



Recherche de chemins dans un graphe à pondération dynamique : application à l'optimisation d'itinéraires dans les réseaux routiers

Mohamed Mejdi Hizem

► To cite this version:

Mohamed Mejdi Hizem. Recherche de chemins dans un graphe à pondération dynamique : application à l'optimisation d'itinéraires dans les réseaux routiers. Automatique / Robotique. Ecole Centrale de Lille, 2008. Français. NNT: . tel-00344958

HAL Id: tel-00344958

<https://theses.hal.science/tel-00344958>

Submitted on 7 Dec 2008

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

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

N° d'ordre :

0	8	6
---	---	---

ÉCOLE CENTRALE DE LILLE

THÈSE

présentée en vue d'obtenir le grade de

DOCTEUR

en

Spécialité : Automatique et Informatique Industrielle

par

Mohamed Mejdi HIZEM

Doctorat délivré par l'École Centrale de Lille

Recherche de chemins dans un graphe à pondération dynamique

Application à l'optimisation d'itinéraires
dans les réseaux routiers

Soutenue le 29 novembre 2008 devant le jury d'examen :

Président	Jean-François PETIN	Maître de Conférences HDR à l'ESIAL à Vandoeuvre-lès-Nancy
Rapporteur	Alexandre DOLGUI	Professeur à l'École Nationale des Mines de Saint-Etienne
Rapporteur	Ouajdi KORBAA	Professeur à l'Université de Sousse (Tunisie)
Membre	Besoa RABENASOLO	Professeur à l'ENSAIT à Roubaix
Directeur de thèse	Armand TOGUYENI	Professeur à l'École Centrale de Lille
Co-directeur de thèse	Emmanuel CASTELAIN	Maître de Conférences HDR à l'École Centrale de Lille

Thèse préparée au sein des Laboratoires LAGIS et LGIL

École Doctorale SPI 072

À la mémoire de mon père

À ma mère

À mon frère Béchir

À mes sœurs Wafa et Wided

Remerciements

En premier lieu, je tiens à exprimer ma profonde gratitude aux directeurs de cette thèse le Professeur Armand Toguyéni et à Emmanuel Castelain, Maître de Conférences - HDR. Je veux les remercier pour leur encadrement plein d'enthousiasme et de rigueur et pour la confiance dont ils ont fait preuve à mon égard. De plus, je tiens à leur manifester ma sincère reconnaissance pour leurs grandes qualités humaines qui ont fait que cette thèse se passe dans la bonne humeur.

Ensuite, je souhaite témoigner de mes remerciements chaleureux à Jean-François Petin, Maître de Conférences - HDR, de m'avoir accordé l'honneur de présider le jury d'examen.

Puis, je tiens à exprimer ma vive gratitude au Professeur Alexandre Dolgui et au Professeur Ouajdi Korbaa pour l'intérêt dont ils ont fait preuve à l'égard de mon travail en acceptant d'être les rapporteurs de cette thèse.

Je désire également remercier le Professeur Besoa Rabenasolo d'avoir accepté d'examiner ce travail de recherche et de faire partie du jury.

Finalement, ces remerciements ne seraient pas complets si je n'y associais

pas toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce travail, en particulier, mes collègues, tout le personnel du LAGIS, du LGIL et de l'École Centrale de Lille pour leur bonne humeur et leur disponibilité.

Table des matières

Table des figures	14
Liste des tableaux	15
Intoduction générale	16
1 État de l’art	20
1.1 Introduction	20
1.2 Les problèmes d’optimisation	21
1.2.1 Définitions générales	21
1.2.2 Optimisation monocritère	23
1.2.2.1 Les méthodes déterministes	24
1.2.2.2 Les méthodes stochastiques	25
1.2.3 Optimisation multicritère	27
1.2.3.1 Optimisation multicritère par agrégation de critères	27
1.2.3.2 Optimisation multicritère par Pareto domi- nance	31
1.3 Graphes et problèmes du plus court chemin	33
1.3.1 Définitions générales	33
1.3.2 Graphes statiques déterministes	35

1.3.2.1	Définition	35
1.3.2.2	Problème classique du plus court chemin . . .	35
1.3.3	Graphes statiques stochastiques	37
1.3.3.1	Définition	37
1.3.3.2	Problème du plus court chemin stochastique .	39
1.3.4	Graphes dynamiques déterministes	40
1.3.4.1	Définition	40
1.3.4.2	Graphes FIFO	40
1.3.4.3	Problème du plus court chemin dynamique . .	41
1.3.5	Graphes dynamiques stochastiques	43
1.3.5.1	Définition	43
1.3.5.2	Problème du plus court chemin dynamique stochastique	47
1.3.6	Graphes statiques avec intervalles	48
1.3.6.1	Définition	48
1.3.6.2	Problème du plus court chemin de déviation robuste	49
1.4	Critères de prise de décision dans un environnement incertain	51
1.4.1	Critères quantitatifs	52
1.4.1.1	Critère de Laplace	52
1.4.1.2	Critère de Wald ou du MaxiMin	52
1.4.1.3	Critère du MaxiMax	53
1.4.1.4	Critère d'Hurwitz	53
1.4.1.5	Critère de Savage ou du MiniMax	54
1.4.2	Critères probabilistes	54
1.4.2.1	Critère de Pascal	54
1.4.2.2	Critère de Markowitz	54

1.5	Méthodes d'interception d'un mobile	55
1.5.1	Interception dans un espace continu	55
1.5.2	Interception dans les graphes	56
1.5.2.1	Le problème de Poursuite-Évasion	56
1.5.2.2	Le problème du Rendez-vous	57
1.6	Conclusion	60
2	Interception d'un mobile dans un graphe	61
2.1	Introduction	61
2.2	Problème d'interception un-à-un	62
2.2.1	Cas des graphes statiques	62
2.2.1.1	Le modèle	62
2.2.1.2	Algorithme d'interception	67
2.2.1.3	Optimalité de l'algorithme	69
2.2.1.4	Complexité de l'algorithme	70
2.2.1.5	Implémentation de l'algorithme	71
2.2.1.6	Comparaison avec une approche par poursuite	73
2.2.2	Cas des graphes FIFO	77
2.2.2.1	Le modèle	77
2.2.2.2	Algorithme d'interception	78
2.2.2.3	Optimalité et complexité de l'algorithme . . .	80
2.2.2.4	Implémentation de l'algorithme	80
2.3	Problème d'interception plusieurs-à-plusieurs	81
2.3.1	Cas des graphes statiques	81
2.3.1.1	Le modèle	81
2.3.1.2	Algorithme d'affectation	84
2.3.1.3	Complexité de l'algorithme	86
2.3.1.4	Implémentation de l'algorithme	88

2.3.2	Cas des graphes FIFO	89
2.4	Conclusion	90

3 Problème du plus court chemin dans un graphe dynamique avec intervalles 92

3.1	Introduction	92
3.2	Graphe dynamique avec intervalles	93
3.2.1	Définition générale	93
3.2.2	Graphe FIFO avec intervalles	94
3.3	Problème du plus court chemin	95
3.3.1	Problème du plus court chemin de déviation robuste	95
3.3.2	Problème du plus court chemin suivant un critère d'optimisme	97
3.3.3	Résolution du problème du plus court chemin suivant un critère d'optimisme	99
3.3.3.1	Facteur d'optimisme	99
3.3.3.2	Calcul du plus court chemin	103
3.4	Calcul du plus court chemin à l'aide d'un facteur d'optimisme dynamique	103
3.4.1	Le paramètre α_{profil}	105
3.4.1.1	Définition	105
3.4.1.2	Exemple	105
3.4.2	Le paramètre β	106
3.4.2.1	Définition	106
3.4.2.2	Exemple	106
3.4.3	Le paramètre γ	111
3.4.3.1	Définition	111
3.4.3.2	Exemple	111

3.4.4	Le paramètre δ	112
3.4.4.1	Définition	112
3.4.4.2	Exemple	113
3.4.5	Agrégation des paramètres	113
3.4.5.1	Agrégation par somme pondérée	113
3.4.5.2	Agrégation par l'intégrale de Choquet	114
3.5	Problème du plus court chemin suivant un critère d'optimisme et un critère de déviation robuste	115
3.5.1	Définition du problème	115
3.5.2	Les variables du problème	115
3.5.2.1	Le coût du chemin suivant un critère d'opti- misme	115
3.5.2.2	La déviation robuste maximale	116
3.5.3	Résolution	118
3.5.3.1	Résolution par agrégation de critères	118
3.5.3.2	Résolution par Pareto dominance	120
3.6	Synthèse	121
3.7	Conclusion	121
Conclusion et perspectives		124
Bibliographie		140
A Étude d'un exemple d'optimisation multicritère par l'inté- grale de Choquet		141
B Étude d'un exemple d'utilisation des critères de décision dans un environnement incertain		146
B.1	Critères quantitatifs	147

B.1.1	Critère de Laplace	147
B.1.2	Critère de Wald ou du MaxiMin	147
B.1.3	Critère du MaxiMax	148
B.1.4	Critère d'Hurwitz	148
B.1.5	Critère de Savage ou du MiniMax	148
B.2	Critères probabilistes	149
B.2.1	Critère de Pascal	150
B.2.2	Critère de Markowitz	150
C	Algorithme d'interception de référence	152

Table des figures

1.1	Exemple de dominance et d'optimalité au sens de Pareto . . .	32
1.2	Exemple d'un graphe orienté	34
1.3	Exemple d'un graphe non orienté	34
1.4	Exemple d'un graphe statique déterministe	36
1.5	Algorithme de Dijkstra	38
1.6	Exemple d'un graphe statique stochastique	39
1.7	Exemple d'un graphe dynamique déterministe	41
1.8	Version dynamique de l'algorithme de Dijkstra	44
1.9	Exemple d'une bucket-list	45
1.10	Itération typique du paradigme Chrono-SPT	45
1.11	Exemple d'un graphe dynamique stochastique	46
1.12	Exemple d'un graphe statique avec intervalles	48
1.13	Exemple d'un scénario	49
2.1	Exemple de graphes G_o et G_p	64
2.2	Exemple de la duplication d'un nœud	65
2.3	Exemple de graphes G_o et G_p avec un itinéraire de MO	66
2.4	Algorithme d'interception un-à-un pour les graphes statiques .	68
2.5	Logiciel de simulation	74

2.6	Évolution du temps d'exécution moyen de l'algorithme d'interception et de l'algorithme de référence en fonction du nombre de nœuds	75
2.7	Algorithme de poursuite	76
2.8	Algorithme d'interception un-à-un pour les graphes FIFO . . .	79
2.9	Évolution du temps d'exécution moyen de l'algorithme d'interception en fonction du nombre de nœuds pour les graphes FIFO et les graphes statiques	82
2.10	Exemple avec plusieurs mobiles objectifs et plusieurs mobiles poursuivants	83
2.11	Premier exemple d'une matrice de coût où $K = L$	85
2.12	Deuxième exemple d'une matrice de coût où $K = L$	86
2.13	Exemple d'une matrice de coût avec un MO virtuel	87
2.14	Exemple d'une matrice de coût avec un MP virtuel	87
2.15	Évolution du temps d'exécution moyen en fonction du nombre de MP/MO	90
3.1	Exemple d'un graphe dynamique avec intervalles	94
3.2	Application d'un facteur d'optimisme de 0.25 sur un graphe dynamique avec intervalles	102
3.3	Exemple schématique de la notion d'horizon	107
3.4	Exemple d'un graphe dynamique avec une pondération sous forme d'intervalles	108
3.5	Graphe dynamique avec des horizons spécifiés	110
3.6	Exemple de calcul de la durée approximative pour atteindre un nœud	110
3.7	Graphe du contre-exemple	119

3.8	Poids du graphe du contre-exemple pour le scénario de la dé- viation robuste maximale	119
3.9	Poids du graphe du contre-exemple pour le scénario de la dé- viation robuste maximale avec modification d'un seul poids . .	120
3.10	Synthèse des visions et des algorithmes proposés	122
C.1	Algorithme d'interception de référence	153

Liste des tableaux

1.1	Description des états de l'exemple	29
2.1	Temps d'exécution moyen en fonction du nombre de nœuds de l'algorithme d'interception et de l'algorithme de référence . . .	74
2.2	Comparaison entre l'approche par interception et l'approche par poursuite	77
2.3	Temps d'exécution moyen de l'algorithme d'interception en fonction du nombre de nœuds pour les graphes FIFO et les graphes statiques	81
2.4	Temps d'exécution moyen en fonction du nombre de MP/MO	89
A.1	Scores des entreprises et résultat de l'agrégation par somme pondérée	142
A.2	Scores des entreprises et résultat de l'agrégation par somme pondérée et par l'intégrale de Choquet	145
B.1	Résultats des actions suivant les états	147
B.2	Matrice des regrets	149

Introduction générale

Le besoin de se déplacer est devenu un besoin fondamental pour l'homme moderne. Ce besoin est aussi bien important pour la vie sociale de l'être humain que pour sa vie professionnelle. Avec l'extension continue des zones urbaines, l'augmentation de la population et l'amélioration du niveau de vie des citoyens, le nombre de voitures ne cesse d'augmenter. Dans l'Union européenne, le nombre de voitures a augmenté de 38% entre 1990 et 2004 pour atteindre en 2004 le chiffre de 472 voitures pour 1000 habitants¹. En France, cette augmentation a été de 18,6% pour atteindre en 2004 le chiffre de 491 voitures pour 1000 habitants¹. Ce grand nombre de véhicules, qui est en augmentation continue, provoque de nombreux problèmes tant environnementaux qu'économiques. En effet, avec ce grand nombre de voitures sur les réseaux routiers, de nombreux phénomènes de congestions routières sont observés quotidiennement. Ces embouteillages induisent une surconsommation de carburant, des émissions inutiles de gaz à effets de serre et une perte de temps importante. Ces pertes d'énergie et de temps infligent des coûts importants à la société. Une étude a été menée en 2002 par le gouvernement canadien sur le coût de la congestion urbaine des neuf plus grandes zones urbaines du Canada[26]. Cette étude révèle que le coût de la congestion récurrente dans les zones urbaines pour les canadiens est compris entre

1. Source EUROSTAT

2.3 et 3.7 milliards de dollars annuellement. Ce coût est réparti de la manière suivante. D'abord, 90% du coût est attribué au temps perdu par les conducteurs et les passagers dans la circulation. Ensuite, 7% du coût est induit par l'augmentation de la consommation de carburant. Finalement, 3% du coût représente l'augmentation des émissions de gaz à effet de serre. Par conséquent, l'optimisation du transport urbain s'impose comme un sujet primordial à traiter.

Le transport urbain est un sujet très complexe à traiter et il possède différentes facettes. Une de ces facettes est le choix du mode de transport. Effectivement, dans les zones urbaines, les usagers ont le choix entre différents moyens de transport pour accomplir leur itinéraire : voiture personnelle, métro, bus, train, ... Souvent, il est plus efficace, en termes de temps de trajet ou de coût financier, de choisir une combinaison de ces moyens de transport que de choisir un seul mode. Dans le Laboratoire d'Automatique, de Génie Informatique et de Signal différents travaux ont été réalisés afin de fournir des outils aidant les usagers à choisir leur itinéraire avec la meilleure combinaison possible de moyens de transport [132] [76] [131]. Une autre facette de la problématique de l'optimisation du transport urbain concerne le choix de l'itinéraire. En effet, pour un automobiliste ou un chauffeur de camion, un choix judicieux de l'itinéraire peut représenter une économie financière et un gain de temps importants. C'est dans ce contexte que s'inscrit ce travail. Plusieurs chercheurs se sont penchés sur la problématique du calcul d'un itinéraire optimal entre deux points d'un réseau routier depuis plusieurs années. Néanmoins ce sujet est toujours un thème de recherche d'actualité. Effectivement, le réseau routier est un système dynamique et très complexe. La dynamique du système se localise à différents niveaux. Par exemple, l'état du trafic sur une route donnée varie au cours du temps. Ou encore, cer-

tains événements imprévus (accidents, ...) ou prévus (travaux, événements sportifs, ...) peuvent influencer énormément sur l'état du trafic. De plus, les automobilistes ont un impact sur l'état du réseau. En effet, leur choix individuel d'itinéraire provoque des encombrements plus ou moins importants sur les routes. Ces paramètres et beaucoup d'autres influent sur l'état du système et, par conséquent, sur l'itinéraire optimal entre deux points du réseau. L'objectif de cette thèse est de déterminer un itinéraire optimal dans ce système en considérant deux aspects dynamiques différents. Premièrement, nous nous intéressons au calcul d'un itinéraire optimal pour rejoindre une destination mobile dans le réseau. Deuxièmement, nous développons un modèle de réseaux routiers intégrant simultanément la dynamique du réseau routier et l'incertitude sur son état et, par la suite, nous étudions le problème du calcul d'un itinéraire optimal pour ce modèle. La première problématique est d'un intérêt particulier pour le monde industriel. En effet, cette problématique a été soulevée par des entreprises de transport en commun qui sont confrontées quotidiennement à des situations nécessitant d'envoyer un véhicule rejoindre un bus en mouvement pendant son trajet. La deuxième problématique se concentre sur le développement d'une extension des modèles des réseaux routiers existants. Effectivement, les modèles actuels représentent soit le fait que l'état du réseau dépende du temps soit le fait qu'il existe une incertitude sur son état. Dans notre modèle, nous intégrons ces deux aspects pour fournir un modèle dont l'état n'est pas déterministe et qui dépend du temps.

Cette thèse est décomposée en trois chapitres. Le premier chapitre définit les différentes notions qui sont employées pour résoudre les problématiques traitées. D'abord, une présentation générale de l'optimisation est réalisée. Ensuite, nous indiquons les différents types de graphes présents dans la littérature. Puis, nous introduisons la théorie de la décision en nous intéressant

aux critères de décision dans un environnement incertain. Enfin, un état de l'art des méthodes d'interception est présenté. Le deuxième chapitre traite la première problématique qui est l'interception d'un mobile dans un graphe. Cette problématique est étudiée pour différents cas de figures et pour différents types de graphes. Le dernier chapitre s'intéresse au développement d'un modèle dynamique des réseaux routiers intégrant l'incertitude sur son état. Pour ce nouveau type de graphes, le problème du plus court chemin est étudié suivant différentes visions : l'optimisation monocritère et l'optimisation multicritère.

Chapitre 1

État de l’art

1.1 Introduction

Pour être résolus, certains problèmes nécessitent des approches multidisciplinaires. C’est dans cet esprit que s’inscrit notre démarche. Pour traiter les problèmes présentés dans les chapitres suivants, des notions issues de l’optimisation, de la théorie des graphes et de la théorie de la décision sont utilisées. L’objectif de ce chapitre est de présenter ces différentes notions. D’abord, nous nous intéressons à l’optimisation en définissant l’optimisation monocritère et l’optimisation multicritère. Pour chaque type d’optimisation, des méthodes de résolution sont présentées. Puis, nous définissons différentes classes de graphes. Pour chaque classe de graphe, le problème du plus court chemin est présenté. Ensuite, une introduction à la théorie de la décision est réalisée. Enfin, nous présentons une revue de la littérature des problèmes d’interception.

1.2 Les problèmes d'optimisation

Les problèmes d'optimisation sont des problèmes très utiles pour l'ensemble de la communauté scientifique. En effet, ils fournissent des techniques “génériques” qui peuvent être utilisées dans différents domaines. Ils peuvent être divisés en deux grandes catégories : l'optimisation monocritère (ou mono-objectif) et l'optimisation multicritère (ou multi-objectif). Pour l'optimisation monocritère, le but est de trouver la meilleure solution suivant une seule dimension ou aspect du problème (coût, temps, taux d'erreur, ...). Par contre, pour l'optimisation multicritère, il faut trouver la meilleure solution suivant un ensemble de dimensions ou aspects du problème (coût + temps, temps+taux d'erreur ...). Le but de cette section est d'introduire des notions et des notations, en particulier les méthodes d'optimisation multicritère, qui seront utilisées dans le chapitre 3. Dans la suite, un ensemble de définitions formelles pour les problèmes d'optimisation sont présentées.

1.2.1 Définitions générales

Définition 1 *Une variable du problème est une variable qui exprime une donnée quantitative ou qualitative sur une dimension du problème : coût, temps, taux d'erreurs, ...*

Définition 2 *Un vecteur de décision est un vecteur représentant un état donné sur toutes les dimensions du problème. Il est noté $x = (x_1, x_2, \dots, x_n)$ avec n le nombre de variables ou de dimensions du problème et x_k la variable sur la dimension k .*

Définition 3 *Une contrainte du problème est une condition que doivent respecter les vecteurs de décision du problème. Une contrainte est notée $G_k(x)$ avec $1 \leq k \leq m$ et m le nombre de contraintes.*

Définition 4 *Un critère de décision est un critère sur lequel sont jugés les vecteurs de décision pour déterminer le meilleur vecteur. Un critère peut être une variable du problème ou une combinaison de variables.*

Définition 5 *Une fonction objectif est une fonction qui modélise le but à atteindre dans le problème d'optimisation sur l'ensemble des critères. Il s'agit de la fonction qui doit être optimisée. Elle est noté $F(x)$. De manière générale, $F(x)$ est un vecteur $F(x) = (F_1(x), F_2(x), \dots, F_l(x))$.*

Définition 6 *Un problème d'optimisation multicritère consiste à trouver le vecteur de décision idéal x^* tel que les contraintes $G_k(x^*)$ soient satisfaites pour $1 \leq k \leq m$ et dont $F(x^*)$ est optimal.*

L'exemple illustratif suivant permet d'expliciter les différentes définitions présentées ci-dessus. Soit un automobiliste qui souhaite planifier un trajet entre deux villes éloignées. Pour chaque route, il dispose des informations suivantes : le temps nécessaire pour parcourir la route, le prix éventuel du péage et la longueur kilométrique de la route. Par suite, pour ce problème, il est possible de définir ce qui suit :

Les variables : le temps, le prix du péage et la distance ;

Un vecteur de décision : la durée du trajet, la somme des coûts des péages et la longueur totale d'un itinéraire formé par un ensemble quelconque de routes ;

Contraintes : deux contraintes doivent être imposées aux différents vecteurs de décisions. La première contrainte est que le point de départ de l'itinéraire est la ville de départ de l'automobiliste. La deuxième contrainte est que le point d'arrivée de l'itinéraire est la ville d'arrivée de l'automobiliste. D'autres contraintes du type temps maximal ou distance maximale à ne pas dépasser peuvent être ajoutées ;

Critères de décision : il est possible de choisir deux types de critères de décision. Premièrement, il est possible de choisir les variables du problème comme critères de décision. Dans ce cas, il y aura trois critères de décision : le temps, le prix du péage et la distance. Deuxièmement, il est possible de réaliser une combinaison des variables *prix du péage* et *distance*. En effet, supposons que la consommation du véhicule de l'automobiliste est connue. Il est possible alors de calculer le coût total du voyage en réalisant la somme du prix du péage et celui de l'essence consommé. Dans ce cas, il y aura deux critères de décision : le temps et le prix total du voyage ;

Fonction objectif : Soit $x = (x_1, x_2, x_3)$ un vecteur de décision. Dans le cas où les critères de décision sont le temps, le prix du péage et la distance, $F(x) = (F_1(x), F_2(x), F_3(x))$ avec $F_1(x) = x_1$, $F_2(x) = x_2$ et $F_3(x) = x_3$. Dans le cas où les critères sont le temps et le prix total du voyage $F(x) = (F_1(x), F_2(x))$ avec $F_1(x) = x_1$ et $F_2(x) = x_2 + x_3 \times cons$ en notant *cons* la consommation du véhicule de l'automobiliste.

1.2.2 Optimisation monocritère

Ce type d'optimisation est le premier à avoir été traité. Dans ce cadre, la fonction objectif est un 1-uplet ($F(x) = (F_1(x))$). Par suite, il est simple de comparer deux vecteurs de décision. Ce type d'optimisation a été largement étudié et plusieurs méthodes ont été proposées. Ces méthodes peuvent être divisées en deux types : les méthodes déterministes et les méthodes stochastiques [22].

1.2.2.1 Les méthodes déterministes

Ces méthodes se caractérisent par une exploration systématique de l'espace des vecteurs de décision. Deux exemples de méthodes déterministes largement utilisées sont l'algorithme du Simplexe et l'algorithme de Séparation-Évaluation. Dans la suite, ces deux méthodes sont introduites brièvement.

L'algorithme du Simplexe Cet algorithme est très utilisé dans le cadre de la programmation linéaire [36]. Son idée directrice est de partir d'un vecteur de décision quelconque. Puis, à partir de ce vecteur, rechercher un vecteur de décision adjacent qui est meilleur. S'il est trouvé, ce vecteur de décision deviendra le vecteur courant. S'il n'existe pas, le vecteur courant est alors le vecteur optimal.

L'algorithme de Séparation-Évaluation Cet algorithme a été initialement proposé par Land et Doig [84]. Il repose sur deux principes : la séparation et l'évaluation.

Le principe de séparation consiste à découper l'espace de recherche initial en domaines de plus en plus restreints afin d'isoler l'optimum global. Ce découpage est représenté sous la forme d'un arbre où chaque nœud représente une région de l'espace (un ensemble de vecteurs de décision). Dans cet arbre, les nœuds fils représentent un découpage de la région représentée par le nœud père. En effet, soient P un nœud de l'arbre et F_1, F_2, \dots, F_n ses nœuds fils. Les deux propriétés suivantes sont vérifiées :

- $F_i \subset P \ \forall i \in \{1, 2, \dots, n\}$;
- $\bigcup_{i=1}^n F_i = P$;
- $F_i \cap F_j = \emptyset \ \forall i, j \in \{1, 2, \dots, n\}$.

Le principe d'évaluation consiste à évaluer la borne inférieure (ou la borne supérieure) de la fonction objectif au niveau d'un nœud de l'arbre.

L'algorithme consiste à construire l'arbre de recherche progressivement en partant de la totalité de l'espace de recherche (la racine de l'arbre) jusqu'à arriver à un singleton représentant le vecteur optimal (une feuille de l'arbre). Lors de cette construction, l'idée directrice est de ne développer que le nœud de l'arbre ayant la meilleure évaluation.

1.2.2.2 Les méthodes stochastiques

Les méthodes stochastiques sont généralement utilisées pour les problèmes pour lesquels il n'existe pas un algorithme de résolution optimal en un temps polynomial. Ces méthodes ne réalisent pas une exploration exhaustive de l'espace des vecteurs de décision. En conséquence, elles ne fournissent pas forcément le vecteur optimal mais un vecteur approché. Néanmoins, en général, la qualité des vecteurs fournis est très bonne. Elles se caractérisent par deux propriétés :

- Un choix pseudo-aléatoire des vecteurs de décision ;
- Une heuristique qui permet de guider la convergence de l'algorithme.

Le recuit simulé et les algorithmes évolutionnistes sont deux exemples de méthodes stochastiques largement utilisées dans différentes applications.

Le recuit simulé Cette méthode a été inspirée du processus physique du "recuit" utilisé en métallurgie [79]. Ce processus consiste en une suite de cycles de refroidissement lent pour obtenir un matériau homogène et de très bonne qualité. En effet, lorsque le solide est à une forte température, chaque particule possède une très grande énergie et, par conséquent, peut réaliser de

grands déplacements aléatoires dans la matière. En refroidissant le matériau, l'énergie de chaque particule est diminuée et, par conséquent, l'étendue de ses déplacements est réduite. La méthode d'optimisation réalise une analogie avec ce processus physique. Effectivement, la fonction à optimiser F est assimilée à l'énergie du système et un paramètre T représentant la température du système est introduit. L'algorithme se base sur les principes suivants :

- À partir d'un vecteur initial x_0 , un déplacement aléatoire est effectué pour obtenir un deuxième vecteur x_1 ;
- Si x_1 est meilleur que x_0 alors x_1 est accepté et x_0 devient x_1 ;
- Si x_1 n'est pas meilleur que x_0 alors x_1 est accepté avec une probabilité $\exp(-\frac{F(x_1)-F(x_0)}{T})$.

