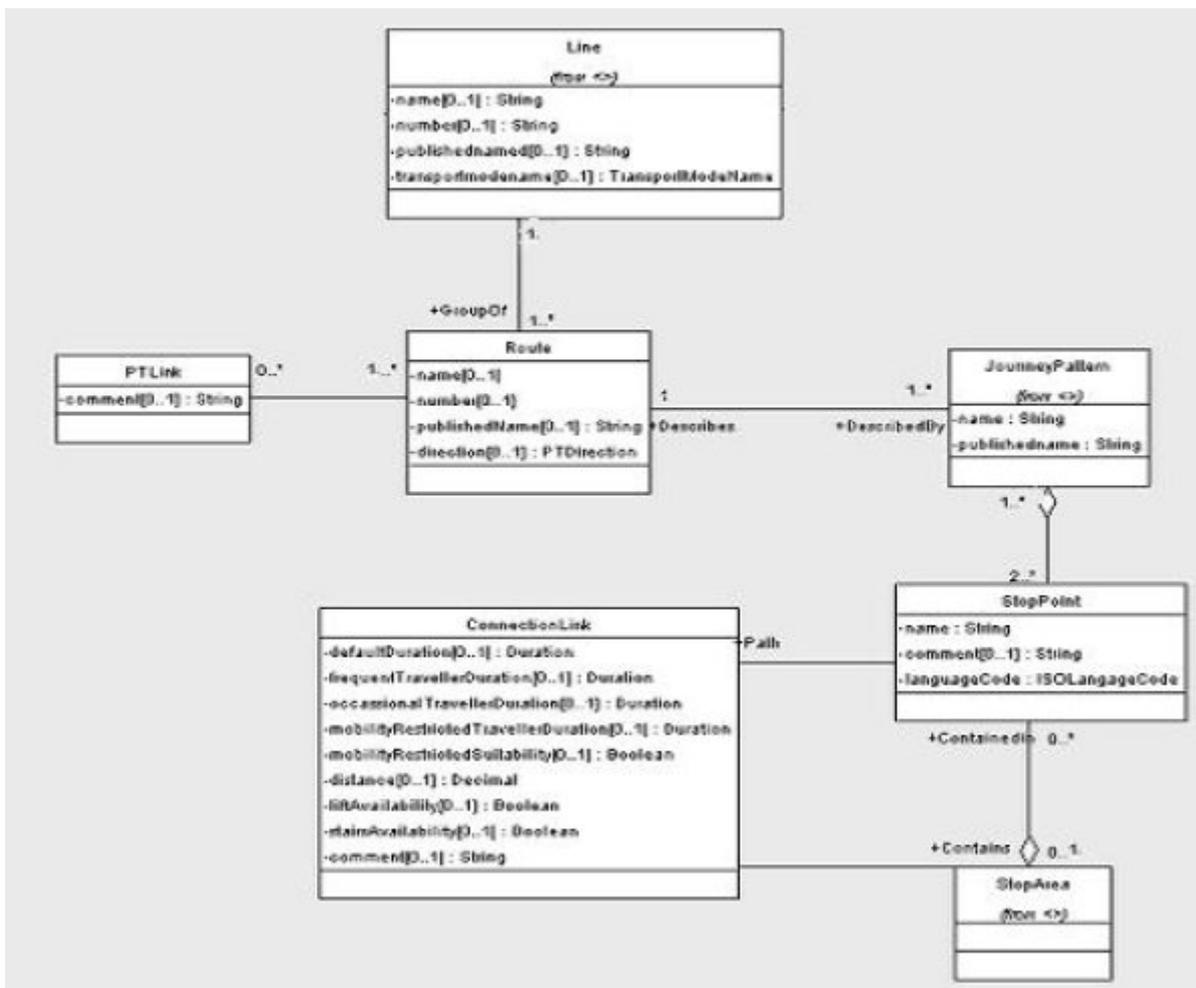


Figure A.3 – Schéma UML TRIDENT (line type)