Les algorithmes évolutionnistes Cette méthode a été inspirée par la théorie de l'évolution et le processus de sélection naturelle. En effet, les algorithmes évolutionnistes manipulent des populations d'individus qui évoluent au cours du temps. Ces populations sont constituées par des individus différents qui sont plus au moins adaptés à leur environnement. Les individus les plus adaptés se reproduisent plus efficacement et survivent plus longtemps. En se reproduisant, les différents individus transmettent une part de leurs caractéristiques à leurs descendants. Pour les problèmes d'optimisation, les vecteurs de décision sont assimilés aux individus et le degré d'adaptation à l'environnement est mesuré à l'aide de la fonction objectif. Les algorithmes génétiques [68] [54] sont les algorithmes évolutionnistes les plus populaires. Ils sont utilisés dans des domaines aussi variés que le transport [128] ou les systèmes de production [71]. Ces algorithmes reposent sur les concepts suivants :

Le concept de sélection consiste à choisir les individus qui vont se re-

produire et perdurer dans la population. Ce concept est analogue au processus de sélection naturelle. En effet, ce sont les individus les plus adaptés qui sont favorisés par rapport aux autres moins adaptés qui disparaissent de la population ;

Le concept de croisement consiste à créer un individu fils à partir de deux individus parents. Ce concept est similaire au concept de reproduction naturelle. Effectivement, les caractéristiques de l'individu fils sont issues des caractéristiques des deux parents ;

Le concept de mutation consiste à générer de manière aléatoire une portion des caractéristiques du fils lors de l'étape de croisement. Le taux de mutation est généralement défini entre 0.001 et 0.01. La valeur de ce taux doit être relativement faible afin de ne pas tomber dans une recherche aléatoire.

1.2.3 Optimisation multicritère

Dans le cas de l'optimisation multicritère, la fonction objectif est un vecteur ($F(x) = (F_1(x), F_2(x), \dots, F_l(x))$). Pour la résolution, deux approches sont généralement utilisées : l'approche par agrégation de critères et l'approche par Pareto dominance.

1.2.3.1 Optimisation multicritère par agrégation de critères

Dans cette approche, le but consiste à ramener le problème multicritère à un problème monocritère qui est plus simple à traiter. Dans la suite, deux techniques d'agrégation de critères sont présentées.

Agrégation par somme pondérée Ce type d'agrégation est une technique classique pour l'optimisation multicritère. Elle a été introduite par

Yager [129]. Elle consiste à ramener la valeur de la fonction objectif à un réel en réalisant la somme pondérée des composantes de $F(x)$. La nouvelle fonction objectif dans ce cas est définie comme suit :

Définition 7 Soient un ensemble de critères $E_c = \{c_1, c_2, \dots, c_l\}$, $x = (x_1, x_2, \dots, x_n)$ un vecteur de décision. La fonction objectif sur E_c est $F(x) = (F_1(x), F_2(x), \dots, F_l(x))$. La valeur de la nouvelle fonction objectif en x est :

$$F_{sp}(x) = \sum_{k=1}^l w_k \times F_k(x)$$

Avec w_k une pondération associée au critère c_k . Cette pondération permet d'exprimer des préférences sur les critères de décision. Ces pondérations vérifient les deux propriétés suivantes :

- $w_k \in [0,1] \ \forall k / 1 \leq k \leq n$;
- $\sum_{k=1}^n w_k = 1$

Agrégation par l'intégrale de Choquet L'intégrale de Choquet est une deuxième technique pour l'agrégation de critères plus sophistiquée que la première [59] [57]. Son intérêt est qu'elle permet de décrire des situations difficilement modélisable avec la première technique. Un exemple complet de l'utilisation de cette technique est présenté dans l'annexe A.

Définition 8 Soit un ensemble de critères $E_c = \{c_1, c_2, \dots, c_l\}$. Une mesure floue ou une capacité sur l'ensemble de critères E_c est la fonction $\mu : P(E_c) \longrightarrow [0,1]$, qui vérifie les axiomes suivants :

1. $\mu(\emptyset) = 0, \mu(E_c) = 1$
2. $\forall A, B \subseteq E_c, A \subseteq B \Rightarrow \mu(A) \leq \mu(B)$

$\mu(A)$ représente l'importance du sous-ensemble de critères $A \subseteq E_c$ dans la prise de décision.

Définition 9 Soient μ une mesure floue sur E_c , $x = (x_1, x_2, \dots, x_n)$ un vecteur de décision et $F(x) = (F_1(x), F_2(x), \dots, F_l(x))$ la fonction objectif par rapport à E_c . L'intégrale de Choquet discrète par rapport à la capacité μ est définie par :

$$F_\mu(x) = \sum_{i=1}^n (F_{\sigma(i)}(x) - F_{\sigma(i-1)}(x)) \times \mu(A_i)$$

avec

- σ une permutation sur E_c tel que $F_{\sigma(1)}(x) \leq F_{\sigma(2)}(x) \leq \dots \leq F_{\sigma(n)}(x)$;
- $F_{\sigma(0)}(x) = 0$;
- $A_i = \{F_{\sigma(i)}(x), F_{\sigma(i+1)}(x), \dots, F_{\sigma(n)}(x)\}$.

Deux notions sont importantes pour l'analyse sémantique de l'intégrale de Choquet. Ces deux notions sont l'importance globale d'un critère et l'interaction entre les critères.

Importance globale d'un critère Par définition $\mu(\{c_k\})$ dénote l'importance du critère c_k . Néanmoins, il ne faut pas se contenter de $\mu(\{c_k\})$ pour mesurer l'impact de c_k sur l'ensemble du problème. En effet, dans certains cas, $\mu(\{c_k\}) \approx 0$ alors qu'à chaque fois que le critère c_k est joint à un sous-ensemble de critères, l'importance de ce dernier augmente sensiblement. Ceci implique, qu'en réalité, le critère c_k est très important. Pour montrer ceci, considérons un exemple tiré de [59]. Dans cet exemple, les critères d'optimisation sont au nombre de trois : $E_c = \{c_1, c_2, c_3\}$. Les valeurs de la mesure floue pour tous les sous-ensembles de E_c sont indiquées dans le tableau 1.1.

A	$\{c_1\}$	$\{c_2\}$	$\{c_3\}$	$\{c_1, c_2\}$	$\{c_1, c_3\}$	$\{c_2, c_3\}$	$\{c_1, c_2, c_3\}$
$\mu(A)$	0	0.2	0.2	0.8	0.8	0.4	1

TAB. 1.1 – Description des états de l'exemple

Dans cet exemple, $\mu(\{c_1\}) = 0$ alors qu'en ajoutant le critère c_1 au critère c_2 ou c_3 la mesure floue de l'ensemble augmente considérablement. Par suite, le critère c_1 n'est pas inutile mais, au contraire, il est très important. Par conséquent, $\mu(\{c_k\})$ représente l'importance relative du critère c_k pris seul. Par contre, l'importance globale d'un critère est mesurée grâce à un indice appelé *indice de Shapley* noté ϕ [119]. Cet indice est obtenu en calculant la somme des plus-values que le critère c_k apporte par rapport à tous les sous-ensembles de $E_c \setminus \{c_k\}$. Il est défini comme suit pour $c_k \in E_c$:

$$\phi(c_k) = \sum_{A \subseteq E_c \setminus \{c_k\}} \frac{(|E_c| - |A| - 1)! |A|!}{|E_c|!} \times (\mu(A \cup \{c_k\}) - \mu(A))$$

Cet indice vérifie la propriété suivante : $\sum_{c_k \in E_c} \phi(c_k) = 1$. Pour l'exemple précédent, les *indices de Shapley* des critères sont : $\phi(c_1) = 0.4$, $\phi(c_2) = \phi(c_3) = 0.3$.

Interaction des critères Pour certains problèmes d'optimisation multicritère, il peut exister une synergie entre les critères. Cette synergie peut être négative ou positive. Par exemple, soient E_c un ensemble de critère pour un problème donné et c_l et c_m tel que $c_l, c_m \in E_c$. Si une synergie négative existe entre c_l et c_m alors il existe peu de vecteurs de décision qui présentent un bon score sur c_l et c_m simultanément. Dans ce cas, il serait pertinent de favoriser ce type de vecteurs. Pour d'autres problèmes, une synergie positive existe entre c_l et c_m . Dans ce cas, la majorité des vecteurs de décision qui possèdent un bon score sur c_l , possède un bon score sur c_m et inversement. En conséquence, il ne faut pas favoriser énormément les vecteurs présentant un bon score sur c_l et c_m par rapport aux autres. Grâce aux mesures floues, ce type de préférences peut être modélisé. Soient les deux critères c_l et c_m .

Trois cas se présentent à nous :

- Aucune synergie n'existe entre c_l et c_m . Les deux critères sont dits indépendants. Dans ce cas, $\mu(\{c_l, c_m\}) = \mu(\{c_l\}) + \mu(\{c_m\})$;
- Une synergie positive existe entre c_l et c_m . Les deux critères sont dits complémentaires. Dans ce cas, $\mu(\{c_l, c_m\}) > \mu(\{c_l\}) + \mu(\{c_m\})$;
- Une synergie négative existe entre c_l et c_m . Les deux critères sont dits redondants. Dans ce cas, $\mu(\{c_l, c_m\}) < \mu(\{c_l\}) + \mu(\{c_m\})$.

Pour mesurer l'interaction entre deux critères c_l et c_m , Murofushi et Soneda ont proposé un indice appelé *indice d'interaction* [99]. Cet indice se base sur la quantité de synergie entre c_l et c_m en présence d'un sous-ensemble de critères A . Et pour calculer la valeur de l'indice d'interaction, il faut calculer la moyenne de ces quantités de synergie avec A qui parcourt tous les sous-ensembles possibles de $E_c \setminus \{c_l, c_m\}$. L'indice d'interaction est défini par :

$$I_{l,m} = \sum_{A \subseteq E_c \setminus \{c_l, c_m\}} \frac{(|E_c| - |A| - 2)! |A|!}{(|E_c| - 1)!} \left(\mu(A \cup \{c_l, c_m\}) - \mu(A \cup \{c_l\}) - \mu(A \cup \{c_m\}) + \mu(A) \right)$$

Cette définition a été étendue par Grabish à n'importe quel sous-ensemble de critères $M \subseteq E_c$ [58].

$$I(M) = \sum_{A \subseteq E_c \setminus M} \frac{(|E_c| - |A| - |M|)! |A|!}{(|E_c| - |M| + 1)!} \sum_{B \subseteq M} (-1)^{|M| - |B|} \mu(A \cup B)$$

1.2.3.2 Optimisation multicritère par Pareto dominance

Pour pouvoir résoudre un problème d'optimisation multicritère, il est indispensable de pouvoir comparer deux vecteurs de décision [33]. Avec les techniques d'agrégation de critères la fonction objectif est ramenée à un réel. Par conséquent, la tâche est aisée. Néanmoins dans l'optimisation multicritère, $F(x)$ est un vecteur. C'est dans ce sens que cette deuxième approche se base sur la dominance et l'optimalité au sens de Pareto. Effectivement, grâce

à ces deux concepts, il est possible de comparer deux vecteurs de décision et de déterminer les vecteurs optimaux. Par conséquent, différentes techniques et métaheuristiques tel que les algorithmes génétiques peuvent être employées pour résoudre le problème [69] [45] [134] [19].

Définition 10 Soient $x = (x_1, x_2, \dots, x_n)$ et $y = (y_1, y_2, \dots, y_n)$ deux vecteurs de décision. y est dit dominé par x (ou x domine y) si $\forall k, 1 \leq k \leq l, F_k(x) \leq F_k(y)$ et $\exists k, 1 \leq k \leq l$ tel que $F_k(x) < F_k(y)$.

Définition 11 Soit $x = (x_1, x_2, \dots, x_n)$ un vecteur de décision. x est dit Pareto optimal s'il n'existe pas de solution y qui domine x .

Définition 12 Le front ou frontière de Pareto est l'ensemble des solutions Pareto optimales

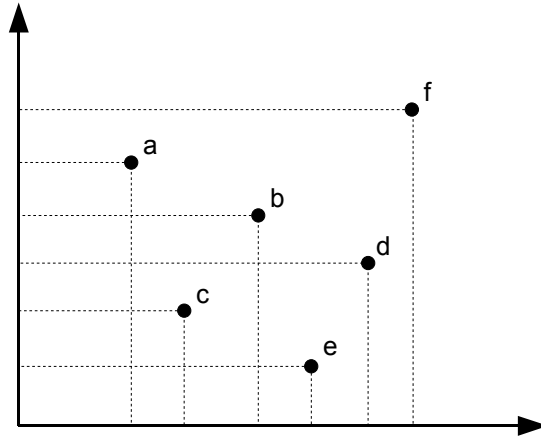


FIG. 1.1 – Exemple de dominance et d'optimalité au sens de Pareto

Pour mieux expliciter les notions précédemment définies, considérons l'exemple suivant. Soient six vecteurs de décision $a = (a_1, a_2)$, $b = (b_1, b_2)$, $c = (c_1, c_2)$, $d = (d_1, d_2)$, $e = (e_1, e_2)$, $f = (f_1, f_2)$. Soit la fonction objectif $F(x) =$

$(F_1(x_1), F_2(x_2))$. La représentation graphique de l'image de ces six vecteurs par la fonction F est fournie dans la figure 1.1. Dans cet exemple,

- f est dominé par tous les autres vecteurs ;
- b est dominé uniquement par c ;
- d est dominé par c et e ;
- a , c et e ne sont dominés par aucun vecteur.

Par conséquent, a , c et e sont Pareto optimaux. Donc, le front de Pareto est l'ensemble $\{a, c, e\}$.

1.3 Graphes et problèmes du plus court chemin

Cette section possède un double objectif. En premier lieu, elle situe le contexte des chapitres 2 et 3 en présentant une vision d'ensemble des différents types de graphes et des problématiques de plus court chemin associées. En deuxième lieu, elle permet de présenter les graphes statiques avec intervalles qui seront étendus dans le chapitre 3.

1.3.1 Définitions générales

Les graphes sont des concepts mathématiques utilisés pour modéliser des relations binaires entre des objets d'un même ensemble. Ils sont fréquemment utilisés pour modéliser des systèmes qui se présentent sous la forme d'un réseau. Il existe deux types de graphes : les graphes orientés et les graphes non orientés.

Définition 13 *Un graphe orienté G est un couple (N, A) avec N un ensemble dont les éléments sont appelés nœuds et A un ensemble dont les éléments sont des couples ordonnés de nœuds appelés arcs.*

Définition 14 *Un graphe non orienté G est un couple (N,A) avec N un ensemble dont les éléments sont appelés nœuds et A un ensemble dont les éléments sont des paires de nœuds appelées arrêtes.*

Par exemple, soient $G = (N,A)$, $i,j \in N$. i et j sont appelés nœuds du graphe G . Si G est un graphe orienté, $(i,j) \in A$ est appelé arc de G , i représente l'origine de l'arc et j représente sa destination. Si G est un graphe non orienté, $\{i,j\} \in A$ est appelée arrête de G et nous ne pouvons pas parler d'origine ou de destination dans ce cas. En effet, dans le cas d'un graphe orienté, $(i,j) \neq (j,i)$. Dans le cas contraire, $\{i,j\} \sim \{j,i\}$. Les figures 1.2 et 1.3 présentent respectivement un exemple d'un graphe orienté et un exemple d'un graphe non orienté.

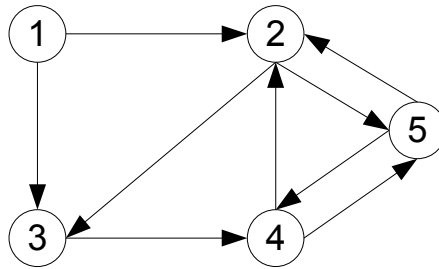


FIG. 1.2 – *Exemple d'un graphe orienté*

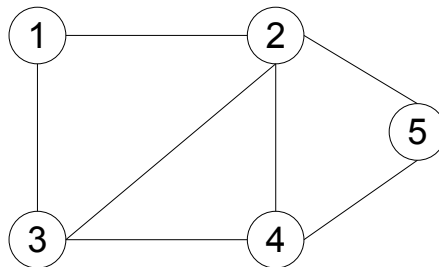


FIG. 1.3 – *Exemple d'un graphe non orienté*

Définition 15 *Un chemin entre deux nœuds i et j du graphe est une suite d'arcs liant i à j . Ce chemin est noté $Ch(i,j)$. L'ensemble de ces chemins est noté $ECh(i,j) = \{Ch(i,j)\}$. Formellement, $Ch(i,j) = (u_0, u_1, \dots, u_n)$ avec*

- $u_k \in N, \forall k \in [0, n]$
- $u_0 = i, u_n = j$
- $\forall u_k, u_{k+1} \in Ch(i,j), (u_k, u_{k+1}) \in A$

Un graphe G est dit pondéré si une fonction de poids lui est associée. Cette fonction, notée p , peut prendre différentes formes et, suivant sa nature, plusieurs classes de graphes peuvent être définies. De manière générale, le poids d'un arc appartient à \mathbb{R} . Cependant, dans la suite p sera supposée positive ($\forall (i,j) \in A, p(i,j) > 0$). Cette hypothèse a été considérée car elle permet de modéliser un coût positif afin de refléter la réalité des problèmes traités.

1.3.2 Graphes statiques déterministes

1.3.2.1 Définition

Soit $G = (N, A)$ un graphe orienté et pondéré. G est dit graphe statique déterministe si le poids de chaque arc est une valeur constante. La figure 1.4 présente un exemple d'un tel graphe.

1.3.2.2 Problème classique du plus court chemin

Le problème du plus court chemin pour les graphes statiques déterministes a fait l'objet de nombreuses études. Il est devenu un problème classique de la théorie des graphes [38]. L'objectif est de calculer le plus court chemin entre deux nœuds d'un graphe statique déterministe. Dans cette optique, les

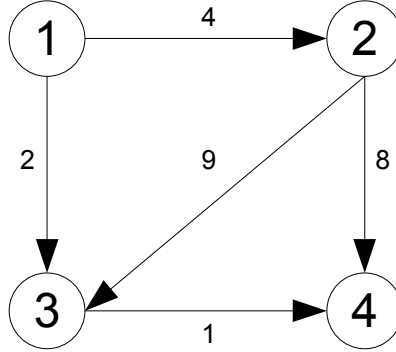


FIG. 1.4 – Exemple d'un graphe statique déterministe

notions de fonction de coût d'un chemin et du plus court chemin sont définies.

Définition 16 Soit $G = (N, A)$ un graphe statique déterministe. La fonction de coût d'un chemin dans G , notée C , est définie comme suit

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

Définition 17 Soit $G = (N, A)$ un graphe statique déterministe. Le plus court chemin de $i \in N$ à $j \in N$ est noté $PCC(i, j)$. $PCC(i, j) = Ch(i, j)$ avec $C(Ch(i, j)) = \min_{P \in ECh(i, j)} \{C(P)\}$. Dans le cas où il existe plusieurs chemins avec un coût minimal, un parmi eux est choisi aléatoirement.

De nombreux algorithmes ont été proposés pour résoudre ce problème [32] [51]. Néanmoins, l'algorithme le plus célèbre et le plus utilisé est l'algorithme proposé par E.W. Dijkstra [41]. Pour un nœud source donné, cet algorithme permet de déterminer le plus court chemin entre la source et tous les autres nœuds du graphe. Cependant, l'algorithme peut être utilisé pour calculer le plus court chemin entre un nœud source et un nœud destination. Plusieurs améliorations de l'algorithme initial ont été proposées afin de diminuer sa

complexité. Ces améliorations concernaient les structures de données utilisées dans l'algorithme [39] [74] [126]. Soient un graphe $G = (N, A)$ et $S_{source} \in N$ un nœud de G . L'algorithme de Dijkstra permettant de calculer le plus court chemin de S_{source} vers tous les autres nœuds du graphe est présenté dans la figure 1.5. Cet algorithme utilise deux ensembles particuliers : l'ensemble des nœuds candidats et l'ensemble des nœuds fermés. L'ensemble des nœuds candidats, noté Q , est l'ensemble des nœuds dont le plus court chemin n'a pas encore été déterminé. Par contre, l'ensemble des nœuds fermés, noté F , est l'ensemble des nœuds dont le plus court chemin a été calculé de manière définitive.

1.3.3 Graphes statiques stochastiques

1.3.3.1 Définition

Soit $G = (N, A)$ un graphe orienté et pondéré. G est appelé graphe statique stochastique si le poids d'un arc $(i, j) \in A$ est défini par une distribution de probabilité discrète. Le poids de l'arc (i, j) est défini de la manière suivante

$$p(i, j) = \begin{cases} c_1, & \rho_1 \\ c_2, & \rho_2 \\ \vdots & \vdots \\ c_m, & \rho_m \end{cases}$$

avec $c_1, c_2, \dots, c_m \in \mathbb{R}_+$ les poids possibles pour l'arc (i, j) et $\rho_1, \rho_2, \dots, \rho_m$ les probabilités associées aux poids. Ces probabilités doivent vérifier la condition suivante : $\sum_{k=1}^m \rho_k = 1$. Un exemple d'un graphe statique stochastique est présenté dans la figure 1.6. Dans cet exemple, le poids de l'arc $(1, 2)$ est égal à 1 avec une probabilité de 0.8 et il est égal à 3 avec une probabilité de 0.2.

```

1 //Initialiser les plus court chemins
2  $PCC(S_{source}, S_{source}) = (S_{source})$ 
3  $\forall i \in N \setminus i \neq S_{source}, PCC(S_{source}, i) = \emptyset$ 
4 //Initialiser l'ensemble  $Q$ 
5  $Q = \{S_{source}\}$ 
6 //Initialiser l'ensemble  $F$ 
7  $F = \emptyset$ 
8 Tant que  $Q \neq \emptyset$  faire
9     Choisir  $i$  de  $Q$  tel que  $C(PCC(S_{source}, i))$  soit minimal
10     $Q = Q \setminus \{i\}$  et  $F = F \cup \{i\}$ 
11    Développer les successeurs  $j$  de  $i$ 
12    Pour chaque  $j$  faire
13        Si  $C(PCC(S_{source}, i)) + p(i, j) < C(PCC(S_{source}, j))$  Alors
14             $PCC(S_{source}, j) = PCC(S_{source}, i) \cup \{j\}$ 
15             $Q = Q \cup \{j\}$ 
16        Fin Si
17    Fin Pour
18 Fin Tant que

```

FIG. 1.5 – *Algorithme de Dijkstra*

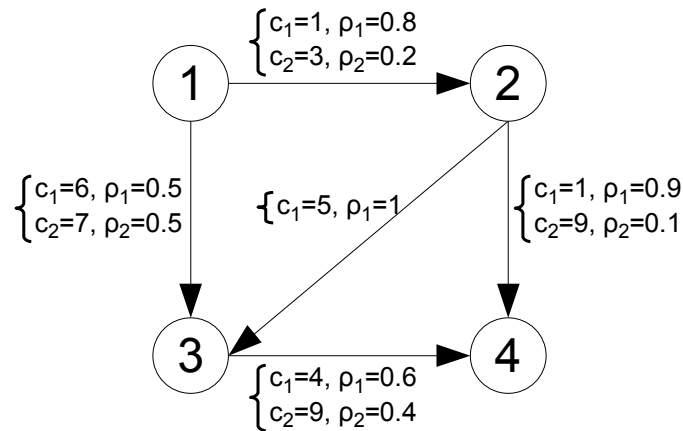


FIG. 1.6 – Exemple d'un graphe statique stochastique

1.3.3.2 Problème du plus court chemin stochastique

La notion de chemin optimal pour le problème du plus court chemin stochastique a été définie de différentes manières dans la littérature. La définition la plus utilisée est celle qui considère comme optimal le chemin qui maximise la valeur espérée d'une fonction d'utilité définie par le décideur [88] [18] [96] [100] [114]. D'autres auteurs ont défini le chemin optimal comme étant le chemin qui maximise la probabilité que la longueur totale du chemin n'excède pas une certaine limite fixée [47] [15]. Une autre définition présente dans la littérature considère le chemin optimal comme étant celui qui possède la plus forte probabilité d'être le chemin le plus court [120] [1] [75]. La définition la plus simple à traiter dans le cas statique est celle qui décrit l'optimalité d'un chemin par rapport au coût espéré du chemin [73]. Dans ce cas, pour calculer le plus court chemin, il faut calculer l'espérance mathématique du poids de chaque arc. Ensuite, il suffit de dérouler un algorithme de calcul du plus court chemin pour les graphes déterministes comme l'algorithme de Dijkstra en considérant les espérances mathématiques calculées comme étant

les poids des arcs.

1.3.4 Graphes dynamiques déterministes

1.3.4.1 Définition

Soit $G = (N, A)$ un graphe orienté et pondéré. G est dit graphe dynamique déterministe si le poids d'un arc est une valeur déterministe mais qui dépend du temps. Par conséquent, le poids d'un arc $(i, j) \in A$ est décrit par la fonction $p(i, j, t)$ avec t la date de départ du nœud i . Pour ce type de graphes, le temps peut être considéré continu ou discret. Cependant, pour simplifier le problème, le temps est généralement supposé discret ou il est supposé continu avec p une fonction constante par intervalles par rapport au temps. Dans ce dernier cas, le temps est divisé en périodes égales de longueur T . Durant chaque période T , la fonction de poids est supposée constante. Il faut noter que, contrairement aux réseaux de Pétri, le temps n'est pas réinitialisé au niveau de chaque nœud. La figure 1.7 présente un exemple d'un tel cas de figure. Dans cet exemple, le poids de chaque arc est défini pour trois intervalles de temps. Supposons que $T = 5$ unités de temps. Le poids de l'arc $(1, 2)$ est égal à 2 pour $t \in [0, 5[$, il est égal à 5 pour $t \in [5, 10[$ et il est égal à 4 pour $t \in [10, 15[$.

1.3.4.2 Graphes FIFO

Les graphes FIFO sont une sous-classe des graphes dynamiques déterministes. Un graphe dynamique déterministe est dit graphe FIFO si tous ses arcs sont des arcs FIFO. Un arc $(i, j) \in A$ est appelé arc FIFO si sa fonction de poids est définie de telle manière que partir du nœud i pour le nœud j à $t' \geq t$ implique nécessairement d'arriver à j plus tard que $t + p(i, j, t)$.

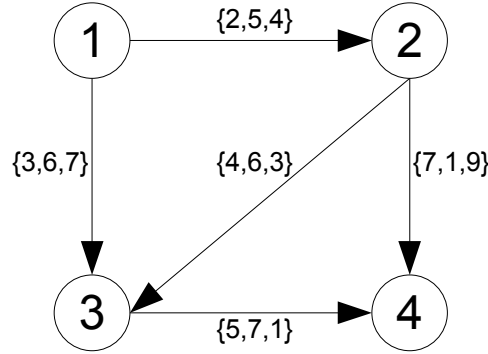


FIG. 1.7 – Exemple d'un graphe dynamique déterministe

Formellement, la définition est la suivante

Définition 18 Soit $G = (N, A)$ un graphe dynamique déterministe. Un arc $(i, j) \in A$ est dit arc FIFO s'il vérifie la propriété suivante :

$$\forall t, t' \geq 0, t \leq t' \implies t + p(i, j, t) \leq t' + p(i, j, t')$$

Pour illustrer un arc FIFO, soit une route avec une seule voie. La première voiture qui emprunte cette route est certaine d'arriver la première car aucune autre voiture ne peut la dépasser. Par conséquent, cette route peut être modélisée par un arc FIFO.

1.3.4.3 Problème du plus court chemin dynamique

Le problème du plus court chemin dynamique a été formulé pour la première fois par L. Cooke et E. Halsey [34]. La majorité des travaux qui ont suivi ont étudié le problème en supposant que le temps est discret [28] [29] [44] [80]. Le cas où le temps est considéré comme continu n'a pas reçu beaucoup d'attention. Les travaux de référence qui ont traité ce cas sont ceux de A. Orda et R. Rom [103] [104]. Plusieurs variantes du problème existent mais

toutes peuvent être exprimées comme des cas particuliers de deux problématiques fondamentales [37]. La première problématique est le calcul du plus court chemin d'un nœud source vers tous les autres nœuds du graphe pour une date de départ du nœud source fixée. La deuxième problématique est le calcul du plus court chemin d'un nœud source vers tous les autres nœuds du graphe pour toutes les dates de départ possibles du nœud source. Dans la suite, nous allons nous focaliser sur la première problématique. Les notions de fonction de coût d'un chemin et du plus court chemin pour un graphe dynamique déterministe sont définies comme suit.

Définition 19 Soit $G = (N, A)$ un graphe dynamique déterministe. Soit CD la fonction de coût d'un chemin dans G . $CD^{t_0}((u_0, u_1, \dots, u_m))$ indique le coût du chemin (u_0, u_1, \dots, u_m) en partant du nœud u_0 à t_0 . La valeur de $CD^{t_0}((u_0, \dots, u_m))$ est définie comme suit :

$$\begin{cases} CD^{t_0}((u_0, \dots, u_{m-1})) + p(u_{m-1}, u_m, CD^{t_0}((u_0, \dots, u_{m-1}))) & \text{Si } m > 0 \\ t_0 & \text{Sinon } m = 0 \end{cases}$$

Définition 20 Soit $G = (N, A)$ un graphe dynamique déterministe. Le plus court chemin de $i \in N$ à $j \in N$ en partant de i à l'instant t_0 est noté $PCC^{t_0}(i, j)$. $PCC^{t_0}(i, j) = Ch(i, j)$ avec $CD^{t_0}(Ch(i, j)) = \min_{P \in ECh(i, j)} \{CD^{t_0}(P)\}$. Au cas où plusieurs chemins avec un coût minimal existent, un parmi eux est choisi aléatoirement.

Le problème du calcul du plus court chemin dynamique pour une date de départ fixée a été prouvé comme étant *NP-Difficile* [103]. Néanmoins, il a été prouvé qu'il est polynomial pour la classe des graphes FIFO. Dans ce dernier cas, un algorithme de Dijkstra adapté au contexte dynamique peut être utilisé et le résultat qu'il retourne est optimal [2] [78]. Soient $G = (N, A)$ un graphe FIFO, $S_{source} \in N$. L'algorithme de Dijkstra calculant le plus court

chemin de S_{source} vers tous les autres nœuds de G en partant de S_{source} à t_0 est présenté dans la figure 1.8.

Pour les graphes non-FIFO, un paradigme algorithmique, appelé *Chrono-SPT*, a été proposé par S. Pallottino et M.G. Scutella [106]. Ce paradigme est valable pour les cas où le temps est considéré comme discret. L'idée est de visiter les nœuds dans un ordre chronologique. Dans cette optique, les auteurs définissent un échéancier avec une structure de données appelée *bucket-list*. Cette liste, notée B , est composée de q éléments $B = \{B_1, B_2, \dots, B_q\}$. Chaque élément de la liste B_k contient les nœuds qui sont visités à l'instant t_k . Un exemple d'une telle liste est présenté dans la figure 1.9. L'idée du paradigme consiste à visiter les éléments de B dans un ordre chronologique : B_0, B_1, B_2, \dots . Dans une itération typique, tous les successeurs des nœuds contenus dans B_k sont alors traités pour mettre à jour leur label. Un exemple de cette itération est présentée dans la figure 1.10. Dans cet exemple, $Pred_i(t)$ dénote le prédécesseur courant du nœud i à l'instant t (noté i_t) et $L_i(t)$ dénote le label associé au nœud i à l'instant t .

1.3.5 Graphes dynamiques stochastiques

1.3.5.1 Définition

Soit $G = (N, A)$ un graphe orienté et pondéré. G est appelé graphe dynamique stochastique si le poids d'un arc $(i, j) \in A$ est défini par une distribution de probabilité discrète et que cette dernière dépende du temps. D'où, le poids de l'arc (i, j) est défini par la fonction $p(i, j, t)$ avec t la date de départ de i . La fonction $p(i, j, t)$ est déterminée, à chaque instant t , par une distribution

```

1 //Initialiser les plus court chemins
2  $PCC^{t_0}(S_{source}, S_{source}) = (S_{source})$ 
3  $\forall i \in N \setminus i \neq S_{source}, PCC^{t_0}(S_{source}, i) = \emptyset$ 
4 //Initialiser l'ensemble  $Q$ 
5  $Q = \{S_{source}\}$ 
6 //Initialiser l'ensemble  $F$ 
7  $F = \emptyset$ 
8 Tant que  $Q \neq \emptyset$  faire
9     Choisir  $i$  de  $Q$  tel que  $CD^{t_0}(PCC^{t_0}(S_{source}, i))$  soit minimal
10     $Q = Q \setminus \{i\}$  et  $F = F \cup \{i\}$ 
11    Développer les successeurs  $j$  de  $i$ 
12    Pour chaque  $j$  faire
13        Si  $CD^{t_0}(PCC^{t_0}(S_{source}, i)) + p(i, j, CD^{t_0}(PCC^{t_0}(S_{source}, i))) <$ 
14         $CD^{t_0}(PCC^{t_0}(S_{source}, j))$  Alors
15             $PCC^{t_0}(S_{source}, j) = PCC^{t_0}(S_{source}, i) \cup \{j\}$ 
16             $Q = Q \cup \{j\}$ 
17        Fin Si
18    Fin Pour
19 Fin Tant que
    
```

FIG. 1.8 – Version dynamique de l'algorithme de Dijkstra

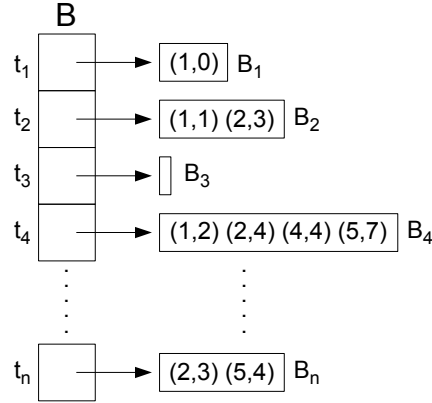


FIG. 1.9 – *Exemple d'une bucket-list*

Sélectionner i de B_k

$B_k = B_k \setminus \{i\}$

Développer les successeurs j de i

Pour chaque j faire

$t_h = t_k + p(i, j, t_k)$

Si $L_i(t_k) + p(i, j, t_k) < L_j(t_h)$

$Pred_j(t_h) = i_k$

Si $j \notin B_h$ Alors $B_h = B_h \cup \{j\}$

Fin Si

Fin Si

Fin Pour

FIG. 1.10 – *Itération typique du paradigme Chrono-SPT*

de probabilité de la forme

$$p(i,j,t) = \begin{cases} c_1^t, & \rho_1^t \\ c_2^t, & \rho_2^t \\ \vdots & \vdots \\ c_m^t, & \rho_m^t \end{cases}$$

avec $c_1^t, c_2^t, \dots, c_m^t \in \mathbb{R}_+$ les poids possibles pour l'arc (i,j) à l'instant t et $\rho_1^t, \rho_2^t, \dots, \rho_m^t$, les probabilités associées aux poids à l'instant t . La condition suivante sur les probabilités ρ_k^t doit être vérifiée $\forall t$, $\sum_{k=1}^m \rho_k^t = 1$. Un exemple d'un tel graphe est présenté dans la figure 1.11. Dans cet exemple, $\forall t < 10$, le poids de l'arc $(1,2)$ est égal à 7 avec une probabilité de 0.5 et il est égal à 5 avec une probabilité de 0.5. Par contre, $\forall t \geq 10$, le poids de l'arc $(1,2)$ est égal à 5 avec une probabilité de 1.

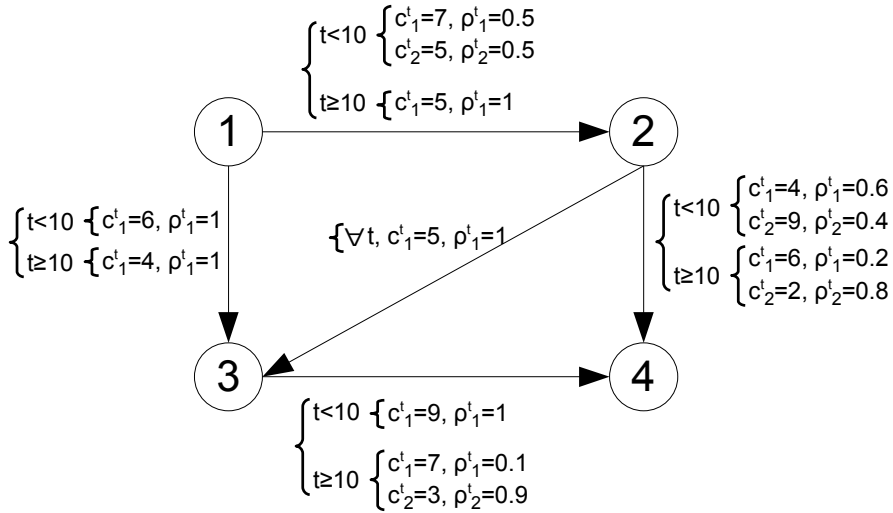


FIG. 1.11 – Exemple d'un graphe dynamique stochastique

1.3.5.2 Problème du plus court chemin dynamique stochastique

R.W. Hall est le premier auteur à avoir traité les graphes dynamiques stochastiques [60]. Il a étudié le problème du calcul d'un chemin entre deux nœuds avec un coût espéré minimal. Le problème s'est avéré plus compliqué que dans le cas des graphes statiques stochastiques à cause de la nature stochastique et dynamique du graphe. Dans son étude, l'auteur a proposé deux approches pour le calcul d'un tel chemin. La première approche consiste en un calcul *a priori* du chemin. La deuxième approche consiste à développer une stratégie de *routage* en-ligne permettant de réagir aux poids effectifs des nœuds intermédiaires pour choisir efficacement les nœuds suivants. Cependant, R.W. Hall a montré que la deuxième approche était plus efficace que la première. Concernant la première approche, D. Pretolani a prouvé que le problème du calcul a priori du chemin avec un coût espéré minimal est *NP-Difficile* [110]. Pour la résolution, R.W. Hall a proposé un algorithme de Séparation-Évaluation [60] et E.D. Miller-Hooks et H.S. Mahmassani ont proposé un algorithme à correction de label [94] [95]. La deuxième approche a reçu plus d'attention de la part des auteurs. Dans la majorité des travaux, la stratégie de routage est basée uniquement sur les temps d'arrivées au niveau des nœuds de décision et les distributions des poids des arcs sont supposées indépendantes [60] [27] [16]. Deux stratégies de référence existent. La première se base sur le concept DOT (Decreasing Order of Time) [53]. Ce concept, développé par I. Chabini [30], implique l'affectation des labels aux nœuds dans un ordre décroissant par rapport au temps. La deuxième stratégie se base sur un algorithme à correction de label développé par E.D. Miller-Hooks et H.S. Mahmassani et qui représente une évolution de l'algorithme qu'ils ont proposé pour le calcul d'un chemin optimal a priori [95].

1.3.6 Graphes statiques avec intervalles

1.3.6.1 Définition

Soit $G = (N, A)$ un graphe orienté et pondéré. Le graphe G est appelé graphe statique avec intervalles si un intervalle noté $[a_{ij}, b_{ij}]$ avec $0 < a_{ij} \leq b_{ij}$ est associé à chaque arc du graphe (i, j) . Le poids de (i, j) , noté $p(i, j)$ peut prendre n'importe quelle valeur de $[a_{ij}, b_{ij}]$ ($p(i, j) \in [a_{ij}, b_{ij}]$). Un exemple d'un tel graphe est fourni dans la figure 1.12. Dans cet exemple, l'intervalle $[2, 5]$ est associé à $(1, 2)$ d'où $p(1, 2) \in [2, 5]$.

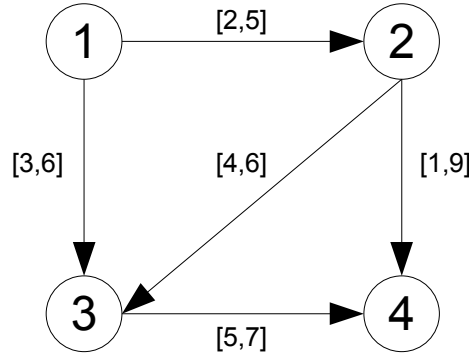


FIG. 1.12 – Exemple d'un graphe statique avec intervalles

Pour ce type de graphe, la notion de scénario est introduite.

Définition 21 *Un scénario est la réalisation des poids des arcs du graphe. En d'autres termes, pour un scénario s le poids de chaque arc $(i, j) \in A$ est fixé. Le poids de l'arc (i, j) pour le scénario s est noté $p^s(i, j)$.*

La figure 1.13 représente un scénario du graphe de la figure 1.12. En fixant le scénario, le graphe statique avec intervalles devient un graphe statique déterministe comme présenté précédemment. Ce type de graphe sera étendu

au cas dynamique (les poids des arcs dépendent du temps) dans le chapitre 3.

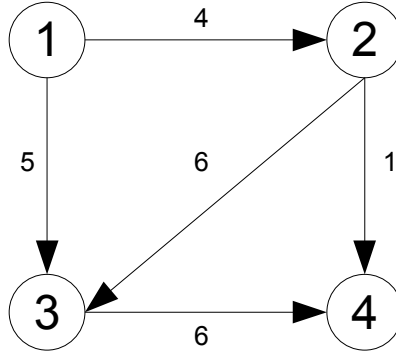


FIG. 1.13 – *Exemple d'un scénario*

1.3.6.2 Problème du plus court chemin de déviation robuste

Les graphes statiques avec intervalles ont été proposés afin de modéliser l'incertitude sur les poids des arcs [77] [40]. Pour ce type de graphes et dans le cadre du problème de plus court chemin, il fallait définir la notion de chemin optimal. En conséquence, la notion de déviation robuste d'un chemin a été proposée [77]. Cette notion est définie comme suit :

Définition 22 *La déviation robuste d'un chemin entre $i, j \in N$ pour un scénario s est la différence entre le coût de $Ch(i, j)$ dans le scénario s et le coût du plus court chemin entre i et j dans le scénario s . Cette déviation est notée $D^s(Ch(i, j))$. La déviation robuste maximale de $Ch(i, j)$ pour tous les scénarios possibles est notée $D_{max}(Ch(i, j)) = \max_s \{D^s(Ch(i, j))\}$.*

Pour illustrer cette notion, soient le scénario s présenté dans la figure 1.13 et le chemin $(1, 3, 4)$. Dans le scénario s , le plus court chemin du nœud 1 au nœud 4 est le chemin $(1, 2, 4)$. Par conséquent, $D^s((1, 3, 4)) = C((1, 3, 4)) -$

$C((1,2,4)) = (5 + 6) - (4 + 1) = 6$. Ayant ce critère de déviation robuste, le plus court chemin de déviation robuste est défini comme suit :

Définition 23 Soit $Ch(i,j)$ $i,j \in N$ un chemin de i à j . $Ch(i,j)$ est dit le plus court chemin de déviation robuste, noté $PCCDR(i,j)$, s'il possède la plus petite déviation robuste maximale parmi tous les chemins de i à j . Dans le cas où il existe plusieurs chemins avec une déviation robuste maximale qui est minimale, un parmi eux est choisi aléatoirement.

Il a été prouvé que le problème du plus court chemin de déviation robuste est *NP-Difficile* pour un nombre fini et infini de scénarios [130] [81] [133]. Dans la littérature, deux heuristiques de référence existent. La première repose sur une exploration itérative des chemins possibles [97]. La deuxième implémente la technique de Séparation-Évaluation [98]. Cependant, les deux algorithmes se basent sur la même observation. Effectivement, il a été prouvé que le scénario qui maximise la déviation robuste d'un chemin est le scénario où les arcs formant le chemin possèdent des poids maximaux et tous les autres arcs du graphe possèdent des poids minimaux [77]. En d'autres termes, soient $G = (N,A)$ un graphe statique avec intervalles et P un chemin dans G . Le scénario qui maximise la déviation robuste est le scénario s tel que

$$\forall (i,j) \in A, p^s(i,j) = \begin{cases} b_{ij} & \text{Si } (i,j) \in P \\ a_{ij} & \text{Sinon} \end{cases}$$

Cette observation est utilisée par les deux algorithmes pour calculer la déviation maximale pour un chemin donné. En conséquence, la différence principale entre les deux algorithmes est la méthode d'exploration des chemins dans l'ensemble des chemins possibles.

1.4 Critères de prise de décision dans un environnement incertain

La nécessité de prendre des décisions face à l'incertitude fait partie intégrante de notre vie. Effectivement, des situations où il n'est pas possible de connaître à l'avance et de manière exacte les conséquences d'une décision se présentent aussi bien dans la vie professionnelle que dans la vie privée. Afin de fournir une méthodologie rationnelle de prise de décision, la théorie de décision a été développée [101] [63] [8]. Vu l'importance du sujet pour les décideurs dans les entreprises qui doivent prendre des décisions sur le long terme, cette théorie a trouvé un large écho dans le monde économique [56] [122]. Une situation de prise de décision dans un environnement incertain peut être définie formellement de la manière suivante. Soit une situation de la nature qui possède m états possibles e_1, e_2, \dots, e_m . Dans ce contexte, il est possible de réaliser n actions a_1, a_2, \dots, a_n . Pour chaque couple "action/état", il y a un résultat $R(a_i, e_j)$ qui peut être positif ou négatif. Dans cet environnement incertain, une action à entreprendre doit être choisie. Il faut noter que le système est considéré sans mémoire et, par conséquent, le résultat ne dépend que de l'état actuel et en aucune manière des actions précédentes. Dans le but de réaliser un choix rationnel de l'action, plusieurs critères de décision ont été développés dans le cadre de la théorie de la décision. Pour ces différents critères, il faut calculer en premier lieu la valeur $V(a_i)$ de chaque action en tenant compte des différents états possibles e_j . Par la suite, il faut déterminer l'action optimale a^* en se basant sur les valeurs des actions calculées. Dans la suite, les différents critères de prise de décision sont présentés. Ils se divisent en deux grandes catégories : les critères quantitatifs et les critères probabilistes. Le premier type de critères ne se base que sur les

résultats attendus pour calculer les valeurs des actions. Le deuxième type de critères, tient compte des probabilités d'occurrence de chaque état en plus des résultats attendus des actions. Pour ce type de critères, la probabilité d'occurrence de l'état e_j est supposée connue et elle est notée $P(e_j)$. Dans la suite, les différents critères sont définis formellement. Un exemple d'utilisation est présenté dans l'annexe B. Cette notion de critère de prise de décision dans l'incertain sera utilisée par la suite dans le chapitre 3 afin de choisir un itinéraire dans un modèle intégrant l'incertitude dans sa définition.

1.4.1 Critères quantitatifs

1.4.1.1 Critère de Laplace

Ce critère repose sur une hypothèse d'équiprobabilité entre les différents évènements. Il consiste à évaluer chaque action en calculant la moyenne des résultats possibles comme suit :

$$V(a_i) = \frac{1}{m} \sum_{j=1}^m R(a_i, e_j)$$

L'action choisie est celle avec la moyenne la plus forte :

$$a^* \in \{a_1, a_2, \dots, a_n\} / V(a^*) = \max_{i \in \{1, 2, \dots, n\}} \{V(a_i)\}$$

1.4.1.2 Critère de Wald ou du MaxiMin

Ce critère est un critère de pessimisme total. En effet, pour chaque action, il considère le pire des cas. Ce cas est défini comme l'évènement avec le résultat minimal :

$$V(a_i) = \min_{j \in \{1, 2, \dots, m\}} \{R(a_i, e_j)\}$$

L'action choisie est celle avec le meilleur résultat dans le pire des cas :

$$a^* \in \{a_1, a_2, \dots, a_n\} / V(a^*) = \max_{i \in \{1, 2, \dots, n\}} \{V(a_i)\}$$

1.4.1.3 Critère du MaxiMax

Contrairement au critère précédent, ce critère est un critère d'optimisme total. En effet, pour chaque action, il considère le meilleur des cas. Ce cas est défini comme l'évènement avec le résultat maximal :

$$V(a_i) = \max_{j \in \{1,2,\dots,m\}} \{R(a_i, e_j)\}$$

L'action choisie est celle avec le meilleur résultat dans le meilleur des cas :

$$a^* \in \{a_1, a_2, \dots, a_n\} / V(a^*) = \max_{i \in \{1,2,\dots,n\}} \{V(a_i)\}$$

1.4.1.4 Critère d'Hurwitz

Ce critère combine l'approche optimiste et l'approche pessimiste en instaurant un coefficient d'optimisme. En effet, pour chaque action il considère le résultat maximal et le résultat minimal. La valeur maximale sera affectée d'un coefficient d'optimisme α et la valeur minimale d'un coefficient $(1 - \alpha)$:

$$V(a_i) = \alpha \times \max_{j \in \{1,2,\dots,m\}} \{R(a_i, e_j)\} + (1 - \alpha) \times \min_{j \in \{1,2,\dots,m\}} \{R(a_i, e_j)\}$$

L'action choisie est celle avec la valeur la plus élevée :

$$a^* \in \{a_1, a_2, \dots, a_n\} / V(a^*) = \max_{i \in \{1,2,\dots,n\}} \{V(a_i)\}$$

Ce critère est une généralisation des critères MaxiMin et MaxiMax. En effet, en définissant $\alpha = 1$, le critère d'Hurwitz se ramène au critère d'optimisme total (MaxiMax). En définissant $\alpha = 0$, le critère d'Hurwitz se ramène au critère de pessimisme total (MaxiMin). Dans la suite, ce critère sera utilisé dans la section 3.3.3 afin de calculer un itinéraire optimal dans des graphes où les poids des arcs ne sont pas connus de manière certaine.

1.4.1.5 Critère de Savage ou du MiniMax

Ce critère se base sur la notion de regret pour un état donné. Le regret pour un état est défini comme étant le manque à gagner en ne choisissant pas “la bonne action” pour cet état. Pour chaque action, il faut d’abord calculer le regret pour chaque état. Puis la valeur de chaque action est défini comme étant le regret maximal :

$$V(a_i) = \max_{j \in \{1,2,\dots,m\}} \left\{ \left(\max_{k \in \{1,2,\dots,n\}} \{R(a_k, e_j)\} - R(a_i, e_j) \right) \right\}$$

L’action choisie est celle avec le regret maximal le plus faible :

$$a^* \in \{a_1, a_2, \dots, a_n\} / V(a^*) = \min_{i \in \{1,2,\dots,n\}} \{V(a_i)\}$$

1.4.2 Critères probabilistes

1.4.2.1 Critère de Pascal

Ce critère ramène le calcul de la valeur d’une action au calcul de son espérance mathématique. La valeur de l’action est donnée par la formule suivante :

$$V(a_i) = E(a_i) = \sum_{j=1}^m R(a_i, e_j) \times P(e_j)$$

L’action choisie est celle avec l’espérance la plus élevée :

$$a^* \in \{a_1, a_2, \dots, a_n\} / V(a^*) = \max_{i \in \{1,2,\dots,n\}} \{V(a_i)\}$$

1.4.2.2 Critère de Markowitz

Comme le critère de Pascal, le critère de Markowitz repose sur le calcul de l’espérance mathématique de chaque action. Cependant, il tient compte du risque lié à une action en intégrant la dispersion de ses résultats. Cette

dispersion est mesurée en calculant l'écart-type défini par la formule suivante :

$$\sigma(a_i) = \sqrt{\sum_{j=1}^m (R(a_i, e_j) - E(a_i))^2 \times P(e_j)}$$

Dans ce cas, la valeur d'une action est définie comme suit :

$$V(a_i) = \frac{E(a_i)}{\sigma(a_i)}$$

L'action choisie est celle avec le pourcentage d'espérance par unité d'écart type le plus élevé :

$$a^* \in \{a_1, a_2, \dots, a_n\} / V(a^*) = \max_{i \in \{1, 2, \dots, n\}} \{V(a_i)\}$$

1.5 Méthodes d'interception d'un mobile

1.5.1 Interception dans un espace continu

L'interception dans un espace continu a été largement étudiée pour ses applications pratiques. En effet, l'interception d'un mobile dans le plan ou dans l'espace présente un grand intérêt pour le monde de la robotique. L'interception dans un plan à deux dimensions a été traitée par de nombreux auteurs. Plusieurs articles ont étudié le cas où la trajectoire du mobile à intercepter était prévisible. Dans ce cas, il faut déterminer un point d'interception unique [107] [92]. Ce problème a connu une première extension consistant à supposer que la trajectoire du mobile objectif était parfaitement prévisible mais avec incertitude. En conséquence, le point d'interception devait être constamment réadapté [35] [72]. Ensuite, le problème a été étendu au cas où la trajectoire de l'objectif était imprévisible. Avec cette hypothèse, il n'est plus possible de déterminer un point d'interception mais il faut poursuivre l'objectif jusqu'à réaliser la rencontre [85] [20] [52]. Ces différents travaux ont

trouvé des applications pratiques dans le domaine de la robotique [118] [25] [21]. Par la suite, le problème d'interception a été généralisé à l'espace à trois dimensions. Certains auteurs ont étudié les mécanismes mis en œuvre par les humains pour intercepter des objets tels que des balles [90] [91]. L'objectif de ces études était de déterminer des méthodes et des algorithmes permettant à des robots d'intercepter des mobiles dans l'espace [124] [23] [121].

1.5.2 Interception dans les graphes

À notre connaissance, les seuls travaux qui traitent de l'interception d'un mobile dans un graphe sont ceux de M.M. Hizem et *al.* [64] [65] [66] [67]. Cependant, il existe certains travaux qui étudient des problèmes assez similaires.

1.5.2.1 Le problème de Poursuite-Évasion

Le problème de Poursuite-Évasion a été initialement motivé par la communauté de spéléologie soucieuse d'améliorer ses techniques de recherche des personnes perdues dans les cavités souterraines [24]. Par la suite, ce problème a été formalisé par T. Parsons en termes de théorie des graphes [108] [109]. D'une manière générale, le problème de Poursuite-Évasion dans un graphe (appelé aussi jeu du gendarme et du voleur) est défini comme suit. Soit $G = (N, A)$ un graphe avec N l'ensemble des nœuds et A l'ensemble des arcs. Soient deux catégories d'agents qui se déplacent d'une manière alternée sur le graphe G : un intrus et des poursuivants. Le déplacement d'un agent sur le graphe s'effectue en passant du nœud dans lequel il se trouve vers un nœud adjacent (lié par un arc). Tous les agents sont autorisés à ne pas se déplacer et à rester dans le nœud qu'ils occupent. Les poursuivants ont pour but la capture de l'intrus qui tente de leur échapper. Un poursuivant

capture l'intrus s'il occupe le même nœud que lui au même moment. Tous les agents sont supposés avoir une connaissance complète de la position des autres agents. Les différents auteurs ont commencé par traiter le cas avec un intrus et un seul poursuivant [102] [111] [112] [113]. Par la suite, le problème a été généralisé à k poursuivants et un intrus [13] [14] [48] [49] [62] [89]. Les objectifs principaux de ces différentes études étaient de déterminer si la capture était possible ou pas et de trouver le nombre minimal de poursuivants capables de capturer l'intrus. Par exemple, il a été prouvé que trois poursuivants étaient toujours capables de capturer un intrus dans un graphe [3]. De plus, il a été démontré que le problème consistant à déterminer si k poursuivants pouvaient capturer un intrus est *EXPTIME-Comple*¹ [55]. Avec le temps de nombreuses variantes du problème classique sont apparues. Par exemple, dans l'une d'elles, les k poursuivants doivent capturer l'intrus avant qu'il n'atteigne un nœud spécial appelé "abri" [31] [115] [116].

1.5.2.2 Le problème du Rendez-vous

Le problème du Rendez-vous a été initialement posé par T. Schelling [117]. Dans son livre, l'auteur donnait l'exemple de deux parachutistes qui atterrissent en un territoire inconnu et qui doivent se retrouver. Il discutait alors le problème posé à ces deux parachutistes et des méthodes possibles pour qu'ils puissent se rencontrer. Par la suite, le problème a été présenté comme étant un problème d'optimisation. D'où, deux formulations ont été suggérées. La première formulation, connue sous le nom du "*problème de l'astronaute*", a été proposée comme formulation générale. La deuxième, connue sous le nom du "*problème du téléphone*", représente une version discrète et simplifiée de

1. Un problème est dit EXPTIME s'il peut être résolu par un algorithme déterministe de complexité $O(2^{p(n)})$ avec $p(n)$ un polynôme fonction de n .

la formulation générale.

Formulation 1 (Problème de l'astronaute) : Soient deux astronautes qui atterrissent sur un corps céleste de forme sphérique. Les deux astronautes possèdent des détecteurs leur permettant de connaître leur position respective à l'intérieur d'un certain rayon de détection. Au-delà de ce rayon, les astronautes ne peuvent plus se voir. La surface du corps céleste est supposée plus grande que le rayon de détection mais suffisamment petite pour que la totalité de la surface puisse être explorée plusieurs fois. Les astronautes sont supposés dépourvus de moyens de positionnement commun leur permettant de coordonner leurs mouvements dans l'espace. D'où, ils ne peuvent pas convenir d'un point de rencontre commun. En définissant la rencontre des deux astronautes comme étant le moment où ils se trouvent à l'intérieur du rayon de détection, comment doivent-ils se déplacer afin de se rencontrer en un minimum de temps?

Formulation 2 (Problème du téléphone) : Soient deux salles séparées. Dans chaque salle se trouve n téléphones. Dans chaque salle, chaque téléphone est connecté, aléatoirement, à un téléphone de l'autre salle. À des instants discrets $t = 0, 1, 2, \dots$ un agent dans chaque salle décroche un téléphone et dit "bonjour". Comment doivent procéder les deux agents afin de minimiser le temps où ils décrochent deux téléphones connectés pour qu'ils puissent communiquer?

Dans ces deux formulations, les agents sont supposés indiscernables. En conséquence, la même stratégie doit être utilisée simultanément par les deux agents pour aboutir au rendez-vous. Plus tard, cette situation a été définie comme étant une version distincte du problème et elle a été appelée *problème symétrique du rendez-vous*. Une nouvelle version où les deux agents peuvent

utiliser des stratégies différentes a été proposée et elle a été appelée *problème asymétrique du rendez-vous* [4]. Dans la suite, un exemple d'une stratégie symétrique et d'une stratégie asymétrique sont présentés.

L'exploration aléatoire (“random walk”) Cette stratégie est un exemple d'une stratégie symétrique. Dans cette stratégie, les deux agents choisissent simultanément une direction dans l'espace de recherche de manière aléatoire, ils parcourent une certaine distance dans la direction choisie, ils s'arrêtent et ils choisissent encore une nouvelle direction. Ces étapes sont répétées jusqu'à la rencontre des deux agents.

“Attendre maman” (“wait for mummy”) Cette stratégie est une stratégie très simple et elle est communément utilisée pour les problèmes asymétriques du rendez-vous. Elle consiste à ce que l'un des agents reste immobile pendant que le deuxième explore la totalité de l'espace de recherche.

Au cours du temps, le problème du rendez-vous a connu une extension en termes de nombre d'agents. En effet, au début, les auteurs n'ont traité le problème que pour deux agents uniquement. Par la suite, le problème a été étendu au cas où plusieurs agents (plus que deux) doivent se rassembler en un lieu unique [46] [86] [125] [87]. D'une manière générale, les travaux traitant du problème du rendez-vous peuvent être divisés en deux grandes catégories. La première étudie le problème pour des espaces géométriques : une droite [7] [9] [50], un cercle [5] [70] ou le plan [10] [11]. La deuxième catégorie pose le problème pour les graphes [4] [6]. Pour les graphes, le problème du rendez-vous est défini comme suit. Soit un graphe $G = (N, A)$ avec N l'ensemble des nœuds et A l'ensemble des arcs. Initialement, les agents sont placés aléatoirement sur deux nœuds du graphe. Pour se déplacer, un agent se positionne sur un nœud adjacent (lié par un arc) au nœud dans lequel il

se trouve. Les agents se rencontrent si, après un déplacement, ils occupent simultanément le même nœud. Pour la version asymétrique du problème, il a été prouvé que la stratégie “*Attendre maman*” est optimal [12]. Pour les graphes, cette stratégie consiste à ce que l’un des agents reste immobile dans sa position initiale alors que le deuxième agent explore la totalité des nœuds du graphe. Quant à la version symétrique, aucune stratégie n’a pu être démontré comme étant optimal mais la stratégie “*Exploration aléatoire*” fournit de bons résultats. Dans cette stratégie, chaque agent choisit aléatoirement un arc sortant du nœud dans lequel il se trouve et il se place sur le nœud relié à l’autre extrémité de cet arc. Il a été prouvé qu’avec cette stratégie la rencontre des deux agents a lieu, au plus, après $\frac{16}{27}|N|^3$ déplacements [123].

1.6 Conclusion

Dans ce chapitre, les différentes notions qui vont être utilisées dans la suite ont été présentées. D’abord, les notions d’optimisation monocritère et multicritère ont été définies. Pour chaque type d’optimisation, différentes méthodes de résolution ont été présentées. Ensuite, plusieurs classes de graphes orientés et pondérés ont été définies. Ces classes se différencient par le type de pondération associée à chaque arc : déterministe ou non-déterministe et indépendante ou dépendante du temps. Pour chaque classe, la problématique du calcul du plus court chemin a été présentée. Puis, une introduction à la théorie de la décision a été réalisée. Cette introduction s’est axée sur la présentation des critères quantitatifs et probabilistes de prise de décision dans un environnement incertain. Finalement, une revue de la littérature des problèmes d’interception a été réalisée. Dans cette revue, nous nous sommes intéressé à deux problématiques particulières : “Poursuite-Évasion” et “Rendez-vous”.

Chapitre 2

Interception d'un mobile dans un graphe

2.1 Introduction

La théorie des graphes a été largement étudiée et de nombreux problèmes de la vie réelle ont été résolus grâce à elle. Une grande partie des problèmes résolus sont des problèmes de calcul d'itinéraire dans le domaine du transport. Néanmoins, pour certaines applications, aucun des algorithmes présents dans la littérature n'est approprié. Une de ces applications est le calcul d'un itinéraire pour rejoindre un véhicule se déplaçant dans un réseau routier. Ce besoin a été exprimé par les entreprises de transport public urbain qui doivent gérer des interventions sur les bus de leur flotte. En effet, parfois, le chauffeur d'un bus est confronté à un incident pendant son trajet. Cet incident peut être de nature diverse : incident technique, des passagers qui provoquent un désordre, ... Dans ce cas, une équipe d'intervention compétente doit être envoyée pour régler le problème. Si l'incident est grave, le bus est immobilisé. En conséquence, l'équipe d'intervention n'a aucun mal à calculer son itinéraire

pour le rejoindre dans les plus brefs délais. Par contre, si l'incident est mineur et ne nécessite pas l'arrêt du bus, ce dernier doit continuer son itinéraire. Effectivement, les compagnies de transport public possèdent des chartes de qualité à respecter en termes d'horaires. De plus, dans le cas où la compagnie de transport est une entreprise privée, les autorités locales infligent des pénalités financières importantes en cas de non-respect des tableaux de marche affichés dans les stations. Par suite, une première problématique consiste à calculer l'itinéraire de l'équipe d'intervention pour rejoindre le bus, qui est en mouvement, le plus rapidement possible. Il est à noter que le bus et l'équipe d'intervention sont équipés d'une balise GPS permettant d'obtenir leur position exacte en temps réel. De plus il y a généralement plusieurs équipes d'intervention disponibles et plusieurs incidents peuvent se déclarer en même temps. Par conséquent, une deuxième problématique consiste à réaliser une affectation optimale des équipes.

Dans ce chapitre, nous formalisons d'abord, le problème de l'interception pour différents cas de figure. Ensuite, nous proposons un algorithme optimal de résolution dans chacun des cas. Enfin nous vérifions l'efficacité des solutions proposées en termes de temps d'exécution par simulation.

2.2 Problème d'interception un-à-un

2.2.1 Cas des graphes statiques

2.2.1.1 Le modèle

Le réseau routier est représenté par un graphe statique déterministe¹ $G = (N, A)$ avec N l'ensemble des nœuds et A l'ensemble des arcs. Dans

1. Consulter la section 1.3.1 pour une définition formelle.

la pratique, un nœud modélise un arrêt de bus ou un croisement et un arc représente une route liant directement deux nœuds. Soit $p(u,v)$ le poids de l'arc $(u,v) \in A$. Dans la pratique, le poids d'un arc représente le temps nécessaire pour le parcourir. Cette métrique a été choisie car elle est fréquemment utilisée dans le domaine des transports et elle est simple à mesurer. Vu que dans cette partie nous traitons des graphes statiques, le poids $p(u,v)$ est indépendant du temps.

Les bus et les voitures n'utilisent pas nécessairement les mêmes voies de circulation et ne roulent pas forcément à la même vitesse. En effet, dans la majorité des réseaux routiers urbains, des voies dédiées aux bus sont aménagées afin qu'ils évitent les embouteillages. En conséquence, nous considérons deux graphes G_o et G_p . $G_o = (N_o, A_o)$ est le graphe représentant les routes sur lesquelles les bus se déplacent et $G_p = (N_p, A_p)$ celui où les voitures se déplacent. La fonction de poids associée à chaque graphe est respectivement p_o et p_p . Dans notre modèle, G_o et G_p doivent avoir des nœuds en commun mais ne partagent aucun arc. Même si les bus et les voitures utilisent la même route (où il n'existe pas de voie bus), cette route est représentée par deux arcs distincts, l'un dans G_o et l'autre dans G_p . Par conséquent, dans notre modèle, $N_o \cap N_p \neq \emptyset$ et $A_o \cap A_p = \emptyset$.

Dans la suite, le terme Mobile Objectif (MO) désigne le bus et le terme Mobile Poursuivant (MP) désigne l'équipe d'intervention. $Init_{MO} \in N_o$ et $Init_{MP} \in N_p$ sont des nœuds spéciaux représentant respectivement la position initial de MO dans G_o et MP dans G_p . La figure 2.1 présente un exemple. G_o possède neuf nœuds $N_o = \{1,2,4,5,6,7,8,9,10\}$ et G_p possède neuf nœuds $N_p = \{1,2,3,4,5,6,7,8,9\}$. L'ensemble des nœuds partagés est $N_o \cap N_p = \{1,2,4,5,6,7,8,9\}$. Dans la figure 2.1, les arcs appartenant à A_o sont représentés par des lignes discontinues et les arcs appartenant à A_p sont

représentés par des lignes continues. La position initiale de MO est le nœud 1 et la position initiale de MP est le nœud 7.

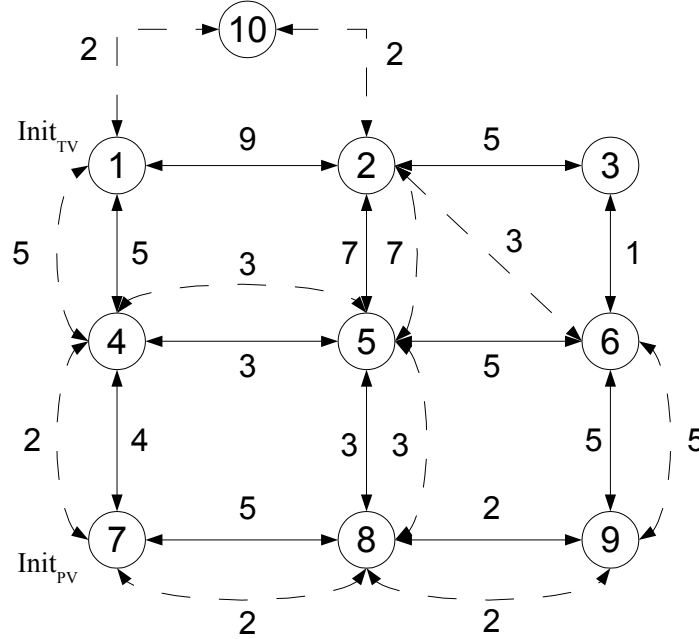


FIG. 2.1 – Exemple de graphes G_o et G_p

Dans la suite, un chemin² entre $Init_{MO}$ et $w \in N_o$ est noté $Ch_{MO}(w)$. L'ensemble de ces chemins est noté $ECh_{MO}(w)$. De même, un chemin entre $Init_{MP}$ et $r \in N_p$ est noté $Ch_{MP}(r)$. l'ensemble de ces chemins est noté $ECh_{MP}(r)$.

Hypothèse 1 MO ne peut suivre qu'un seul chemin appelé itinéraire du mobile objectif et noté $It(MO)$. Ce chemin relie la position initiale de MO ($Init_{MO}$) à sa position finale ($Dest_{MO}$). $It(MO)$ est supposé connu et fixe. De plus MO ne peut passer par le même nœud plus d'une fois.

L'hypothèse précédente est adaptée à notre contexte de travail car un bus

2. Consulter la section 1.3.1 pour une définition formelle.

possède un itinéraire connu et fixe. De plus, en général, il ne passe pas par le même croisement ou par le même arrêt plus d'une fois. Cependant, il est toujours possible de modéliser cette dernière situation. En effet, il suffit de dupliquer le nœud par lequel le bus passe plusieurs fois en dupliquant tous les arcs le reliant aux autres nœuds dans G_o et G_p . La figure 2.2 présente un exemple où le nœud 1 a été dupliqué. Supposons qu'au départ, $It(MO) = (1,4,5,2,1,10)$, en dupliquant le nœud 1, le nouvel itinéraire de MO devient le suivant $It(MO) = (1,4,5,2,1',10)$

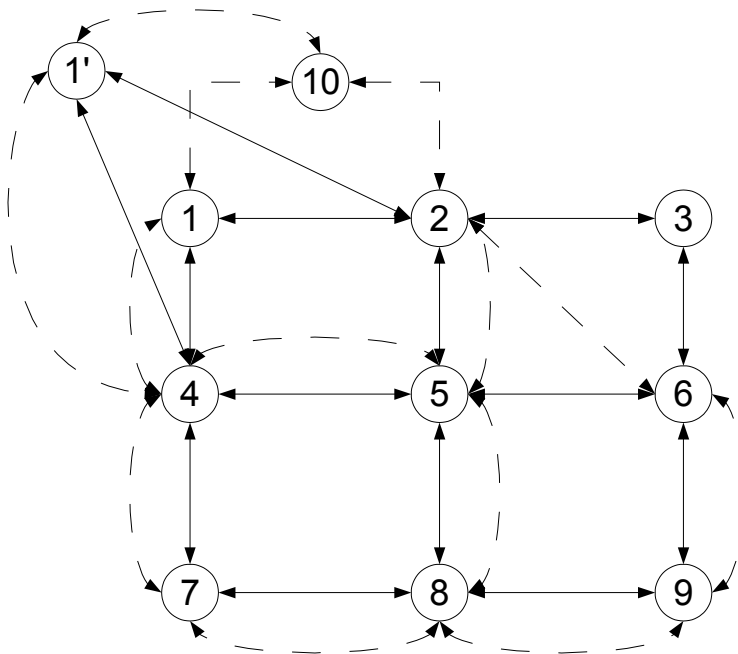


FIG. 2.2 – Exemple de la duplication d'un nœud

Définition 24 Soit $It(MO) = (u_0, \dots, u_q, \dots, u_l)$ avec $u_0 = Init_{MO}$ et $u_l = Dest_{MO}$. l'itinéraire partiel au nœud u_q est défini comme étant le chemin $It(MO, u_q) = (u_0, \dots, u_q)$.

Dans l'exemple de la figure 2.3, $Init_{MO} = 1$, $Dest_{MO} = 9$, $It(MO) =$

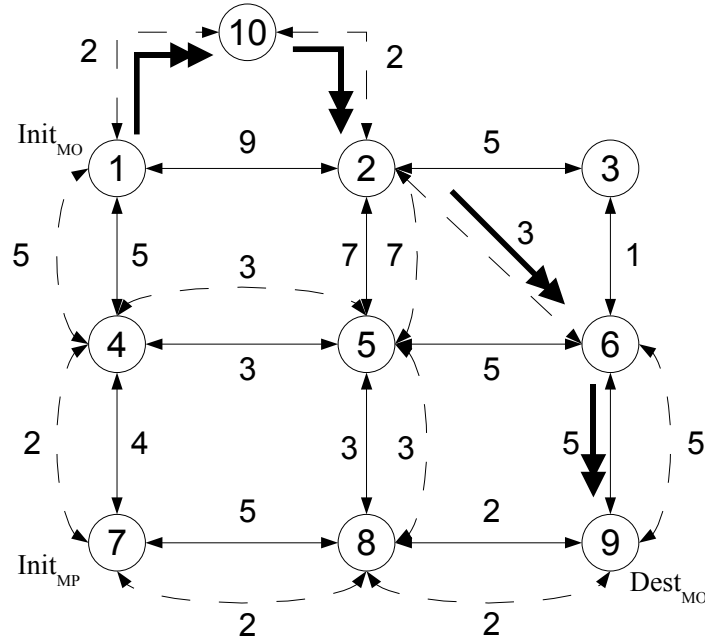


FIG. 2.3 – Exemple de graphes G_o et G_p avec un itinéraire de MO

$(1,10,2,6,9)$ et $It(MO,2) = (1,10,2)$.

Dans la suite, la fonction de coût³ associée à G_o (respectivement à G_p) est notée C_o (respectivement C_p). De plus, le plus court chemin³ de $Init_{MP}$ à $r \in N_p$ est noté $PCC_{MP}(r)$.

Définition 25 Un nœud w est appelé nœud d'interception si

1. $w \in N_o \cap N_p$ (w est un nœud partagé entre G_o et G_p);
2. $w \in It(MO)$ (w appartient à l'itinéraire de MO);
3. $C_o(It(MO,w)) \geq C_p(PCC_{MP}(w))$ (MP atteint w avant MO).

But : Trouver un nœud d'interception OPT tel que $C_o(It(MO,OPT))$ soit minimal. Un tel nœud est appelé *nœud d'interception optimal*.

3. Consulter la section 1.3.2 pour une définition formelle.

2.2.1.2 Algorithme d'interception

Une première idée pour trouver le nœud d'interception optimal consiste à calculer les plus courts chemins de la position initiale de MP à tous les autres nœuds du graphe. Il devient alors facile de trouver l'ensemble des nœuds d'interception et de choisir le meilleur. Cet algorithme est appelé algorithme de référence et il est détaillé dans l'annexe C. Néanmoins, cette démarche implique d'explorer entièrement le graphe représentant le réseau routier. Étant donné que ces graphes peuvent être d'une taille importante, cette solution est rejetée. D'où une autre approche est proposée. Cette approche consiste à adapter un algorithme de Dijkstra au cas de l'interception afin de minimiser le nombre de nœuds traités. Cet algorithme est fréquemment utilisé pour calculer le plus court chemin entre deux nœuds d'un graphe. L'idée directrice consiste à changer le nœud destination pendant le déroulement. Au début, le premier nœud de $It(MO) \cap G_p$ en partant de $Init_{MO}$ est choisi comme étant le nœud destination Obj . Pendant l'exécution, si $C_p(PCC_{MP}(Obj)) \leq C_o(It(MO), Obj)$ alors Obj est le nœud d'interception optimal. Sinon le nœud suivant de $It(MO) \cap G_p$ est choisi comme étant le nouveau nœud destination. L'algorithme est décrit dans la figure 2.4. Dans l'algorithme, l'ensemble F représente l'ensemble des nœuds avec un coût final et l'ensemble Q représente l'ensemble des nœuds candidats. Soit $It(MO) = (u_0, \dots, u_q, \dots, u_m)$, la fonction 'suivant($It(MO), u_q$)' est définie comme suit

$$suivant(It(MO), u_q) = \begin{cases} u_k & \text{Si } q < k \leq m \text{ et } u_k \in It(MO) \cap G_p \\ & \text{et } \nexists l \in [q+1, k-1] \text{ tel que} \\ & u_l \in It(MO) \cap G_p \\ \emptyset & \text{Sinon} \end{cases}$$

1	//Calculer le coût des nœuds de $It(MO)$	31	$Obj = \text{suivant}(It(MO), Obj)$
2	Soit $It(MO) = (u_0, u_1, u_2, \dots, u_m)$	32	// ÉCHEC 2
3	$\forall u_k \in It(MO)$ calculer $C_o(It(MO), u_k)$	33	Si $Obj = \emptyset$ Alors retourner ÉCHEC
4	//Initialiser les plus court chemins de G_p	34	Fin Si
5	$PCC_{MP}(Init_{MP}) = (Init_{MP})$	35	// CONDITION 3
6	$\forall r \in N_p / r \neq Init_{MP}, PCC_{MP}(r) = \emptyset$	36	Tant que $Obj \neq \emptyset$ et $Obj \in F$ faire
7	// $PCC_{MP}(r) = \emptyset \Rightarrow C_p(PCC_{MP}(r)) = \infty$	37	// CONDITION 4
8	//Initialiser l'ensemble Q	38	Si $C_p(PCC_{MP}(Obj)) \leq C_o(It(MO), Obj)$
9	$Q = \{Init_{MP}\}$	39	Alors retourner SUCCÈS
10	//Initialiser l'ensemble F	40	Sinon
11	$F = \emptyset$	41	// SUIVANT 2
12	//Initialiser le nœud destination	42	$Obj = \text{suivant}(It(MO), Obj)$
13	$Obj = Init_{MO}$	43	Fin Si
14	Si $Obj \notin It(MO) \cap G_p$ Alors	44	Fin Tant que
15	$Obj = \text{suivant}(It(MO), Obj)$	45	// ÉCHEC 3
16	//Si aucun nœuds partagés alors ÉCHEC	46	Si $Obj = \emptyset$ Alors
17	// ÉCHEC 1	47	retourner ÉCHEC
18	Si $Obj = \emptyset$ Alors retourner ÉCHEC	48	Fin Si
19	Fin Si	49	Fin Si
20	Fin Si	50	Développer les successeurs S_j de S_i
21	Tant que $Q \neq \emptyset$ faire	51	Pour chaque S_j faire
22	Choisir S_i de Q avec $C_p(PCC_{MP}(S_i))$ minimal	52	Si $C_p(PCC_{MP}(S_i)) + p_p(S_i, S_j) <$
23	$Q = Q \setminus \{S_i\}$ et $F = F \cup \{S_i\}$	53	$C_p(PCC_{MP}(S_j))$
24	// CONDITION 1	54	Alors $PCC_{MP}(S_j) = PCC_{MP}(S_i) \cup \{S_j\}$
25	Si $C_p(PCC_{MP}(S_i)) \leq C_o(It(MO), Obj)$ Alors	55	$Q = Q \cup \{S_j\}$
26	// CONDITION 2	56	Fin Si
27	Si $S_i = Obj$ Alors retourner SUCCÈS	57	Fin Pour
28	Fin Si	58	Fin Tant que
29	Sinon		
30	// SUIVANT 1		

 FIG. 2.4 – *Algorithme d'interception un-à-un pour les graphes statiques*

2.2.1.3 Optimalité de l'algorithme

Dans cette section l'optimalité de la solution fournie par notre algorithme est prouvée.

Lemme 1 *Si, dans une itération de l'algorithme, $C_p(PCC_{MP}(S_i)) > C_o(It(MO, Obj))$ alors $C_p(PCC_{MP}(Obj)) > C_o(It(MO, Obj))$.*

Preuve 1 *Ce lemme est une conséquence directe de l'algorithme de Dijkstra. En effet, dans n'importe quelle itération de l'algorithme, le nœud en cours de traitement possède un coût inférieur au coût des nœuds qui seront traités dans les itérations suivantes. Comme le nœud Obj n'a pas été encore sélectionné, alors $C_p(PCC_{MP}(Obj)) \geq C_p(PCC_{MP}(S_i)) > C_o(It(MO, Obj))$.*

Lemme 2 *Si l'algorithme traite un nœud Obj alors $\nexists w \in It(MO) \cap G_p \setminus \{Obj\}$ tel que $C_o(It(MO, w)) < C_o(It(MO, Obj))$ et $C_p(PCC_{MP}(w)) \leq C_o(It(MO, w))$. En d'autres termes, il n'existe pas un nœud d'interception w qui soit situé avant Obj dans $It(MO)$.*

Preuve 2 *Si la fonction 'suivant($It(MO), Obj$)' est appelée alors Obj courant n'est pas un nœud d'interception. En effet, si la fonction 'suivant($It(MO), Obj$)' est appelée, alors $C_p(PCC_{MP}(S_i)) > C_o(It(MO, Obj))$ (SUIVANT 1 ou 2). D'où, d'après le lemme 1, Obj n'est pas un nœud d'interception. Comme la fonction 'suivant' est appelée pour changer le nœud Obj , alors lorsque l'algorithme traite un nœud de $It(MO)$, tous les nœuds traités précédemment ne sont pas des nœuds d'interception.*

Lemme 3 *Si l'algorithme retourne SUCCÈS alors le nœud Obj est un nœud d'interception. Autrement dit, $C_p(PCC_{MP}(Obj)) \leq C_o(It(MO, Obj))$.*

Preuve 3 Si l'algorithme retourne *SUCCÈS* alors $C_p(PCC_{MP}(Obj)) \leq C_o(It(MO, Obj))$ d'après *CONDITION 1* et *2* ou *CONDITION 3* et *4*. En conséquence, *Obj* est un nœud d'interception.

Proposition 1 Si l'algorithme retourne *SUCCÈS* alors *Obj* est le nœud d'interception optimal. En d'autres termes, il n'existe pas un nœud d'interception $w \in It(MO) \cap G_p \setminus \{Obj\}$ tel que $C_o(It(MO, w)) < C_o(It(MO, Obj))$.

Preuve 4 D'après le lemme 3, si l'algorithme retourne *SUCCÈS* alors *Obj* est un nœud d'interception. De plus, d'après le lemme 2, il n'existe pas un nœud d'interception avec un coût inférieur à $C_o(It(MO, Obj))$. Par conséquent, *Obj* est le nœud d'interception optimal.

Proposition 2 Si l'algorithme retourne *ÉCHEC* alors il n'existe aucun nœud d'interception.

Preuve 5 Si l'algorithme retourne *ÉCHEC* alors soit il n'existe aucun nœud partagé entre G_o et G_p (*ÉCHEC 1*), soit il n'existe pas de nœud partagé w tel que $C_o(It(MO, w)) \geq C_p(PCC_{MP}(w))$ (*ÉCHEC 1* ou *ÉCHEC 2* avec lemme 2). En conclusion, il n'existe aucun nœud d'interception.

2.2.1.4 Complexité de l'algorithme

L'algorithme développé dans cette section résulte de l'ajout de certaines instructions à l'algorithme de Dijkstra original. Ces instructions sont localisées de la ligne 1 à 3, de la ligne 12 à 20 et de la ligne 25 à 49. La majorité des instructions ajoutées sont des instructions conditionnelles « SI » qui n'ont aucun effet sur la complexité. Cependant, il existe deux boucles. La première se situe à la ligne 3 et la deuxième se situe de la ligne 36 à 44. La boucle à la

ligne 3 calcule le temps d'arrivée pour tous les nœuds de $It(MO)$. D'où, au pire des cas, cette boucle réalise $|N_o|$ itérations. Quant à la boucle localisée de la ligne 36 à 44, elle n'a aucune influence sur la complexité de la boucle principale de la ligne 21 à 58. En effet, cette boucle permet de changer le nœud destination Obj grâce à la fonction "suivant($It(MO), u_q$)". Cette fonction permet d'obtenir le nœud suivant de l'ensemble $It(MO) \cap G_p$. Puisque $It(MO) \cap G_p$ contient au plus $|N_p|$ nœuds, le résultat de cette fonction est \emptyset après, au plus, $|N_p|$ appels. En conséquence, la boucle de la ligne 36 à 44 peut réaliser, au plus, $|N_p|$ itérations pour toute l'exécution de l'algorithme indépendamment de la boucle principale. La complexité de cette dernière est $O(|A_p| + |N_p|.ln(N_p))$ selon [17]. En conséquence, la complexité de la totalité de l'algorithme est $O(|N_o| + |A_p| + |N_p|.ln(N_p))$. D'où en considérant $|N_o| \leq |N_p|$, la complexité devient $O(|A_p| + |N_p|.ln(N_p))$.

2.2.1.5 Implémentation de l'algorithme

La méthode Waxman Waxman [127] a proposé une méthode pour générer aléatoirement des graphes. Cette méthode est devenue un standard *de facto*. Elle nécessite deux étapes. La première consiste à distribuer N nœuds dans un espace rectangulaire. Chaque nœud est placé dans une position avec des coordonnées entières. La deuxième étape consiste à créer des arcs entre les différents nœuds. Soient deux nœuds p et q , d la distance euclidienne entre p et q et L la distance euclidienne maximale entre deux nœuds du graphe. Un arc est créé entre p et q suivant la loi de probabilité

$$P = \beta \exp\left(\frac{-d}{L\alpha}\right)$$

α et β sont des paramètres entre 0 et 1 permettant d'influer sur la probabilité de création des arcs. Plus la valeur de β augmente, plus la densité des arcs

augmente. Plus α diminue, plus la densité des arcs courts augmente par rapport à celle des arcs longs. Cette procédure de création d'arcs est itérée jusqu'à ce que le graphe devienne complètement connexe.

Procédure de test

Tests et résultats L'algorithme présenté dans la section 2.2.1.2 est implémenté en JAVA afin d'évaluer son temps d'exécution. Pour avoir un point de comparaison, l'algorithme de référence détaillé dans l'annexe C est aussi implémenté. La figure 2.5 présente une capture d'écran du logiciel résultant. Dans les différents tests, G_o et G_p sont égaux. Cette propriété est considérée dans le but de maximiser le nombre de nœuds partagés et de rendre ainsi les calculs plus longs. Pour les tests, plusieurs graphes de tailles différentes sont générés aléatoirement. Un graphe est généré de la manière suivante. La structure du graphe est générée grâce à la méthode Waxman avec $\alpha = 0.15$ et $\beta = 0.2$. Puis, un poids aléatoire est assigné à chacun de ses arcs. Pour chaque graphe, plusieurs tests sont réalisés en générant aléatoirement, à chaque fois, $Init_{MP}$ et $It(MO)$. Tous les tests sont réalisés sur un PC équipé d'un processeur Intel Core Duo de fréquence 1.83 Ghz. Le temps d'exécution moyen pour chaque graphe est donné dans le tableau 2.1. L'ensemble des résultats est représenté dans la figure 2.6.

Interprétation des résultats D'après la figure 2.6, nous remarquons d'abord que notre algorithme d'interception est plus rapide que l'algorithme de référence et que la différence dans les temps d'exécution devient plus importante à mesure que la taille du graphe devient plus importante. Par conséquent, notre algorithme apporte une réelle plus value en termes de temps d'exécution. De plus, l'algorithme d'interception est très rapide et peut être

utilisé dans des applications à fortes contraintes de temps d'exécution. Effectivement, il ne nécessite qu'une dizaine de millisecondes pour s'exécuter même pour des graphes taille importante. Par exemple, dans nos tests, il s'exécute en 41 ms pour un graphe contenant 5000 nœuds. Enfin, notre algorithme ne pose pas de problème de scalabilité. En effet, la figure 2.6 montre que l'évolution du temps d'exécution est pratiquement linéaire. Pour vérifier ceci, intéressons-nous aux mesures recueillies dans le tableau 2.1. Les temps d'exécution moyens pour un graphe contenant 500 nœuds et 5000 nœuds sont respectivement 3 ms et 41 ms. En conséquence, en considérant un graphe dix fois plus grand, le temps d'exécution est multiplié par dix. Cette proportionnalité peut être observée pour toutes les mesures du tableau 2.1. Par suite, la courbe d'évolution du temps d'exécution en fonction du nombre de nœuds du graphe pourrait être aisément approximée par une fonction linéaire de la forme $f(x) = ax + b$. D'où, l'évolution pratique du temps d'exécution est meilleure que la complexité théorique calculée dans la section précédente et qui est de $O(|A_p| + |N_p|.ln(N_p))$.

2.2.1.6 Comparaison avec une approche par poursuite

Une autre approche pour intercepter le mobile objectif consiste à le poursuivre jusqu'à l'atteindre. L'idée générale consiste à tenter de minimiser constamment la distance séparant l'objectif et le poursuivant. Dans un graphe, ce comportement peut se traduire par l'algorithme décrit dans la figure 2.7. Dans ce dernier, $G = (N, A)$ est un graphe dans lequel se trouvent le poursuivant et l'objectif, Pos_p est le nœud dans lequel se trouve le poursuivant et Pos_o est le nœud dans lequel se trouve l'objectif. Cette approche a été implémentée afin de la comparer avec l'algorithme d'interception développé dans ce chapitre. Le but de cette comparaison est de déterminer l'algorithme

Nombre de nœuds	Algorithme d'interception (ms)	Algorithme de référence (ms)
500	3	4
1000	6	10
1500	9	20
2000	11	31
2500	15	39
3000	23	57
3500	22	67
4000	25	96
4500	30	119
5000	41	145

TAB. 2.1 – Temps d'exécution moyen en fonction du nombre de nœuds de l'algorithme d'interception et de l'algorithme de référence

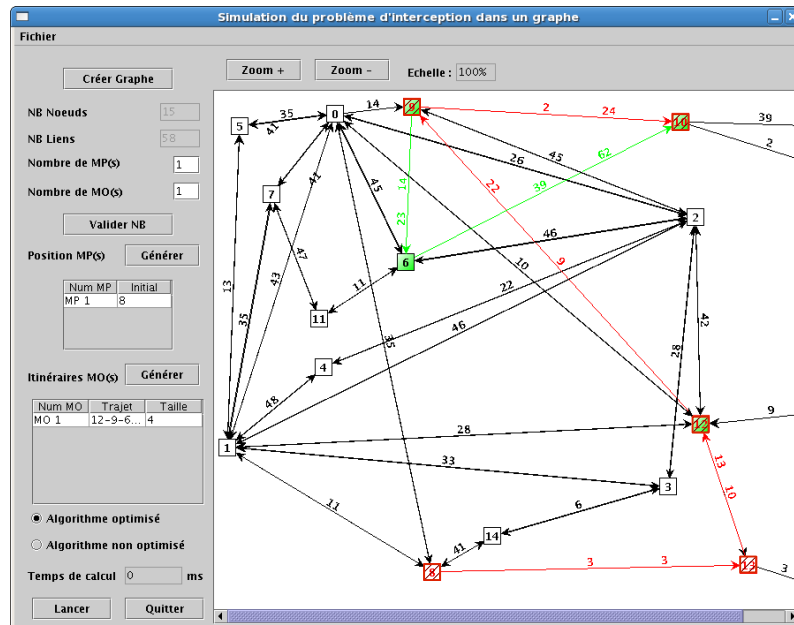


FIG. 2.5 – Logiciel de simulation

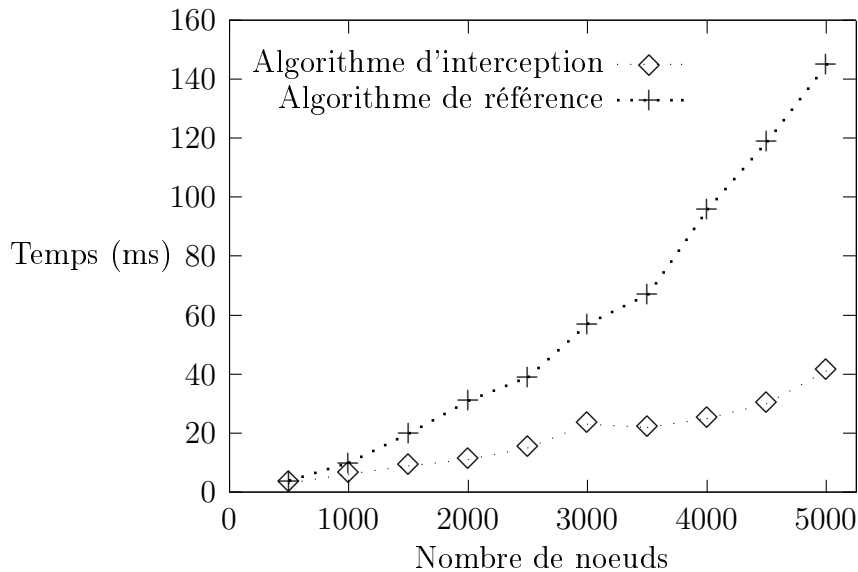


FIG. 2.6 – Évolution du temps d'exécution moyen de l'algorithme d'interception et de l'algorithme de référence en fonction du nombre de nœuds

le plus adéquat à notre problématique.

Tests et résultats L'algorithme de poursuite présenté précédemment a été implémenté en JAVA. Pour les tests, plusieurs graphes aléatoires sont générés en utilisant la même technique que dans la section 2.2.1.5. Il est rappelé que G_o et G_p sont égaux pour tous les tests. Pour chaque graphe, plusieurs tests sont réalisés avec, à chaque fois, un $Init_{MP}$ et un $It(MO)$ générés aléatoirement. Pour comparer l'efficacité des deux approches, deux critères sont considérés. Le premier critère est le pourcentage d'interceptions réussies. Une interception réussie a lieu si le poursuivant atteint l'objectif avant que ce dernier n'atteigne le dernier nœud de son itinéraire ($Dest_{MO}$). Sinon, l'interception est un échec. Le deuxième critère considéré est le pourcentage d'interceptions optimales. Une interception est optimale si le poursuivant intercepte l'objectif dans le nœud d'interception optimale. Les résultats de

- 1 Déterminer Pos_o
- 2 Calculer le plus court chemin entre Pos_p et Pos_o
- 3 Placer le poursuivant sur le nœud suivant du plus court chemin calculé
- 4 Si $Pos_p = Pos_o$ Alors retourner SUCCÈS
- 5 Sinon retour à l'étape 1

FIG. 2.7 – *Algorithme de poursuite*

ces tests sont présentés dans le tableau 2.2. Il est à noter que dans tous les tests réalisés il existe au moins un nœud d'interception.

Interprétation des résultats Les résultats du tableau 2.2 montrent que l'approche par poursuite échoue dans 35% des cas à intercepter l'objectif bien que l'interception est possible. Ce résultat montre que l'approche par interception calcule l'itinéraire de manière plus "intelligente" que l'approche par poursuite. En effet, l'algorithme d'interception utilise une information fondamentale que l'algorithme de poursuite n'utilise pas. Cette information est la connaissance a priori de l'itinéraire du *MO*. De plus, même si l'objectif est intercepté, l'interception n'est optimale que dans 5% des cas. Ce résultat montre encore que le fait de ne pas utiliser la connaissance a priori de l'itinéraire du *MO* conduit à des résultats qui ne sont pas optimaux. En conclusion, l'approche par poursuite n'est pas appropriée pour l'interception dans les graphes statiques.

	Approche par interception	Approche par poursuite
Interceptions réussies	100%	65%
Interceptions optimales	100%	5%

TAB. 2.2 – *Comparaison entre l'approche par interception et l'approche par poursuite*

2.2.2 Cas des graphes FIFO

2.2.2.1 Le modèle

Le modèle présenté dans la section 2.2.1.1 est étendu pour les graphes FIFO⁴. La condition FIFO ($\forall t, t' \geq 0, t \leq t' \implies t + p(i, j, t) \leq t' + p(i, j, t')$) n'est pas toujours vérifiée pour les réseaux routiers. Néanmoins, elle peut être vraie dans de nombreux cas spécialement quand les routes sont congestionnées. C'est pourquoi de tels graphes peuvent être utilisés pour représenter les réseaux routiers. Donc, G_o et G_p sont considérés maintenant comme des graphes FIFO. La fonction de poids associée à chaque graphe est maintenant dépendante du temps. Les fonctions de poids sont $p_o(u, v, t)$ pour G_o et $p_p(u, v, t)$ pour G_p . Comme pour le modèle précédent, G_o et G_p doivent partager certains nœuds mais ne doivent partager aucun arc.

Hypothèse 2 *Les fonctions p_o et p_p sont supposées connues par avance pour toute la période d'étude.*

L'hypothèse précédente est nécessaire afin de réaliser une solution qui fonctionne hors-ligne.

Hypothèse 3 *MO ne peut suivre qu'un seul chemin appelé itinéraire du mobile objectif et noté $It(MO)$. Ce chemin relie la position initiale de MO*

4. Consulter la section 1.3.4.2 pour une définition formelle.

($Init_{MO}$) à sa position finale ($Dest_{MO}$). $It(MO)$ est supposé connu et fixe. De plus MO ne peut passer par le même nœud plus d'une fois.

Dans la suite, la fonction de coût dynamique⁵ associée à G_o (respectivement à G_p) est notée CD_o (respectivement CD_p). De plus, le plus court chemin⁵ de $Init_{MP}$ à $r \in N_p$ en partant de $Init_{MP}$ à t_0 est noté $PCC_{MP}^{t_0}(r)$.

Définition 26 Un nœud w est appelé nœud d'interception à la date de départ t_0 , si $w \in N_o \cap N_p$, $w \in It(MO)$ et $CD_p^{t_0}(PCC_{MP}^{t_0}(w)) \leq CD_o^{t_0}(It(MO, w))$. Ceci implique que w est un nœud partagé entre G_o et G_p , qu'il appartient à l'itinéraire du mobile objectif et que MP atteint w avant MO quand les deux quittent leur position initiale à l'instant t_0 .

But : Trouver un nœud d'interception OPT à la date de départ t_0 tel que $CD_o^{t_0}(It(MO, OPT))$ est minimal. Un tel nœud est appelé *nœud d'interception optimal* à la date de départ t_0 .

2.2.2.2 Algorithme d'interception

Notre solution pour l'interception dans un graphe FIFO consiste à adapter l'algorithme d'interception pour les graphes statiques présenté à la section 2.2.1.2. En effet, il a été prouvé que n'importe quel algorithme de calcul de plus court chemin dans les graphes statiques peut être étendu aux graphes FIFO avec la même complexité [2] [78]. D'où, pour traiter le cas des graphes FIFO, l'algorithme de la section 2.2.1.2 est réutilisé en remplaçant la fonction de coût et la fonction de poids d'un arc. Les fonctions de coût C_o et C_p sont remplacées respectivement par $CD_o^{t_0}$ et $CD_p^{t_0}$ avec t_0 la date de départ. Les fonctions P_o et P_p sont remplacées respectivement par p_o et p_p . L'algorithme est décrit dans la figure 2.8.

5. Consulter la section 1.3.4 pour une définition formelle.

1	//Calculer le coût des nœuds de $It(MO)$	31	$Obj = \text{suivant}(It(MO), Obj)$
2	Soit $It(MO) = (u_0, u_1, u_2, \dots, u_m)$	32	// ÉCHEC 2
3	$\forall u_k \in It(MO)$ calculer $CD_o^{t_0}(It(MO, u_k))$	33	Si $Obj = \emptyset$ Alors retourner ÉCHEC
4	//Initialiser les plus court chemins de G_p	34	Fin Si
5	$PCC_{MP}^{t_0}(Init_{MP}) = (Init_{MP})$	35	// CONDITION 3
6	$\forall r \in N_p / r \neq Init_{MP}, PCC_{MP}^{t_0}(r) = \emptyset$	36	Tant que $Obj \neq \emptyset$ et $Obj \in F$ faire
7	// $PCC_{MP}^{t_0}(r) = \emptyset \Rightarrow CD_p^{t_0}(PCC_{MP}^{t_0}(r)) = \infty$	37	// CONDITION 4
8	//Initialiser l'ensemble Q	38	Si $CD_p^{t_0}(PCC_{MP}^{t_0}(Obj)) \leq CD_o^{t_0}(It(MO, Obj))$
9	$Q = \{Init_{MP}\}$	39	Alors retourner SUCCÈS
10	//Initialiser l'ensemble F	40	Sinon
11	$F = \emptyset$	41	// SUIVANT 2
12	//Initialiser le nœud destination	42	$Obj = \text{suivant}(It(MO), Obj)$
13	$Obj = Init_{MO}$	43	Fin Si
14	Si $Obj \notin It(MO) \cap G_p$ Alors	44	Fin Tant que
15	$Obj = \text{suivant}(It(MO), Obj)$	45	// ÉCHEC 3
16	//Si aucun nœuds partagés alors ÉCHEC	46	Si $Obj = \emptyset$ Alors
17	// ÉCHEC 1	47	retourner ÉCHEC
18	Si $Obj = \emptyset$ Alors retourner ÉCHEC	48	Fin Si
19	Fin Si	49	Fin Si
20	Fin Si	50	Développer les successeurs S_j de S_i
21	Tant que $Q \neq \emptyset$ faire	51	Pour chaque S_j faire
22	Choisir S_i de Q avec $CD_p^{t_0}(PCC_{MP}^{t_0}(S_i))$ minimal	52	Si $CD_p^{t_0}(PCC_{MP}^{t_0}(S_i)) +$
23	$Q = Q \setminus \{S_i\}$ et $F = F \cup \{S_i\}$	53	$p_p(S_i, S_j, CD_p^{t_0}(PCC_{MP}^{t_0}(S_i))) < CD_p^{t_0}(PCC_{MP}^{t_0}(S_j))$
24	// CONDITION 1	54	Alors $PCC_{MP}^{t_0}(S_j) = PCC_{MP}^{t_0}(S_i) \cup \{S_j\}$
25	Si $CD_p^{t_0}(PCC_{MP}^{t_0}(S_i)) \leq CD_o^{t_0}(It(MO, Obj))$ Alors	55	$Q = Q \cup \{S_j\}$
26	// CONDITION 2	56	Fin Si
27	Si $S_i = Obj$ Alors retourner SUCCÈS	57	Fin Pour
28	Fin Si	58	Fin Tant que
29	Sinon		
30	// SUIVANT 1		

 FIG. 2.8 – *Algorithme d'interception un-à-un pour les graphes FIFO*

2.2.2.3 Optimalité et complexité de l'algorithme

D'après [2] et [78], l'algorithme de Dijkstra est valide pour les graphes FIFO. Par suite, les propositions et les lemmes présentés dans la section 2.2.1.3 peuvent être facilement prouvés pour les graphes FIFO. La complexité de l'algorithme de la section 2.2.1.2 qui a été calculé dans la section 2.2.1.4 est de $\mathcal{O}(|A_p| + |N_p|.ln(|N_p|))$. Comme l'algorithme d'interception pour les graphes FIFO n'ajoute aucune instruction supplémentaire, la complexité reste la même que celle dans le cas des graphes statiques.

2.2.2.4 Implémentation de l'algorithme

Tests et résultats L'algorithme d'interception pour les graphes FIFO présenté dans la section 2.2.2.2 est implémenté en JAVA et il a été testé de la même manière que l'algorithme d'interception pour les graphes statiques. La procédure de test présentée dans la section 2.2.1.5 est suivie : G_o et G_p sont égaux et différents graphes FIFO de tailles différentes sont générés aléatoirement. La structure du graphe est générée en utilisant la méthode Waxman avec $\alpha = 0.15$ et $\beta = 0.2$. La fonction de poids de chaque arc est générée aléatoirement de sorte que la condition FIFO soit respectée. Pour chaque graphe, plusieurs tests sont réalisés en générant aléatoirement, à chaque fois, $Init_{MP}$ et $It(MO)$. Tous les tests sont réalisés sur un PC équipé d'un processeur Intel Core Duo de fréquence 1.83 Ghz. Les résultats de ces tests sont fournis dans le tableau 2.3. La courbe de la figure 2.9 représente la variation du temps d'exécution moyen en fonction du nombre de nœuds de l'algorithme d'interception pour les graphes statiques et celui pour les graphes FIFO. Évidemment, les graphes statiques et les graphes FIFO sont différents et ils sont générés aléatoirement pour chaque jeu de test.

Interprétation des résultats La courbe 2.9 montre que les deux algorithmes possèdent pratiquement le même temps d'exécution. Par conséquent, l'algorithme d'interception pour les graphes FIFO possède les mêmes propriétés de rapidité et de scalabilité que l'algorithme d'interception pour les graphes statiques.

Nombre de nœuds	Graphes FIFO (ms)	Graphes Statiques (ms)
500	5	3
1000	7	6
1500	9	9
2000	12	11
2500	14	15
3000	18	23
3500	20	22
4000	23	25
4500	26	30
5000	33	41

TAB. 2.3 – *Temps d'exécution moyen de l'algorithme d'interception en fonction du nombre de nœuds pour les graphes FIFO et les graphes statiques*

2.3 Problème d'interception plusieurs-à-plusieurs

2.3.1 Cas des graphes statiques

2.3.1.1 Le modèle

Le même modèle que celui dans la section 2.2.1.1 est utilisé excepté que maintenant il existe un ensemble de K mobiles poursuivants $\{MP_1, MP_2, \dots, MP_K\}$

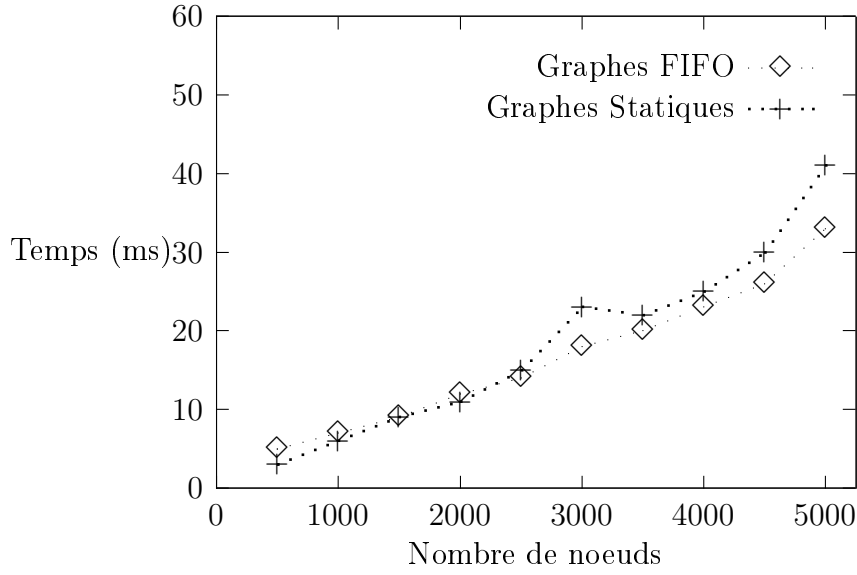
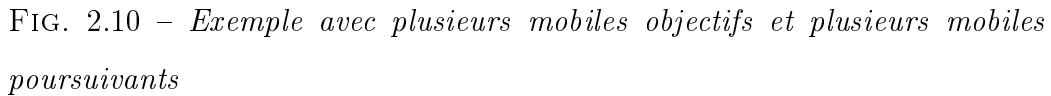


FIG. 2.9 – Évolution du temps d'exécution moyen de l'algorithme d'interception en fonction du nombre de nœuds pour les graphes FIFO et les graphes statiques

dans G_p et un ensemble de L mobiles objectifs $\{MO_1, MO_2, \dots, MO_L\}$ dans G_t . Un exemple est présenté dans la figure 2.10.

Les notations utilisées pour l'interception un-à-un sont étendues à l'interception plusieurs-à-plusieurs comme suit.

- $OPT(MP_i, MO_j)$ représente le nœud d'interception optimale de MO_j par MP_i ;
- $It(MO_j)$ représente l'itinéraire de MO_j ;
- $It(MO_j, w)$ avec $w \in N_t$ représente l'itinéraire partiel de MO_j jusqu'au nœud w ;
- $Init_{MO_j}$ représente le nœud initial de MO_j ;
- $Dest_{MO_j}$ représente le nœud final de MO_j ;
- $Init_{MP_i}$ représente le nœud initial de MP_i .



Définition 27 La fonction de coût *PairC* est une fonction attribuant un coût à la paire (MP_i, MO_j) . Ce coût représente le coût de la poursuite de MO_j par MP_i .

Définition 29 Soit $CG(aff_m)$ une fonction de coût global. $CG(aff_m)$ re-

présente la fonction de coût global résultante de l'affectation de l'ensemble des MP par aff_m . Elle est définie comme suit :

$$CG(aff_m) = \sum_{MP_i \text{ affectés}} PairC(MP_i, aff_m(MP_i))$$

But : Trouver la fonction d'affectation optimale aff_{opt} qui optimise CG .

Pour atteindre notre but, la définition de $PairC$ est un point important à traiter. En effet, cette fonction est l'élément fondamental pour l'évaluation d'une fonction d'affectation. Elle permet aussi de définir le critère d'optimisation dans le modèle. Par exemple, si le but est de maximiser uniquement le nombre d'interception, $PairC$ peut être définie comme suit :

$$PairC(MP_i, MO_j) = \begin{cases} 1 & \text{Si interception} \\ 0 & \text{Sinon} \end{cases}$$

Avec une telle définition, CG doit être maximisée. Cependant, Si le but est de minimiser le temps d'interception moyen, $PairC$ peut être définie comme suit :

$$PairC(MP_i, MO_j) = \begin{cases} C_t(It(MO_j, OPT(MP_i, MO_j))) & \text{Si interception} \\ \infty & \text{Sinon} \end{cases}$$

Dans ce cas, CG doit être minimisée.

2.3.1.2 Algorithme d'affectation

Une solution évidente peut consister à essayer toutes les paires possibles de MP et MO . Mais la complexité d'une telle approche est exponentielle. En conséquence, cette méthode est rejetée.

Cas où $K=L$ Dans ce cas, notre solution se présente en deux étapes. Dans la première étape, nous calculons $PairC(MP_i, MO_j) \forall i \in [1, K]$ et $\forall j \in [1, L]$ en utilisant l'algorithme de la section 2.2.1.2. Ensuite, les résultats sont représentés par une matrice carrée appelée *matrice des coûts*. Des exemples de cette matrice sont présentés dans la figure 2.11 et la figure 2.12. La figure 2.11 montre un exemple où le but est de maximiser uniquement le nombre d'interceptions réussies. La figure 2.12 montre un exemple où le but est de minimiser le temps moyen d'interception. Ayant cette matrice, notre problème devient un Problème d'Affectation de Tâches ("Task Assignment Problem"). La deuxième étape consiste à résoudre le PAT. Il existe différents algorithmes pour résoudre le PAT. Par exemple, *l'algorithme de Kuhn-Munkres* appelé aussi *algorithme hongrois* [82] [83] permet de résoudre le problème d'affectation en un temps polynomial. Il faut noter que la convergence de cet algorithme n'est pas garantie avec des coûts infinis. Néanmoins, ce problème de convergence peut être facilement contourné en fixant une grande valeur, déterminée suivant le contexte, comme étant l'infini.

	MO₁	MO₂	MO₃
MP₁	1	0	1
MP₂	0	1	1
MP₃	1	0	0

FIG. 2.11 – *Premier exemple d'une matrice de coût où $K = L$*

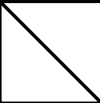
	MO₁	MO₂	MO₃
MP₁	12	∞	15
MP₂	∞	7	10
MP₃	20	∞	∞

FIG. 2.12 – Deuxième exemple d'une matrice de coût où $K = L$

Cas où $K \neq L$ Dans ce cas, une étape supplémentaire est nécessaire pour pouvoir utiliser l'algorithme du PAT. Cette étape consiste à ajouter des MP ou des MO virtuels pour obtenir le même nombre de MP et de MO et revenir ainsi au premier cas. Les mobiles ajoutés ne doivent en aucun cas influencer le résultat final de l'algorithme. C'est pourquoi, les MO virtuels sont des mobiles qui ne peuvent pas être interceptés et les MP virtuels sont des mobiles qui ne peuvent intercepter aucun MO . Les figures 2.13 et 2.14 présentent des exemples avec des MO virtuels et des MP virtuels.

2.3.1.3 Complexité de l'algorithme

Cas où $K=L$ Dans la première étape, l'algorithme d'interception est exécuté pour chaque paire (MP_i, MO_j) . La complexité de cette étape est $\mathcal{O}(K.L.(|A_p| + |N_p|.ln(|N_p|))) = \mathcal{O}(K^2.(|A_p| + |N_p|.ln(|N_p|)))$. Pour la deuxième étape, le PAT peut être résolu avec l'algorithme de Kuhn-Munkres. La complexité de cet algorithme est $\mathcal{O}(K^3)$. Donc la complexité globale est $\mathcal{O}(K^2.(|A_p| + |N_p|.ln(|N_p|)) + K^3)$. Par suite, la complexité en fonction du nombre de nœuds est $\mathcal{O}(|A_p| + |N_p|.ln(|N_p|))$. Quant à la complexité en fonction du nombre de

	MO₁	MO₂	MO_{virtuel}
MP₁	1	0	0
MP₂	0	1	0
MP₃	1	0	0

FIG. 2.13 – *Exemple d'une matrice de coût avec un MO virtuel*

	MO₁	MO₂	MO₃
MP₁	1	0	1
MP₂	0	1	1
MP_{virtuel}	0	0	0

FIG. 2.14 – *Exemple d'une matrice de coût avec un MP virtuel*

MP et MO , elle est égale à $\mathcal{O}(K^2 + K^3) = \mathcal{O}(K^3)$.

Cas où $K \neq L$ Pour ce cas, les seuls opérations supplémentaires sont l'ajout des MO ou MP virtuels. Ces opérations s'exécutent en $\mathcal{O}(L)$ ou $\mathcal{O}(K)$. Par conséquent, la complexité en fonction du nombre des nœuds reste $\mathcal{O}(|A_p| + |N_p|.ln(|N_p|))$ et la complexité en fonction du nombre de MP et MO reste $\mathcal{O}(K^3)$.

2.3.1.4 Implémentation de l'algorithme

Tests et résultats L'algorithme développé dans cette section est implémenté en JAVA. Dans les différents tests, G_t et G_p sont égaux. Tous les tests de cette section sont réalisés sur un graphe ayant 3000 nœuds et 12000 arcs. La structure du graphe est générée aléatoirement avec la méthode Waxman avec $\alpha = 0.15$ et $\beta = 0.2$ et un poids aléatoire est affecté à chaque arc. Lors des tests, l'accent est mis sur l'étude de l'évolution du temps d'exécution en fonction du nombre de MP/MO . En effet, la complexité en fonction de la taille du graphe est la même que celle de l'algorithme présenté à la section 2.2.1.2. De plus, l'étude du temps d'exécution en fonction de la taille du graphe a été déjà traitée dans la section 2.2.1.5. Pour un nombre donné de MP/MO , plusieurs tests sont réalisés en générant aléatoirement la position initiale de chaque mobile poursuivant et l'itinéraire de chaque mobile objectif. Pour tous les tests, le nombre des mobiles objectifs est égal à celui des mobiles poursuivants. En effet, la phase d'ajout de MO ou de MP virtuels n'a pas d'impact significatif sur le temps d'exécution donc il est inutile de considérer le cas où $K \neq L$. Tous les tests sont réalisés sur un PC équipé d'un processeur Intel Core Duo de fréquence 1.83 Ghz. Les résultats des tests sont reportés dans le tableau 2.4.

Interprétation des résultats Les résultats du tableau 2.4 montrent l'efficacité de l'algorithme en termes de temps de calcul. En effet, il nécessite moins de 5 secondes pour trouver l'affectation optimale pour 20 *MP* et 20 *MO*. De plus, la courbe 2.15 permet de vérifier que l'évolution du temps de calcul est une évolution polynomiale suivant une courbe $f(x) = x^3$. Par suite, notre algorithme ne souffre pas de problème de scalabilité.

Nombre de poursuivants/objectifs	Temps moyen d'exécution (ms)
1	22
5	304
10	1149
15	2730
20	4698

TAB. 2.4 – *Temps d'exécution moyen en fonction du nombre de MP/MO*

2.3.2 Cas des graphes FIFO

Ce cas est trivial. En effet, le même modèle utilisé dans le cas de l'interception un-à-un pour les graphes FIFO peut être réutilisé en considérant plusieurs *MO* et plusieurs *MP*. Concernant la solution, l'approche utilisée pour l'interception plusieurs-à-plusieurs pour les graphes statiques peut être intégralement utilisée. En effet, pour obtenir la matrice des coûts, il suffit d'utiliser l'algorithme d'interception pour les graphes FIFO développé dans la section 2.2.2.2. Par la suite, il suffit de résoudre le PAT de la même manière que pour les graphes statiques.

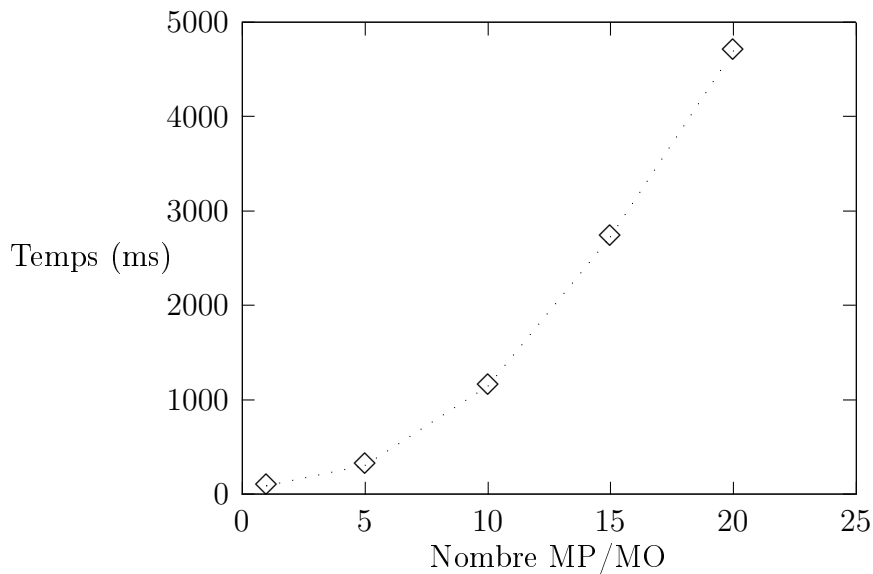


FIG. 2.15 – *Évolution du temps d'exécution moyen en fonction du nombre de MP/MO*

2.4 Conclusion

Dans ce chapitre, le problème de l'interception d'un mobile dans un graphe a été formalisé pour plusieurs cas de figures : un-à-un, plusieurs-à-plusieurs et pour différents types de graphes : statiques, FIFO. Pour chaque cas, un algorithme de résolution a été proposé et l'optimalité de la solution retournée a été prouvée. De plus, l'efficacité des algorithmes en termes de temps de calcul a été vérifiée par simulation.

Les algorithmes développés peuvent parfaitement être utilisés dans un contexte industriel vu leur simplicité et leur efficacité en termes de temps de calcul. Par conséquent, ils peuvent apporter une réponse concrète au problème de la gestion des interventions sur les bus en mouvement.

La principale limitation des modèles et des solutions proposés est le fait que les temps de parcours au niveau des arcs sont supposés connus à l'avance avec

précision. En effet, en général, des prédictions sont réalisées sur les temps de parcours au niveau des différentes routes. Mais toute prédiction induit une erreur dont il faut tenir compte. D'où, il est nécessaire de développer des modèles incluant ce type d'incertitude.

Chapitre 3

Problème du plus court chemin dans un graphe dynamique avec intervalles

3.1 Introduction

Le problème du plus court chemin a fait l'objet d'un grand nombre de travaux [106] [38]. Dans la majorité des cas, le problème est considéré comme étant déterministe. En effet, dans ces travaux, les poids des arcs sont supposés être déterministes. Or, dans de nombreux cas réels, il existe une forte incertitude sur les poids. En effet, considérons un réseau routier sur lequel des prédictions sur les temps de parcours ont été réalisées. Étant donné que chaque prédiction implique une erreur, il est plus pertinent de modéliser le poids de chaque arc par un intervalle. Par exemple, supposons que le résultat de la prédiction du poids d'un arc est égal à 10 avec une erreur de 10%. Par conséquent, le poids de l'arc en question appartiendra à l'intervalle $[9,11]$. Une telle situation où l'état du système est incertain n'est pas spécifique aux

réseaux routiers. Au contraire, l'optimisation sous incertitude est un sujet d'actualité surtout dans le domaine de la logistique [105] [42] [43]. Une approche intégrant l'incertitude sur les poids des arcs a déjà été proposée dans la littérature [77] [40]. Les auteurs définissent un graphe où les poids des arcs sont des intervalles. Pour ce type de graphe, le problème du plus court chemin de déviation robuste et le problème du plus court chemin robuste absolu ont été définis. Depuis, tous les travaux qui ont adopté cette approche supposent que le poids de l'arc est indépendant du temps.

Dans ce chapitre, nous nous proposons d'étendre la définition des graphes avec intervalles au cas dynamique. Effectivement, nous définissons des graphes où le poids d'un arc est un intervalle qui dépend du temps. Nous appelons ce type de graphe "les graphes dynamiques avec intervalles". Par opposition, les graphes définis dans la littérature sont appelés "graphes statiques avec intervalles". Pour les graphes dynamiques avec intervalles, nous définissons, d'abord, deux types de problèmes du plus court chemin. Le premier est le problème du plus court chemin de déviation robuste. Le deuxième est le problème du plus court chemin suivant un critère d'optimisme. Ensuite, une étude sur la difficulté de ces deux problèmes est réalisée. Puis, nous traitons le problème du plus court chemin suivant un critère d'optimisme. Enfin, différentes extensions possibles sont proposées.

3.2 Graphe dynamique avec intervalles

3.2.1 Définition générale

Soit un graphe dynamique¹ $G = (N, A)$. Pour chaque intervalle de temps, le poids de chaque arc est supposé connu avec incertitude. Néanmoins, l'inter-

1. Consulter la section 1.3.4 pour une définition formelle.

valle dans lequel il varie est supposé connu. Par suite, pour chaque intervalle de temps, à un arc du graphe $(i,j) \in A$ est associé un intervalle du type $[a_{ij}^t, b_{ij}^t]$ avec $0 < a_{ij}^t \leq b_{ij}^t$. Donc, à l'instant t , le poids de l'arc (i,j) , noté $p(i,j,t)$, est compris entre a_{ij}^t et b_{ij}^t . Un exemple d'un tel graphe est présenté dans la figure 3.1. Dans cet exemple, trois intervalles de temps sont considérés avec $T = 5$ unités de temps la durée d'un intervalle de temps. Ainsi, le poids de l'arc $(1,2)$ est compris dans l'intervalle $[1,2]$ pour $\forall t \in [0,5[$, dans l'intervalle $[2,5]$ pour $\forall t \in [5,10[$ et dans l'intervalle $[3,6]$ pour $\forall t \in [10,15[$.

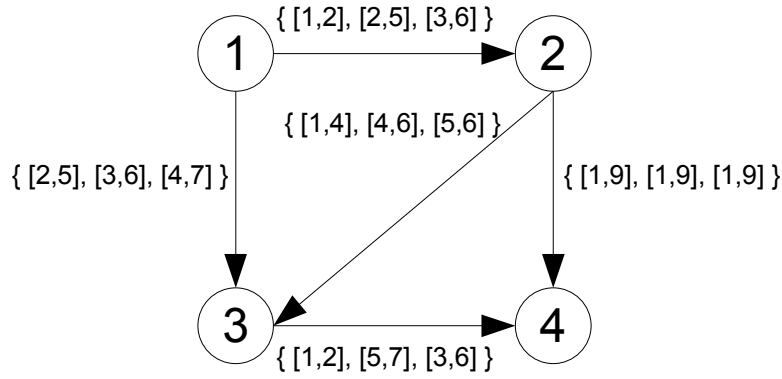


FIG. 3.1 – Exemple d'un graphe dynamique avec intervalles

3.2.2 Graphe FIFO avec intervalles

Une sous-classe des graphes dynamiques avec intervalles est à présent définie. Cette classe s'inspire de la classe des graphes FIFO définie dans la section 1.3.4.2.

Définition 30 *Un graphe dynamique avec intervalle est dit graphe FIFO avec intervalles si tous ses arcs vérifient la condition FIFO.*

Dans le cas général, la condition FIFO pour un arc $(i,j) \in A$ est la suivante :

$$\forall t, t' \geq 0, t \leq t' \implies t + p(i,j,t) \leq t' + p(i,j,t')$$

Or, dans le cas des graphes dynamiques avec intervalles, pour chaque arc $(i,j) \in A$, $p(i,j,t) \in [a_{ij}^t, b_{ij}^t]$. Par conséquent, la condition FIFO dans le cadre des graphes dynamiques avec intervalles peut s'énoncer comme suit

Définition 31 Soient $G = (N, A)$ un graphe dynamique avec intervalles et un arc $(i,j) \in A$. (i,j) est dit arc FIFO si pour $\forall t, t' \geq 0$ et n'appartenant pas au même intervalle de temps il vérifie la propriété suivante

$$t + b_{ij}^t \leq t' + a_{ij}^{t'}$$

3.3 Problème du plus court chemin

Pour les graphes dynamiques avec intervalles, la question du calcul du plus court chemin se pose. Deux approches s'offrent à nous. La première consiste à calculer le plus court chemin de déviation robuste comme dans la section 1.3.6. La deuxième consiste à rendre les poids déterministes en utilisant un critère d'optimisme.

3.3.1 Problème du plus court chemin de déviation robuste

Ce problème constitue une extension du problème du plus court chemin de déviation robuste pour les graphes statiques avec intervalles². Cette approche permet d'obtenir une solution de bonne qualité dans tous les scénarios

2. Consulter la section 1.3.6 pour une définition formelle.

possibles. En effet, pour ce type de problème, le but est de trouver le chemin qui possède la déviation robuste maximale³ la plus faible.

Proposition 3 *Le problème du plus court chemin de déviation robuste est NP-Difficile pour les graphes dynamiques avec intervalles.*

Preuve 6 *La preuve de la proposition précédente se base sur le fait que le problème du plus court chemin de déviation robuste est NP-Difficile pour les graphes statiques avec intervalles [130] [133]. En effet, les graphes statiques avec intervalles sont des cas particuliers des graphes dynamiques avec intervalles. Un graphe statique avec intervalles est un graphe dynamique avec intervalles où, pour tous les arcs, les poids pour tous les intervalles de temps sont égaux. En conséquence, le problème est NP-Difficile pour les graphes dynamiques avec intervalles.*

Proposition 4 *Le problème du plus court chemin de déviation robuste est NP-Difficile pour les graphes FIFO avec intervalles.*

Preuve 7 *Certains graphes statiques avec intervalles peuvent être considérés comme des cas particuliers des graphes FIFO avec intervalles. Effectivement, soit un graphe statique avec intervalles tel que tous les poids des arcs sont compris dans l'intervalle $[1,2]$. Posons T comme étant la durée de l'intervalle de temps. Tous les arcs de ce graphe vérifient la condition FIFO énoncée précédemment. En effet, soient t et t' tel que $t \leq t'$. Si t et t' appartiennent au même intervalle de temps alors pour un arc (i,j) , $p(i,j,t) = p(i,j,t')$. D'où, $t + p(i,j,t) \leq t' + p(i,j,t')$. Si t et t' n'appartiennent pas au même intervalle de temps alors soient δ l'instant de début de l'intervalle de temps auquel appartient t et δ' celui auquel appartient t' . Nous avons alors $\delta + T \leq \delta'$. Par définition, nous avons que $\delta \leq t \leq \delta + (T - 1)$ et $\delta' \leq t' \leq \delta' + (T - 1)$ (car*

3. Consulter la section 1.3.6 pour une définition formelle.

le temps est considéré comme discret). Par définition, nous avons aussi que $a_{ij}^t \leq p(i,j,t) \leq b_{ij}^t$ et $a_{ij}^{t'} \leq p(i,j,t') \leq b_{ij}^{t'}$. Comme $a_{ij}^t = 1$ et $b_{ij}^t = 2$ pour tous les arcs et pour tous les intervalles de temps, nous avons alors $p(i,j,t) \leq 2$ et $1 \leq p(i,j,t')$. Par suite, $t+p(i,j,t) \leq \delta+T+1$ et $\delta'+1 \leq t'+p(i,j,t')$. Or comme $\delta+T \leq \delta'$, alors $t+p(i,j,t) \leq t'+p(i,j,t')$. En conclusion, le graphe statique avec intervalles considéré est un graphe dynamique avec intervalles. Comme le problème du plus court chemin de déviation robuste est NP-Difficile pour les graphes statiques avec intervalles [130] [133], le problème est NP-Difficile pour les graphes FIFO avec intervalles.

En conclusion, le problème du plus court chemin de déviation robuste est NP-Difficile pour les graphes dynamiques avec intervalles et les graphes FIFO avec intervalles. Néanmoins, dans la suite, nous ne tentons pas de résoudre ce problème. Effectivement, dans le reste du chapitre, l'accent est mis sur la résolution du problème du plus court chemin suivant un critère d'optimisme qui est décrit ci-dessous. Cette dernière problématique est favorisée à cause de la grande flexibilité qu'elle offre dans ses applications pratiques.

3.3.2 Problème du plus court chemin suivant un critère d'optimisme

Cette deuxième approche permet de calculer un itinéraire optimal suivant "les préférences d'un décideur". Effectivement, pour chaque intervalle de temps et pour chaque arc, il faut rendre le poids de l'arc déterministe. Il est possible alors de considérer des cas extrêmes (poids minimal, poids maximal) ou un cas intermédiaire. Par exemple, considérons le cas d'un chauffeur routier qui doit transporter des marchandises jusqu'à un port où elles seront transportées par bateau. Supposons maintenant deux cas. Dans le premier

cas, le chauffeur, suite à une panne, part en retard et il doit calculer son itinéraire pour arriver au port avant le départ du bateau. Dans le deuxième cas, le chauffeur planifie son itinéraire à l'avance pour déterminer son heure de départ. Dans le premier cas, le chauffeur a intérêt à être optimiste et de supposer que les poids des arcs seront minimaux pour ne pas rater le bateau. Dans le deuxième cas, le chauffeur a intérêt à être pessimiste et de supposer que les poids des arcs seront maximaux afin de parer à toute éventualité et d'arriver, dans tous les cas, avant le départ du bateau. Cette approche consiste alors à calculer le plus court chemin suivant un degré d'optimisme fixé par un décideur. Ce problème est appelé *problème du plus court chemin suivant un critère d'optimisme*.

Proposition 5 *Le problème du plus court chemin suivant un critère d'optimisme est NP-Difficile pour les graphes dynamiques avec intervalles et il est polynomial pour les graphes FIFO avec intervalles.*

Preuve 8 *En rendant les poids déterministes, le graphe dynamique avec intervalles devient un graphe dynamique déterministe comme présenté dans la section 1.3.4. En conséquence, le problème du plus court chemin suivant un critère d'optimisme devient un problème de calcul du plus court chemin dans un graphe dynamique. Si le graphe de départ est un graphe dynamique avec intervalles quelconque alors le graphe résultant est un graphe dynamique quelconque. Le problème du plus court chemin est NP-Difficile pour ce type de graphes [103]. Par contre, si le graphe de départ est un graphe FIFO avec intervalles alors le graphe résultant est un graphe FIFO. Le problème du plus court chemin est polynomial pour ce type de graphes [2] [78].*

3.3.3 Résolution du problème du plus court chemin suivant un critère d'optimisme

3.3.3.1 Facteur d'optimisme

Pour modéliser les “préférences d'un décideur”, la notion de facteur d'optimisme est utilisée. Cette notion est présente dans la théorie de la décision et plus précisément dans les critères de décision dans un avenir incertain dans la section 1.4. Parmi ces critères, le critère d'Hurwitz utilise un facteur d'optimisme α pour pondérer l'effet du meilleur cas et celui du pire cas. Le critère d'Hurwitz est valable dans le cadre d'un jeu sans mémoire et contre la nature. Ces hypothèses sont vérifiées pour le trafic routier. Par suite, ce facteur d'optimisme peut être utilisé pour notre problématique. Il est défini comme suit :

Définition 32 *Soit le facteur d'optimisme α . $\alpha \in \mathbb{R}$ et il est tel que $0 \leq \alpha \leq 1$. Dans le cas le plus optimiste, $\alpha = 0$. Dans le cas le plus pessimiste $\alpha = 1$.*

Pour représenter le poids d'un arc grâce au facteur d'optimisme, une variable aléatoire X_{ij}^t est utilisée. Cette variable aléatoire décrit la distribution des poids possibles d'un arc dans l'intervalle $[a_{ij}^t, b_{ij}^t]$. X_{ij}^t peut être une variable aléatoire discrète ou continue. Elle peut suivre une loi équiprobable, une loi normale, une loi asymétrique ou n'importe quelle autre loi suivant le problème traité.

Définition 33 *Le poids de l'arc (i,j) à l'instant t , noté $p(i,j,t)$, est défini comme suit*

- Si $p(i,j,t) \in \mathbb{R}_+^*$ alors $p(i,j,t)$ est le réel vérifiant la propriété $P(X_{ij}^t \leq p(i,j,t)) = \alpha$;

- Si $p(i,j,t) \in \mathbb{N}^*$ alors $p(i,j,t) = E(x)$ avec x est le réel vérifiant la propriété $P(X_{ij}^t \leq x) = \alpha$ et E est la fonction partie entière.

Grâce au facteur α , le décideur peut fixer ses préférences pour le choix des poids des arcs. En effet, le poids de l'arc se situe sur l'intervalle $[a_{ij}^t, b_{ij}^t]$. Donc, si le décideur est optimiste alors $\alpha = 0$. Par suite, $p(i,j,t) = a_{ij}^t$ car $P(X_{ij}^t \leq a_{ij}^t) = 0$. Si, par contre le décideur est pessimiste alors $\alpha = 1$. Par conséquent, $p(i,j,t) = b_{ij}^t$ car $P(X_{ij}^t \leq b_{ij}^t) = 1$.

Proposition 6 Si X est une variable aléatoire continue sur $[a,b]$ alors il existe un réel $p \in [a,b]$ unique tel que $P(X \leq p) = \alpha$ avec α une constante.

Preuve 9 Soient X une variable aléatoire continue sur $[a,b]$, f sa densité de probabilité et α une constante. Nous avons

$$\begin{aligned} P(X \leq y) &= \int_{-\infty}^y f(x)dx \\ &= \int_a^y f(x)dx \end{aligned}$$

Posons $F(y) = \int_a^y f(x)dx - \alpha$ défini sur $[a,b]$. Par suite, montrer qu'il existe un seul réel p tel que $P(X \leq p) = \alpha$ revient à montrer qu'il existe un seul réel p tel que $F(p) = 0$.

Montrons d'abord **l'existence** de p . Nous avons

- F est une fonction continue sur $[a,b]$ car elle est dérivable sur $[a,b]$;
- $F(a) = -\alpha \leq 0$ car $0 \leq \alpha \leq 1$;
- $F(b) = 1 - \alpha \geq 0$ car $0 \leq \alpha \leq 1$.

En conséquence, d'après le théorème des valeurs intermédiaires $\exists p \in [a,b]$ tel que $F(p) = 0$. Par suite, $\exists p \in [a,b]$ tel que $P(X \leq p) = \alpha$.

Montrons maintenant **l'unicité** de p . Nous avons

- F est une fonction continue sur $[a,b]$ car elle est dérivable sur $[a,b]$;

- F est strictement monotone sur $[a,b]$ car f est strictement positive sur $[a,b]$.

Par conséquent, d'après le théorème de bijection, F est une bijection sur $[a,b]$. Par suite, $\exists! p \in [a,b]$ tel que $F(p) = 0$. Donc, $\exists! p \in [a,b]$ tel que $P(X \leq p) = \alpha$.

Proposition 7 Si X est une variable aléatoire discrète sur $[a,b]$ alors il n'existe pas toujours un réel $p \in [a,b]$ tel que $P(X \leq p) = \alpha$ avec α une constante.

Preuve 10 Pour montrer la proposition précédente, un contre-exemple est construit. Soit X une variable aléatoire discrète qui ne peut prendre que les valeurs a ou b ($X \in \{a,b\}$) avec $P(a) = 0.5$ et $P(b) = 0.5$. Si nous considérons $\alpha = 0.75$, $\nexists p \in [a,b]$ tel que $P(X \leq p) = \alpha$. En effet, $P(X \leq y) = 0.5 \forall y \in [a,b[$ et $P(X \leq y) = 1$ pour $y = b$.

Hypothèse 5 La variable aléatoire X_{ij}^t est une variable aléatoire continue sur $[a_{ij}^t, b_{ij}^t]$. De plus, elle est supposée connue pour tous les arcs et pour tous les intervalles de temps.

L'hypothèse précédente est introduite pour les raisons suivantes. Premièrement, lorsque X_{ij}^t est une variable aléatoire discrète, l'existence d'un poids n'est plus assurée. Deuxièmement, la connaissance a priori des variables aléatoires associées à chaque arc est indispensable pour réaliser un algorithme hors-ligne.

Étude d'un exemple Soit X_{ij}^t , $i, j \in N$ une variable aléatoire continue qui suit une loi équiprobable. Soit f la densité de probabilité associée à cette variable aléatoire. Étant donné que X_{ij}^t suit une loi équiprobable et qu'elle

est définie sur $[a_{ij}^t, b_{ij}^t]$ alors $f(x) = \frac{1}{b_{ij}^t - a_{ij}^t} \forall x \in [a, b]$. $p(i, j, t)$ est défini tel que $P(X_{ij}^t \leq p(i, j, t)) = \alpha$ d'où

$$\begin{aligned}
 P(X_{ij}^t \leq p(i, j, t)) &= \alpha \\
 \Rightarrow \int_{-\infty}^{p(i, j, t)} f(x) dx &= \alpha \\
 \Rightarrow \int_{a_{ij}^t}^{p(i, j, t)} f(x) dx &= \alpha \\
 \Rightarrow \int_{a_{ij}^t}^{p(i, j, t)} \frac{dx}{b_{ij}^t - a_{ij}^t} &= \alpha \\
 \Rightarrow \frac{p(i, j, t) - a_{ij}^t}{b_{ij}^t - a_{ij}^t} &= \alpha \\
 \Rightarrow p(i, j, t) &= a_{ij}^t + (b_{ij}^t - a_{ij}^t)\alpha
 \end{aligned}$$

D'où pour cet exemple, le poids d'un arc est défini par la fonction $p(i, j, t) = a_{ij}^t + (b_{ij}^t - a_{ij}^t)\alpha$. Dans le cas le plus optimiste, $\alpha = 0$ d'où $p(i, j, t) = a_{ij}^t$. Dans le cas le plus pessimiste, $\alpha = 1$ d'où $p(i, j, t) = b_{ij}^t$. Le graphe de la figure 3.2 reprend le graphe de la figure 3.1 en supposant que toutes les variables aléatoires associées aux arcs suivent une loi équiprobable et en appliquant un facteur d'optimisme de 0.25.

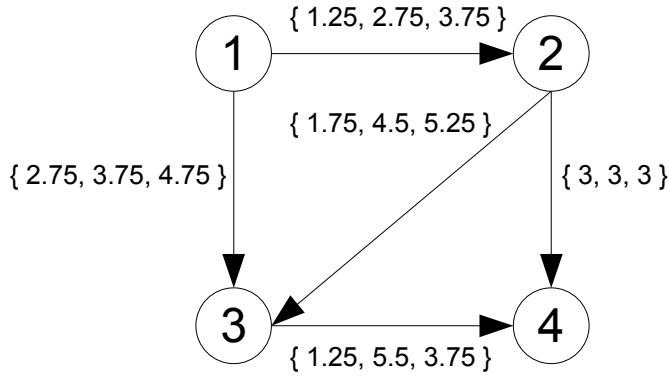


FIG. 3.2 – Application d'un facteur d'optimisme de 0.25 sur un graphe dynamique avec intervalles

3.3.3.2 Calcul du plus court chemin

Après avoir fixé le facteur d'optimisme et rendu les poids des arcs déterministes, la question du calcul effectif du plus court chemin se pose. Dans le cas général, le graphe obtenu est un graphe dynamique. Le problème du plus court chemin dans un graphe dynamique est *NP-Difficile* [103]. Dans ce cas, le paradigme algorithmique *Chrono-SPT* peut être utilisé comme indiqué dans la section 1.3.4.3. Par contre, si au départ, le graphe est un graphe FIFO avec intervalles alors le graphe obtenu après avoir rendu les poids déterministes est un graphe FIFO. Dans ce cas, une version dynamique de l'algorithme Dijkstra peut être utilisée pour calculer le plus court chemin et l'optimalité de la solution est garantie⁴.

3.4 Calcul du plus court chemin à l'aide d'un facteur d'optimisme dynamique

Dans la première partie du chapitre, le critère d'optimisme a été modélisé par un facteur d'optimisme α . Ce facteur, fixé par le décideur, est indépendant du temps et il est le même pour l'ensemble des arcs du graphe. De plus le calcul du plus court chemin est effectué hors-ligne. Cependant, il semble plus pertinent d'avoir une valeur dynamique de α et de réaliser un calcul en-ligne du plus court chemin. En effet, en considérant l'exemple concret du trafic routier, il apparaît que le facteur d'optimisme dépend de deux paramètres essentiels qui sont le temps et l'arc considéré. Effectivement, d'une part, la variation du trafic routier peut être plus ou moins stable suivant la période de la journée. D'où il est intéressant d'être plus optimiste lorsque la variation

4. Voir la section 1.3.4.3.

du trafic est stable que lorsqu'elle est instable. D'autre part, toutes les routes d'un même réseau routier ne sont pas équipées en instruments de mesures du trafic. En conséquence, la quantité d'informations disponibles pour ce type de routes est faible. D'où il est plus prudent d'être plus pessimiste sur les routes non équipées que sur le reste des routes. De plus, l'avantage du calcul en-ligne par rapport au calcul hors-ligne est qu'il est possible d'améliorer les calculs en corrigeant les erreurs commises dans le passé. En d'autres termes, cela revient à réaliser un contrôle en boucle fermée sur le calcul du plus court chemin. C'est pourquoi, dans la suite, le facteur d'optimisme est représenté par une fonction $\alpha(i,j,t)$. Le poids d'un arc (i,j) à l'instant t est défini de la manière suivante :

Définition 34 *Le poids de l'arc (i,j) à l'instant t , noté $p(i,j,t)$, est défini comme suit*

- Si $p(i,j,t) \in \mathbb{R}_+^*$ alors $p(i,j,t)$ est le réel vérifiant la propriété $P(X_{ij}^t \leq p(i,j,t)) = \alpha(i,j,t)$;
- Si $p(i,j,t) \in \mathbb{N}^*$ alors $p(i,j,t) = E(x)$ avec x est le réel vérifiant la propriété $P(X_{ij}^t \leq x) = \alpha(i,j,t)$ et E est la fonction partie entière.

A présent, toute la difficulté réside dans la définition de $\alpha(i,j,t)$. Cette fonction peut être définie de plusieurs manières et chaque définition peut s'avérer plus ou moins efficace. Notre proposition consiste à définir $\alpha(i,j,t)$ à l'aide de quatre paramètres : α_{profil} , $\beta(i,j,t)$, $\gamma(i,j,t)$ et $\delta(i,j,t)$. α_{profil} modélise les désirs du décideur et il est considéré comme étant le paramètre principal. Quant aux paramètres $\beta(i,j,t)$, $\gamma(i,j,t)$ et $\delta(i,j,t)$, ce sont des paramètres secondaires qui ont pour rôle de corriger de α_{profil} . Ils doivent permettre d'augmenter ou de diminuer progressivement α_{profil} afin de tenir compte des différents phénomènes pouvant influencer sur le choix du poids de l'arc. C'est

pourquoi, ces paramètres sont représentés sous la forme d'un pourcentage qu'il faut multiplier par α_{profil} . Par exemple, si le paramètre $\beta(i,j,t)$ est défini tel que sa valeur est égale à 120% pour un arc donné et à un instant donné alors le facteur d'optimisme en prenant en compte uniquement le facteur $\beta(i,j,t)$ est $1.2 \times \alpha_{profil}$.

3.4.1 Le paramètre α_{profil}

3.4.1.1 Définition

α_{profil} est un paramètre constant qui indique les préférences générales du décideur. Le profil du décideur peut être à tendance optimiste ou à tendance pessimiste. Le paramètre α_{profil} est considéré afin de s'adapter à différentes applications. La définition de α_{profil} est la même que celle du facteur d'optimisme α présenté précédemment. Notamment, il garde la même valeur pour tous les arcs du graphe et tous les intervalles de temps.

3.4.1.2 Exemple

Reprenons l'exemple du transporteur routier qui part et qui doit arriver au port avant le départ du bateau présenté initialement dans la page 97. Ce transporteur doit absolument arriver à sa destination avant l'heure limite du départ du bateau. Dans le cas où il part en retard, il doit être optimiste. Dans ce cas α_{profil} peut être défini tel que $\alpha_{profil} = 0.25$. Dans le cas où il planifie son itinéraire à l'avance, il doit être pessimiste pour avoir un grand degré de fiabilité. Dans ce cas, α_{profil} peut être défini tel que $\alpha_{profil} = 0.8$.

3.4.2 Le paramètre β

3.4.2.1 Définition

$\beta(i,j,t)$ est une fonction qui indique l'influence de l'arc (i,j) sur le facteur d'optimisme. En effet, certaines routes peuvent être connues pour des phénomènes de congestion récurrents et imprévisibles dus à leur structure ou à leur localisation (par exemple un tunnel ou un pont). Par conséquent, il est pertinent d'être plus pessimiste sur ce type de route. Cette fonction représente un pourcentage à affecter au facteur d'optimisme permettant de l'augmenter ou de le diminuer.

3.4.2.2 Exemple

Dans cet exemple, $\beta(i,j,t)$ est définie telle qu'elle dépende de la position de l'arc (i,j) par rapport à la position effective du véhicule. En effet, pour le calcul en-ligne d'un chemin optimal, le calcul du chemin est réitéré à chaque déplacement du véhicule. En conséquence, il est pertinent que la valeur de $\beta(i,j,t)$ puisse s'adapter aux différents déplacements du véhicule. Dans ce but, la notion d'horizon est introduite.

Définition de la notion d'horizon Définir un horizon autour du véhicule revient à délimiter un espace spatial ou temporel autour du véhicule [93]. Les horizons peuvent être schématisés par des cercles concentriques ayant comme centre le véhicule. Ces différents horizons sont liés au véhicule. En conséquence, ils se déplacent avec lui. La figure 3.3 présente un exemple schématique. Dans cet exemple, trois horizons, nommés H_1 , H_2 et H_3 , sont représentés.

Pour les graphes, définir un horizon revient à déterminer un sous-ensemble d'arcs du graphe liés par une même propriété spatiale ou temporelle par rap-

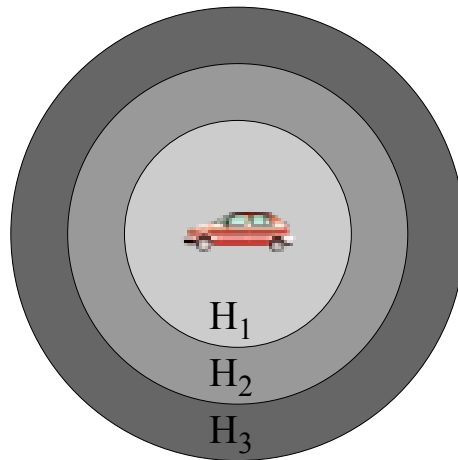


FIG. 3.3 – Exemple schématique de la notion d'horizon

port au véhicule. Plusieurs propriétés sont possibles : distance euclidienne, distance en nombre d'arcs, la durée approximative pour atteindre l'arc, . . . La pertinence de la propriété choisie et le nombre d'horizons dépendent du problème traité, la taille du graphe et du degré de précision souhaité. D'une manière générique, les horizons sont définis de la manière suivante :

Définition 35 Soient $\{H_1, H_2, \dots, H_n\}$ l'ensemble des horizons associé à un graphe $G = (N, A)$. Un horizon H_k est un sous-ensemble des arcs de G vérifiant une propriété spatiale ou temporelle $Prop_k$. D'une manière formelle, H_k est défini comme suit :

$$\forall k / 1 \leq k \leq n, H_k = \{(i, j) \in A / (i, j) \text{ vérifie } Prop_k\}$$

Exemple d'horizons Dans cet exemple, la durée approximative pour atteindre l'arc à partir de la position effective du véhicule est le critère permettant de délimiter les horizons. Considérons le graphe de la figure 3.4. Dans cet exemple, l'intervalle de temps pour les poids est de 5 unités de temps. D'où chaque intervalle de poids est valable pendant 5 unités de temps. En

conséquence, les horizons schématisés dans la figure 3.3 sont définis comme suit :

H_1 est l'ensemble des arcs (i,j) tel que la durée approximative pour atteindre (i,j) à partir de la position effective du véhicule est inférieure à 5 unités de temps

H_2 est l'ensemble des arcs (i,j) tel que la durée approximative pour atteindre (i,j) à partir de la position effective du véhicule est comprise entre 5 et 10 unités de temps

H_3 est l'ensemble des arcs (i,j) tel que la durée approximative pour atteindre (i,j) à partir de la position effective du véhicule est supérieure à 10 unités de temps

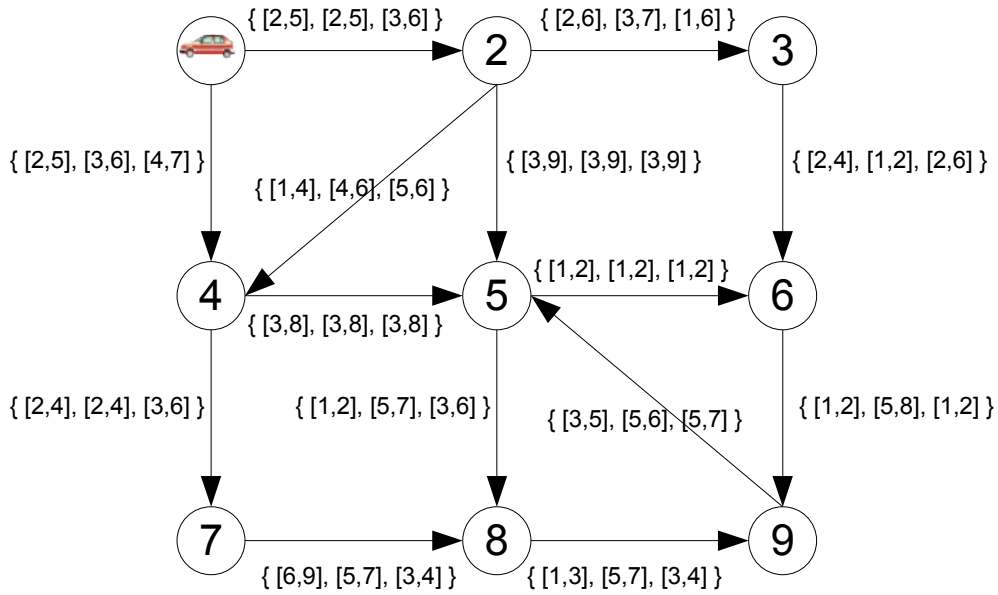


FIG. 3.4 – Exemple d'un graphe dynamique avec une pondération sous forme d'intervalles

La figure 3.5 reprend le graphe de la figure 3.4 en spécifiant les arcs appartenant à chaque horizon. Pour expliciter la méthode de calcul, l'arc (9,5) est considéré. Pour connaître, à quel horizon appartient l'arc, il faut calculer le temps approximatif pour atteindre le nœud 9. Pour réaliser ceci, une heuristique est utilisée. Cette dernière consiste à remplacer chaque intervalle par sa valeur minimale. Ensuite, l'algorithme de Dijkstra est déroulé sur le graphe dynamique à poids discret obtenu pour calculer le plus court chemin de la position effective du véhicule au nœud 9. La figure 3.6 présente le graphe obtenu en suivant cette méthode. Le plus court chemin calculé avec l'algorithme de Dijkstra est le chemin (1,2,3,6,9) et, de plus, sa durée est égale à $2+2+2+5 = 11$ unités de temps. Dans ce calcul, le poids utilisé est le poids de l'arc à l'instant de départ du nœud origine. Sachant que chaque poids ne reste valable que 5 unités de temps, le poids utilisé pour les arcs (1,2), (2,3) et (3,6) est le poids du premier intervalle. Par contre, pour l'arc (6,9), c'est le poids du deuxième intervalle qui est considéré. Étant donné que la durée du chemin est égale à 11, l'arc (9,5) appartient à l'horizon H_3 .

Valeurs de $\beta(i,j,t)$ en fonction des horizons Dans notre exemple, une valeur de $\beta(i,j,t)$ est fixée pour chaque horizon. Pour fixer ces valeurs le principe suivant est respecté : plus l'arc se trouve dans un horizon lointain, plus $\beta(i,j,t)$ influe de manière pessimiste sur le facteur d'optimisme. En effet, ce principe est suivi car plus l'arc est lointain, plus l'incertitude sur l'intervalle à utiliser pour les calculs grandit. En effet, pour un graphe dynamique, l'algorithme de calcul du plus court chemin utilise le poids de l'arc à l'instant de départ de cet arc 1.3.4. Dans notre modèle, les poids des arcs sont définis par des intervalles. Par suite, plus l'arc se trouve dans un horizon lointain, plus l'incertitude sur l'instant de départ de l'arc grandit. C'est pourquoi une

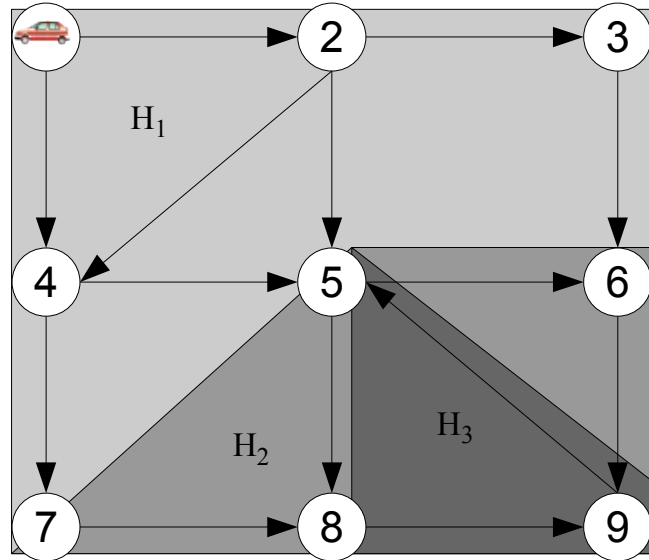


FIG. 3.5 – Graphe dynamique avec des horizons spécifiés

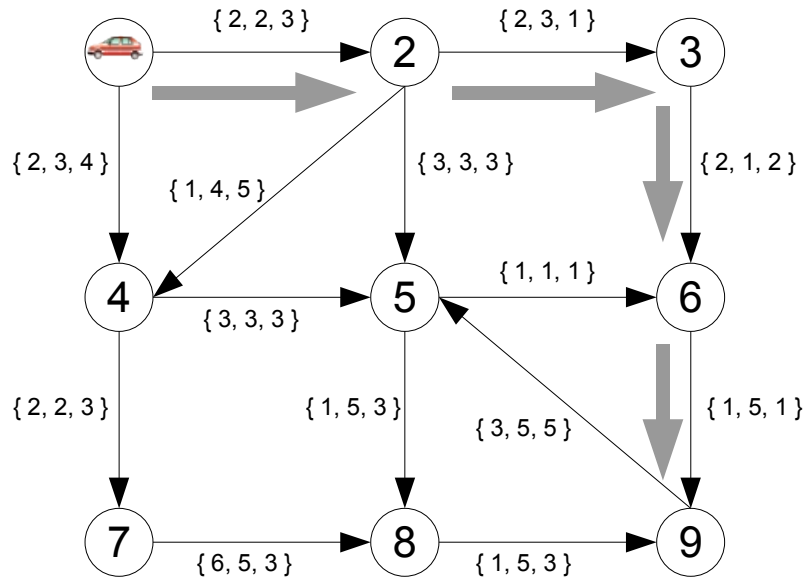


FIG. 3.6 – Exemple de calcul de la durée approximative pour atteindre un nœud

attitude prudente est adoptée. D'où, $\beta(i,j,t)$ est définie comme suit :

$$\beta(i,j,t) = \begin{cases} 100\% & \text{Si } (i,j) \in H_1 \text{ à l'instant } t \\ 110\% & \text{Si } (i,j) \in H_2 \text{ à l'instant } t \\ 120\% & \text{Si } (i,j) \in H_3 \text{ à l'instant } t \end{cases}$$

3.4.3 Le paramètre γ

3.4.3.1 Définition

$\gamma(i,j,t)$ est un paramètre qui permet de tenir compte des erreurs d'estimation réalisées dans le passé. Ce paramètre est particulièrement utile dans le calcul en-ligne de l'itinéraire. Effectivement, son but est de réaliser un contrôle en boucle fermée sur le facteur d'optimisme pour le corriger dynamiquement. En effet, il consiste à dire que si, dans le passé, le facteur était trop optimiste par rapport aux valeurs réelles alors il faut le diminuer et inversement. Ce paramètre représente un pourcentage à affecter au facteur d'optimisme permettant de l'augmenter ou de le diminuer.

3.4.3.2 Exemple

Dans cet exemple, la fonction $\gamma(i,j,t)$ est définie, de manière récursive, à l'aide de $p_{est}(i,j,t-1)$ (poids estimé de l'arc (i,j) à l'instant $t-1$), $p_{eff}(i,j,t-1)$ (poids effectif de l'arc (i,j) à l'instant $t-1$) et $\gamma(i,j,t-1)$. Grâce à cette définition récursive, un cumul des calibrages précédents est réalisé. $\gamma(i,j,t)$ est définie de la manière suivante :

$$\gamma(i,j,t) = \begin{cases} 100\% & \text{Si } t = 0 \\ \gamma(i,j,t-1) \times \left(1 + \left(\frac{p_{eff}(i,j,t-1) - p_{est}(i,j,t-1)}{b_{ij}^{t-1} - a_{ij}^{t-1}} \right) \right) & \text{Sinon} \end{cases}$$

La définition précédente implique que si à $t-1$ notre estimation était trop optimiste (le poids a été sous-estimé) alors $p_{eff}(i,j,t-1) - p_{est}(i,j,t-1) > 0$.

D'où, $\gamma(i,j,t) > \gamma(i,j,t-1)$. Par suite, $\gamma(i,j,t)$ va influencer sur $\alpha(i,j,t)$ pour qu'il soit plus pessimiste que $\alpha(i,j,t-1)$. En suivant le même raisonnement, l'effet contraire est obtenu dans le cas d'une estimation trop pessimiste (le poids est surestimé). Dans le cas particulier où $p_{eff}(i,j,t-1) = p_{est}(i,j,t-1)$, $\gamma(i,j,t) = \gamma(i,j,t-1)$. En conséquence, le facteur $\alpha(i,j,t)$ est bien calibré. Il est à noter que l'erreur d'estimation doit être prise en compte proportionnellement à la largeur de l'intervalle $[a_{ij}^{t-1}, b_{ij}^{t-1}]$. Par conséquent, le terme $b_{ij}^{t-1} - a_{ij}^{t-1}$ est introduit dans la définition de notre fonction.

Soit un arc (i,j) . Supposons que $[a_{ij}^{t-1}, b_{ij}^{t-1}] = [a_{ij}^t, b_{ij}^t] = [1, 10]$ et que $p_{eff}(i,j,t-1) = 5$. Supposons que, dans un premier cas, $p_{est}(i,j,t-1) = 2$. En conséquence, le poids de l'arc (i,j) a été sous-estimé à l'instant $t-1$. Par suite,

$$\begin{aligned}\gamma(i,j,t) &= \gamma(i,j,t-1) \times \left(1 + \left(\frac{5-2}{10-1}\right)\right) \\ &= \gamma(i,j,t-1) \times \left(1 + \frac{1}{3}\right) \\ &= \gamma(i,j,t-1) \times 1.33\end{aligned}$$

D'où la valeur de $\gamma(i,j,t)$ est augmentée d'un tiers par rapport à $\gamma(i,j,t-1)$. Dans un deuxième cas, supposons que $p_{est}(i,j,t-1) = 8$. En conséquence, le poids de l'arc (i,j) a été surestimé à l'instant $t-1$. Par suite,

$$\begin{aligned}\gamma(i,j,t) &= \gamma(i,j,t-1) \times \left(1 + \left(\frac{5-8}{10-1}\right)\right) \\ &= \gamma(i,j,t-1) \times \left(1 - \frac{1}{3}\right) \\ &= \gamma(i,j,t-1) \times 0.66\end{aligned}$$

D'où la valeur de $\gamma(i,j,t)$ est diminuée d'un tiers par rapport à $\gamma(i,j,t-1)$.

3.4.4 Le paramètre δ

3.4.4.1 Définition

$\delta(i,j,t)$ est une fonction qui permet d'intégrer tous les phénomènes résiduels qui peuvent influencer sur le facteur d'optimisme. Par exemple, dans le

cas d'un réseau routier, elle peut représenter l'influence de certains phénomènes météo comme la pluie ou la neige, la probabilité d'un accident sur une route particulière ... Cette fonction dépend de l'arc du graphe et du temps. Cette fonction représente un pourcentage à affecter au facteur d'optimisme permettant de l'augmenter ou de le diminuer.

3.4.4.2 Exemple

Dans cet exemple, seule l'influence des phénomènes météo est considérée. De plus, cette fonction est considérée uniforme pour tous les arcs. La fonction $\delta(i,j,t)$ est définie comme suit :

$$\delta(i,j,t) = \begin{cases} 100\% & \text{Si beau temps} \\ 110\% & \text{Si pluie} \\ 120\% & \text{Si neige} \end{cases}$$

3.4.5 Agrégation des paramètres

Une fois les paramètres définis, la question de leur agrégation pour l'obtention du facteur d'optimisme se pose. Le paramètre principal qui doit influencer sur $\alpha(i,j,t)$ est α_{profil} car il modélise les désirs du décideur. Par contre, les paramètres $\beta(i,j,t)$, $\gamma(i,j,t)$ et $\delta(i,j,t)$ ont pour rôle de corriger α_{profil} . D'où la nécessité d'agréger ces trois paramètres de correction pour les multiplier à α_{profil} . Dans la suite, deux techniques sont utilisées pour effectuer ce travail : l'agrégation par somme pondérée et l'agrégation par l'intégrale de Choquet. Ces deux techniques sont détaillées dans la section 1.2.3.1.

3.4.5.1 Agrégation par somme pondérée

L'agrégation par somme pondérée est la technique la plus populaire et la plus simple à utiliser. Pour calculer la valeur de $\alpha(i,j,t)$, la somme pondérée

de $\beta(i,j,t)$, $\gamma(i,j,t)$ et $\delta(i,j,t)$ est calculée et multipliée par α_{profil} . $\alpha(i,j,t)$ est alors définie comme suit :

$$\alpha(i,j,t) = \min \left\{ 1, \alpha_{profil} \times \left(w_1 \times \beta(i,j,t) + w_2 \times \gamma(i,j,t) + w_3 \times \delta(i,j,t) \right) \right\}$$

w_1 , w_2 et w_3 représentent les pondérations des différents facteurs. Si, pour une application donnée, tous les facteurs possèdent la même importance alors $w_1 = w_2 = w_3 = \frac{1}{3}$. Sinon, une pondération plus importante est affectée aux paramètres importants en respectant la condition $w_1 + w_2 + w_3 = 1$.

3.4.5.2 Agrégation par l'intégrale de Choquet

Cette technique, plus récente que la première, permet de modéliser des aspects qui ne peuvent pas l'être par une somme pondérée. Elle s'appuie sur la définition d'une mesure floue μ permettant de mesurer l'utilité de chaque sous-ensemble de paramètres. Donc, comme étape préliminaire, il faut déterminer la mesure floue μ suivant les préférences du décideur. Par la suite, $\alpha(i,j,t)$ est définie comme suit :

$$\alpha(i,j,t) = \min \left\{ 1, \alpha_{profil} \times \Delta_\mu \left(\beta(i,j,t), \gamma(i,j,t), \delta(i,j,t) \right) \right\}$$

avec Δ_μ l'intégrale de Choquet discrète par rapport à la mesure floue μ . Δ_μ est définie par :

$$\Delta_\mu \left(\beta(i,j,t), \gamma(i,j,t), \delta(i,j,t) \right) = \sum_{i=1}^3 (\Delta_{\sigma(i)} - \Delta_{\sigma(i-1)}) \times \mu(A_i)$$

où

- $\Delta_i \in \left\{ \beta(i,j,t), \gamma(i,j,t), \delta(i,j,t) \right\}$;
- σ une permutation sur $\left\{ \beta(i,j,t), \gamma(i,j,t), \delta(i,j,t) \right\}$ tel que $\Delta_{\sigma(1)} \leq \Delta_{\sigma(2)} \leq \Delta_{\sigma(3)}$;
- $\Delta_{\sigma(0)} = 0$;
- $A_i = \{\Delta_{\sigma(i)}, \dots, \Delta_{\sigma(3)}\}$.

3.5 Problème du plus court chemin suivant un critère d'optimisme et un critère de déviation robuste

3.5.1 Définition du problème

L'objet de la section précédente a été le calcul du plus court chemin suivant un critère d'optimisme dans un graphe dynamique avec intervalles. Cependant, le fait de considérer un critère d'optimisme induit la prise de risque. En effet, le fait de fixer le facteur d'optimisme et de fixer les poids des arcs revient à considérer un seul scénario parmi l'ensemble des scénarios possibles. Par suite, si un scénario très défavorable se produit, alors le chemin calculé avec le facteur d'optimisme peut s'avérer très pénalisant. C'est pourquoi il est intéressant d'inclure la notion de déviation robuste dans le calcul de l'itinéraire. Ainsi, l'objectif devient le calcul du plus court chemin suivant un critère d'optimisme et qui soit le plus robuste possible. Par conséquent, le problème se transforme en un problème d'optimisation multicritère⁵.

3.5.2 Les variables du problème

3.5.2.1 Le coût du chemin suivant un critère d'optimisme

La première variable est le coût du chemin suivant un critère d'optimisme. Cette variable a été étudiée dans la section précédente. Pour calculer le coût du chemin, il faut d'abord fixer le facteur d'optimisme selon les désirs d'un décideur. Ensuite, à l'aide de ce facteur, les poids des arcs sont rendus déterministes. Enfin, le coût du chemin est calculé en utilisant ces poids

5. Consulter la section 1.2.3 pour une définition formelle.

déterministes.

3.5.2.2 La déviation robuste maximale

La deuxième variable du problème est la déviation robuste maximale du chemin⁶. Cette variable est assez simple à calculer pour les graphes statiques avec intervalles mais ne l'est pas pour les graphes dynamiques avec intervalles. Effectivement, il a été prouvé que, pour les graphes statiques avec intervalles, le scénario qui maximise la déviation robuste d'un chemin est le scénario où les arcs formant le chemin possèdent des poids maximaux et tous les autres arcs du graphe possèdent des poids minimaux [77]. Au contraire, le problème du calcul de la déviation robuste d'un chemin pour un graphe dynamique avec intervalles est *NP-Difficile*.

Proposition 8 *Le calcul de la déviation robuste maximale pour un chemin est un problème NP-Difficile pour les graphes dynamiques avec intervalles et les graphes FIFO avec intervalles.*

Preuve 11 *Pour montrer la proposition précédente, une transformation vers un problème prouvé comme étant NP-Difficile est construite. Cette démonstration est valable aussi bien pour les graphes dynamiques avec intervalles quelconques que pour les graphes FIFO avec intervalles. Le problème prouvé comme étant NP-Difficile est le problème du plus court chemin robuste absolu [130]. Dans ce problème, la valeur maximale du coût du chemin pour tous les scénarios possibles est le critère de comparaison des chemins. Le plus court chemin robuste absolu est le chemin qui minimise ce critère de comparaison.*

Soient deux nœuds i et j du graphe et un chemin $Ch(i,j)$ pour lequel il faut

6. Consulter la section 1.3.6 pour une définition formelle.

calculer la déviation robuste maximale $D_{\max}(Ch(i,j))$ ⁷. Soit C la fonction de coût classique qui consiste à réaliser la somme des poids des arcs. Définissons maintenant une nouvelle fonction coût C' . Pour un scénario s , le coût d'un chemin quelconque $Ch'(i,j)$ suivant C' est égal à la différence de coût suivant C de $Ch'(i,j)$ et $Ch(i,j)$. Formellement, C' est définie comme suit :

$$C'(Ch'(i,j)) = C(Ch'(i,j)) - C(Ch(i,j))$$

Pour le scénario s , la déviation robuste de $Ch(i,j)$ est égale à la différence entre le coût de $Ch(i,j)$ dans s et le coût du plus court chemin dans s . Par suite, calculer la déviation robuste de $Ch(i,j)$ dans s revient à trouver le chemin qui minimise C' dans s . Maintenant, pour calculer $D_{\max}(Ch(i,j))$, il faut retrouver, pour tous les scénarios possibles, la déviation robuste maximale. Ceci revient à trouver le chemin qui maximise, pour tous les scénarios possibles, la valeur minimale de la fonction C' . En d'autres termes, calculer $D_{\max}(Ch(i,j))$ revient à calculer le plus court chemin robuste absolu suivant la fonction de coût C' . Or ce problème est NP-Difficile [130]. Par suite, le calcul de la déviation robuste maximale pour un chemin dans un graphe dynamique avec intervalles ou un graphe FIFO avec intervalles est un problème NP-Difficile.

Vu la difficulté du problème, une approximation simple est utilisée afin de réaliser le calcul de la déviation maximale. Cette approximation consiste à calculer la déviation robuste du chemin pour le scénario où tous les poids des arcs du chemin sont maximaux et les poids des autres arcs sont minimaux. Contrairement aux graphes statiques, ce scénario ne fournit pas toujours la déviation robuste maximale et la figure 3.7 présente un contre-exemple. Le

7. Consulter la section 1.3.6 pour une définition formelle.

graphe de la figure 3.7 est un graphe dynamique avec intervalles et plus précisément un graphe FIFO avec intervalles. Par suite, le contre-exemple présenté est valable aussi bien pour les graphes dynamiques avec intervalles que pour les graphes FIFO avec intervalles. Pour cet exemple, la durée de l'intervalle de temps est $T = 1$ unité de temps. Le chemin pour lequel il faut calculer la déviation robuste maximale est le chemin $(1,3,4,2,5)$. Soit le scénario s où tous les poids des arcs de ce chemin sont maximaux et les poids des autres arcs sont minimaux. Ce scénario est présenté dans la figure 3.8. Dans ce scénario, le coût du chemin $(1,3,4,2,5)$ est $CD((1,3,4,2,5)) = 2 + 2 + 2 + 20 = 26$. Dans ce même scénario, le plus court chemin de 1 à 5 est le chemin $(1,2,4,5)$. Le coût du plus court chemin est $CD((1,2,4,5)) = 1 + 2 + 7 = 10$. En conséquence, $D^s((1,3,4,2,5)) = 26 - 10 = 16$. Maintenant, soit un autre scénario s' où seul le poids du deuxième intervalle de temps de l'arc $(2,4)$ a été changé par rapport au scénario s . Le scénario s' est présenté dans la figure 3.9. Dans ce scénario, le plus court chemin reste toujours $(1,2,4,5)$ et $CD((1,2,4,5)) = 1 + 1 + 1 = 3$. De plus, dans s' , $CD((1,3,4,2,5)) = 2 + 2 + 1 + 20 = 25$. D'où, $D^{s'}((1,3,4,2,5)) = 25 - 3 = 22$. Par suite, $D^{s'}((1,3,4,2,5)) > D^s((1,3,4,2,5))$. En conclusion, le scénario s n'est pas le scénario où la déviation robuste de $(1,3,4,2,5)$ est maximale.

3.5.3 Résolution

Pour la résolution des problèmes multicritères, deux approches existent : l'approche par agrégation de critères et l'approche par Pareto dominance.

3.5.3.1 Résolution par agrégation de critères

Cette approche, détaillée dans la section 1.2.3.1, consiste à agréger les différents critères en un seul. Par suite, le problème devient un problème d'op-

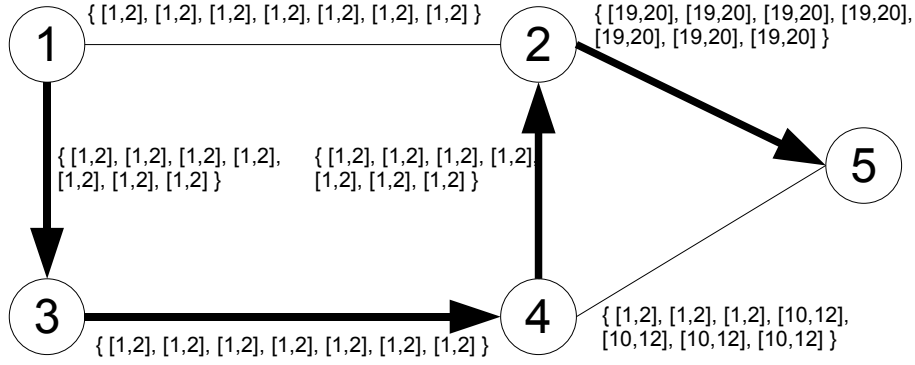


FIG. 3.7 – *Graphe du contre-exemple*

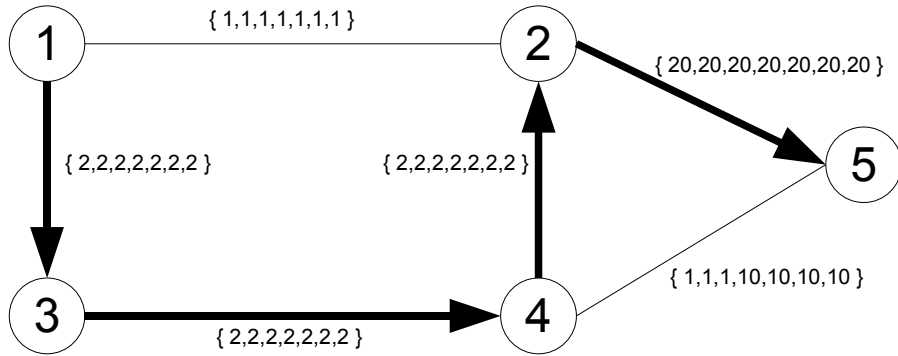


FIG. 3.8 – *Poids du graphe du contre-exemple pour le scénario de la déviation robuste maximale*

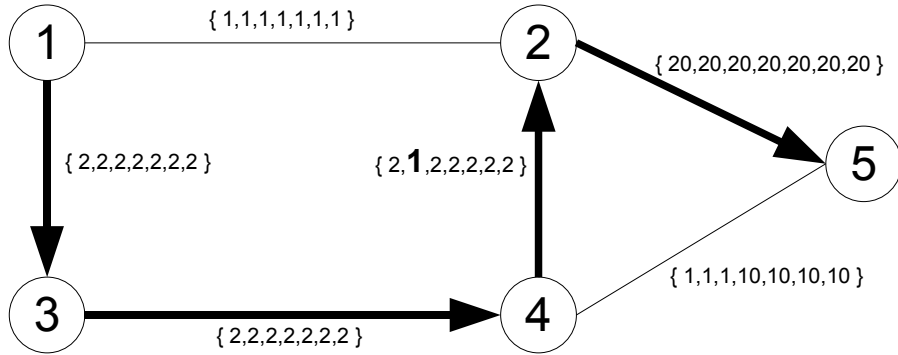


FIG. 3.9 – Poids du graphe du contre-exemple pour le scénario de la déviation robuste maximale avec modification d'un seul poids

timisation monocritère. Dans cette optique, deux techniques peuvent être utilisées. La première est l'agrégation par somme pondérée. La deuxième est l'agrégation par l'intégrale de Choquet. Étant donné que le poids de chaque arc dépend du temps alors le graphe obtenu après agrégation des critères est un graphe dynamique. En conséquence, cette technique pose le problème de l'optimalité du plus court chemin calculé après agrégation. Effectivement, le problème du plus court chemin pour les graphes dynamiques n'est polynomial que pour les graphes FIFO⁸. Par conséquent, si le graphe obtenu après l'agrégation est un graphe FIFO alors ce type de technique peut être adapté au problème. Sinon, il est préférable de résoudre le problème en utilisant l'approche par Pareto dominance.

3.5.3.2 Résolution par Pareto dominance

Cette approche, détaillée dans la section 1.2.3.2, permet de comparer les chemins en gardant les critères séparés. L'algorithme présenté dans [61] a

8. Voir section 1.3.4.3

été développé pour les graphes dynamiques. Il permet de calculer un chemin optimal au sens de Pareto suivant deux critères. Par conséquent, il correspond parfaitement à notre problématique. De plus, l'optimalité au sens de Pareto du chemin calculé a été prouvé pour les graphes FIFO et non-FIFO [61].

3.6 Synthèse

Dans ce chapitre, le problème du plus court chemin pour les graphes dynamiques avec intervalles a été étudié suivant différents points de vue. Dans la majorité des cas, un algorithme de résolution a été proposé. La figure 3.10 permet de réaliser une synthèse des différentes visions et des différents algorithmes proposés.

3.7 Conclusion

Dans ce chapitre, nous avons défini un nouveau type de graphe : les graphes dynamiques avec intervalles. D'abord, des définitions formelles de ce type de graphes et de la sous-classe des graphes FIFO avec intervalles ont été présentées. Par la suite, deux variantes du problème du plus court chemin ont été étudiées pour ce type de graphe. Dans cette étude, l'accent a été mis sur le problème du plus court chemin suivant un critère d'optimisme. Une définition formelle de ce problème a été présentée et une méthode de résolution a été proposée. Puis, deux extensions du problème du plus court chemin suivant un critère optimiste ont été définies. La première est le problème du calcul du plus court chemin suivant un critère d'optimisme et un critère de déviation robuste. Plusieurs méthodes ont été proposées pour résoudre ce problème multicritère. La deuxième extension est l'utilisation d'un critère

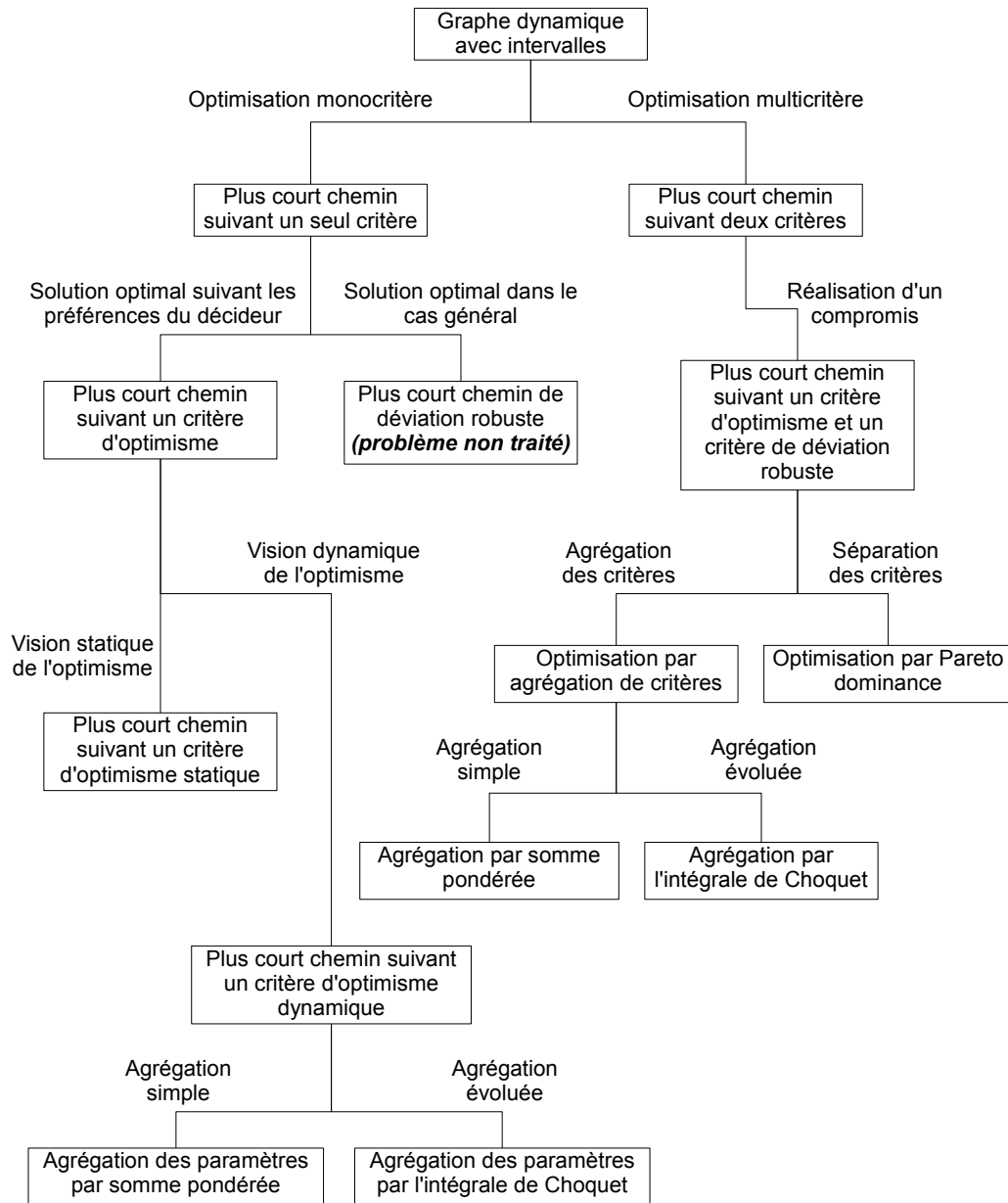


FIG. 3.10 – Synthèse des visions et des algorithmes proposés

d'optimisme dynamique pour le calcul du plus court chemin. Différents paramètres influant sur ce critère d'optimisme ont été étudiés.

Le modèle proposé dans ce travail pose différents problèmes difficiles à résoudre. Effectivement, nous avons prouvé que le problème du plus court chemin de déviation robuste est *NP-Difficile* et que le problème du plus court chemin suivant un critère d'optimisme n'est polynomial que dans le cas des graphes FIFO avec intervalles. Cependant il est très prometteur car il permet de modéliser de manière assez précise les réseaux routiers.

Conclusion et perspectives

L'objectif de cette thèse a été l'optimisation d'itinéraires pour des véhicules dans les réseaux routiers. Pour le calcul de ces itinéraires, deux aspects dynamiques ont été considérés. Le premier est le cas où la destination se déplace sur le réseau. Ce problème a été appelé problème de l'interception d'un mobile dans un graphe. Le deuxième est le cas où l'état du réseau est dynamique et incertain. Ce problème a été appelé problème du plus court chemin dans un graphe dynamique avec intervalles.

Pour l'interception d'un mobile dans un graphe, la problématique a été modélisée pour différentes situations et pour différents types de graphes. Les deux situations étudiées ont été l'interception d'un seul mobile par un seul poursuivant et l'interception de plusieurs mobiles par plusieurs poursuivants. Ces situations ont été résolues pour les graphes statiques et les graphes FIFO. Pour chaque type de graphes, un algorithme de résolution a été proposé et l'optimalité de la solution retournée a été démontrée. De plus, l'efficacité des algorithmes en termes de temps de calcul a été vérifiée par simulation.

Pour le deuxième problème, nous avons d'abord défini les graphes dynamiques avec intervalles et une sous-classe appelée graphes FIFO avec intervalles. L'avantage majeur de notre modèle par rapport aux modèles existants est qu'il est plus réaliste et qu'il très adapté au domaine du transport routier. En effet, il permet de modéliser deux aspects importants des problèmes de

transport. Le premier aspect est l'évolution au cours du temps de l'état du système. Le deuxième est l'incertitude qui peut exister sur son état. Pour les graphes dynamiques avec intervalles, le problème du plus court chemin a été défini de plusieurs manières. La première a été en tant que problème d'optimisation monocritère. La deuxième en tant que problème d'optimisation multicritère. Dans le cas de l'optimisation monocritère, deux formulations ont été d'abord proposées. La première formulation en tant que recherche du plus court chemin de déviation robuste. La deuxième formulation en tant que recherche du plus court chemin suivant un critère d'optimisme. Ensuite, la difficulté de ces formulations a été étudiée et nous avons prouvé que dans la majorité des cas ces problèmes sont *NP-Difficile*. Enfin, nous nous sommes concentrés sur le problème du plus court chemin suivant un critère d'optimisme pour proposer une solution dans le cas où le critère est statique et dans le cas où il est dynamique. Pour l'optimisation multicritère, nous avons tenté de rapprocher les deux formulations du plus court chemin étudiées dans le cas de l'optimisation monocritère.

Le présent travail de recherche offre de nombreuses perspectives possibles. D'une part pour le problème de l'interception d'un mobile dans un graphe. D'autre part pour le problème du plus court chemin dans un graphe dynamique avec intervalles.

Pour le problème de l'interception d'un mobile dans un graphe, une première extension possible est la résolution du problème pour les graphes non-FIFO. Ce type de graphes est assez délicat à traiter. En effet, le problème du plus court chemin classique est *NP-Difficile* pour ce type de graphes. Par suite, le calcul d'un itinéraire d'interception *optimal* est a priori *NP-Difficile*. D'où, il va falloir recourir à des métaheuristiques comme les algorithmes génétiques ou l'algorithme de séparation-évaluation afin de trouver une solution

au problème. Une deuxième extension possible est le calcul de l'itinéraire d'interception sans une connaissance a priori des poids des arcs. Effectivement, cette hypothèse a été considérée afin de développer des algorithmes hors-ligne pour calculer l'itinéraire d'interception. En supprimant cette hypothèse, il faut développer un algorithme en-ligne qui arrive à s'adapter à l'état dynamique du réseau routier. De plus, sans cette hypothèse, le problème de l'interception plusieurs-à-plusieurs devient plus complexe. En effet, l'affectation initiale d'un poursuivant à un mobile n'est plus forcément optimale et il se peut que l'état du système change de telle manière qu'il faut remettre en cause cette affectation.

Le problème du plus court chemin dans un graphe dynamique avec intervalles offre lui aussi plusieurs perspectives. Une première perspective est la résolution du problème du plus court chemin de déviation robuste. Une deuxième perspective est la comparaison des itinéraires calculés en utilisant les graphes dynamiques avec intervalles avec les graphes dynamiques déterministes. Cette comparaison doit avoir lieu dans un cadre proche de la réalité (mesures réelle, prédiction réelle) afin de vérifier l'adéquation des deux modèles à un réseau routier réel. Une dernière perspective consiste à traiter le problème de l'interception d'un mobile pour les graphes dynamiques avec intervalles. Étant donné que le problème du plus court chemin est *NP-Difficile* pour ce type de graphe, le problème de l'interception risque lui aussi d'être *NP-Difficile*.

Bibliographie

- [1] V.G. ADLAKHA. « An improved conditional Monte Carlo technique for stochastic shortest route problem ». *Management Science*, 32(10):1360 – 1367, 1986.
- [2] B.H. AHN et J.Y. SHIN. « Vehicle-Routing with Time Windows and Time-Varying Congestion ». *The Journal of the Operational Research Society*, 42(5):393 – 400, 1991.
- [3] M. AIGNER et M. FROMME. « A game of cops and robbers ». *Discrete Applied Mathematics*, 8:1 – 12, 1984.
- [4] S. ALPERN. « The Rendezvous Search Problem ». *SIAM Journal on Control and Optimization*, 33(3):673 – 683, 1995.
- [5] S. ALPERN. « Asymmetric rendezvous search on the circle ». *Dynamics and control*, 10:33 – 45, 2000.
- [6] S. ALPERN, V.J. BASTON et S. ESSEGAIER. « Rendezvous search on a graph ». *Applied Probabilities*, 36(1):223 – 231, 1999.
- [7] S. ALPERN et S. GAL. « Rendezvous search on the line with distinguishable players ». *SIAM Journal on Control and Optimization*, 33:1270 – 1276, 1995.
- [8] P. ANAND. *Foundations of Rational Choice Under Risk*. Éditions Oxford University Press, 2002.

- [9] E. ANDERSON et S. ESSEGAIER. « Rendezvous search on the line with indistinguishable players ». *SIAM Journal on Control and Optimization*, 33:1637 – 1642, 1995.
- [10] E. ANDERSON et S. FEKETE. « Asymmetric rendezvous on the plane ». Dans *Proceedings of 14th Annual ACM Symposium on Computational Geometry*, 1998.
- [11] E. ANDERSON et S. FEKETE. « Two-dimensional rendezvous search ». *Operations Research*, 49:107 – 118, 2001.
- [12] E.J. ANDERSON et R.R. WEBER. « The rendezvous problem on discrete locations ». *Journal of Applied Probability*, 28:839 – 851, 1990.
- [13] T. ANDREAE. « Note on a pursuit game played on graphs ». *Discrete Applied Mathematics*, 9:111 – 115, 1984.
- [14] T. ANDREAE. « On a pursuit game played on graphs for which a minor is excluded ». *Journal Combinatorial Theory Series B*, 41:37 – 47, 1986.
- [15] G. ANDREATTA, F. RICALDONE et L. ROMEO. « Exploring stochastic shortest path problems ». Rapport Technique, ATTI Giornale di Lavarò, 1985.
- [16] J.W. BANDER. « A heuristic search approach for a nonstationary stochastic shortest path problem with terminal cost ». *Transportation Science*, 36(2):218 – 230, 2002.
- [17] M. BARBEHENN. « A Note on the Complexity of Dijkstra Algorithm for Graphs with Weighted Vertices ». *IEEE Transactions on Computers*, 47(2):263, 1998.
- [18] J.F. BARD et J.E. BENNETT. « Arc reduction and path preference in stochastic acyclic networks ». *Management Science*, 37(2):198 – 215, 1991.

- [19] T. BARTZ-BEIELSTEIN, P. LIMBOURG, J. MEHNEN, K. SCHMITT, K.E. PARSOPOULOS et M.N. VRAHATIS. « Particle swarm optimizers for Pareto optimization with enhanced archiving techniques ». Dans *Proceedings of The 2003 Congress on Evolutionary Computation*, volume 3, pages 1780 – 1787, 2003.
- [20] F. BELKHOUCHE et B. BELKHOUCHE. « A control strategy for tracking-interception of moving objects using wheeled mobile robots ». Dans *Proceedings of 43rd IEEE Conference on Decision and Control*, pages 2129 – 2130, 2004.
- [21] F. BELKHOUCHE et B. BELKHOUCHE. « On the tracking and interception of a moving object by a wheeled mobile robot ». Dans *Proceedings of IEEE Conference on Robotics, Automation and Mechatronics*, pages 130 – 135, 2004.
- [22] Alain BERRO. « *Optimisation multiobjectif et stratégies d'évolution en environnement dynamique* ». PhD thesis, Université des Sciences Sociales Toulouse I, 2001.
- [23] J.A. BORGSTADT et N.J. FERRIER. « Interception of a projectile using a human vision-based strategy ». Dans *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3189 – 3196, 2000.
- [24] R.L. BREISCH. « An intuitive approach to speleotopology ». *Southwestern Cavers*, 6:72 – 78, 1967.
- [25] G.C. BUTTAZZO, B. ALLOTTA et F.P. FANIZZA. « Mousebuster : a robot for real-time catching ». *IEEE Control Systems Magazine*, 14(1):49 – 56, 1994.
- [26] Transports CANADA. « Le coût de la congestion urbaine au Canada ». <http://www.tc.gc.ca/medias/communiques/nat/2006/06->

h006f.htm, 2006.

- [27] I. CHABINI. « Algorithms for k-shortest path problems and other routing problems in time-dependent networks ». To appear in *Transportation Research Part B*.
- [28] I. CHABINI. « A new shortest path algorithm for discrete dynamic networks ». Dans *Proceedings of 8th IFAC Symposium on Transport Systems*, pages 551 – 556, 1997.
- [29] I. CHABINI. « Discrete dynamic shortest path problems in transportation applications : Complexity and algorithms with optimal run time ». *Transportation Research Record*, 1645:170 – 175, 1998.
- [30] I. CHABINI. « A new algorithm for shortest path problems in discrete dynamic networks ». Dans *Proceedings of the 8th IFAC Symposium on Transport Systems*, pages 551 – 556, 1998.
- [31] A.K. CHANDRA, D.C. KOZEN et L.J. STOCKMEYER. « Alternation ». *Journal of the ACM*, 28:114 – 133, 1981.
- [32] B.V. CHERKASSKY, A.V. GOLDBERG et T. RADZIK. « Shortest paths algorithms : Theory and experimental evaluation ». *Mathematical Programming*, 73:129–174, 1996.
- [33] Y. COLLETTE et P. SIARRY. *Optimisation Multiobjectif*. Éditions Eyrolles, 2002.
- [34] L. COOKE et E. HALSEY. « The Shortest Route Through a Network with Time-Dependent Internodal Transit Times ». *Journal of Mathematical Analysis and Applications*, 14:492 – 498, 1966.
- [35] E.A. CROFT, R.G. FENTON et B. BENHABIB. « Optimal rendezvous-point selection for robotic interception of moving objects ». *IEEE Transactions on Systems, Man and Cybernetics*, 28:192 – 204, 1998.

- [36] G.B. DANTZIG. *Linear Programming and Extensions*. Éditions Princeton University Press, 1963.
- [37] B.C. DEAN. « Shortest Paths in FIFO Time-Dependent Networks : Theory and Algorithms ». Rapport Technique, Massachusetts Institute Of Technology, 2001.
- [38] N. DEO et C.Y. PANG. « Shortest path algorithms : Taxonomy and Annotations ». *Networks*, 14:275 – 323, 1984.
- [39] R.B. DIAL. « Algorithm 360 : Shortest path forest with topological ordering ». *Communications of the ACM*, 12:632 – 633, 1969.
- [40] L.C. DIAS et J.N. CLIMACO. « Shortest path problems with partial information: models and algorithms for detecting dominance ». *European Journal of Operational Research*, 121:16 – 31, 2000.
- [41] E.W. DIJKSTRA. « A note on two problems in connexion with graphs ». *Numerische Mathematik*, 1:269 – 271, 1959.
- [42] A. DOLGUI et M.A. OULD-LOULY. « A model for supply planning under lead time uncertainty ». *International Journal of Production Economics*, 78(2):145 – 152, 2002.
- [43] A. DOLGUI et C. PRODHON. « Supply planning under uncertainties in MRP environments: A state of the art ». *Annual Reviews in Control*, 31(2):269 – 279, 2007.
- [44] S.E. DREYFUS. « An appraisal of some shortest-path algorithms ». *Operations Research*, 17:395 – 412, 1969.
- [45] M. ERICKSON, A. MAYER et J. HORN. « The Niche Pareto genetic algorithm 2 applied to the design of groundwater remediation systems ». Dans *Proceedings of International conference on Evolutionary multi-criterion optimization*, volume 1993, pages 681 – 695, 2001.

- [46] P. FLOCCHINI, G. PRENCIPE, N. SANTORO et P. WIDMAYER. « Gathering of asynchronous oblivious robots with limited visibility ». Dans *Proceedings of 18th Annual Symposium on Theoretical Aspects of Computer Science STACS' 2001*, pages 247 – 258, 2001.
- [47] H. FRANK. « Shortest paths in probabilistic graphs ». *Operations Research*, 17:583 – 599, 1969.
- [48] P. FRANKL. « Cops and robbers in graphs with large girth and Cayley graphs ». *Discrete Applied Mathematics*, 37:301 – 305, 1987.
- [49] P. FRANKL. « On a pursuit game on Cayley graphs ». *Combinatorica*, 7:67 – 70, 1987.
- [50] S. GAL. « Rendezvous search on the line ». *Operations Research*, 47:974 – 976, 1999.
- [51] G. GALLO et S. PALLOTTINO. « Shortest path algorithms ». *Annals of Operations Research*, 13:3 – 79, 1988.
- [52] R. GANS. « A control algorithm for automated pursuit ». Dans *Proceedings of IEEE International Conference on Control Applications*, pages 907 – 911, 1997.
- [53] S. GAO et I. CHABINI. « Optimal routing policy problems in stochastic time-dependent networks ». *Transportation Research Part B*, 40:93 – 122, 2006.
- [54] D.E. GOLDBERG. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Éditions Addison-Wesley Publishing, 1989.
- [55] A.S. GOLDSTEIN et E.M. REINGOLD. « The complexity of pursuit on a graph ». *Theoretical Computer Science*, 143:93 – 112, 1995.
- [56] P. GOODWIN et G. WRIGHT. *Decision Analysis for Management Judgment*. Éditions Chichester Wiley, 3rd édition, 2004.

- [57] M. GRABISCH. « The application of fuzzy integrals in multi-criteria decision making ». *European Journal of Operational Research*, 89:445 – 456, 1996.
- [58] M. GRABISCH. « k-order additive discrete fuzzy measures and their representation ». *Fuzzy Sets and Systems*, 92:167 – 189, 1997.
- [59] M. GRABISCH. « L'utilisation de l'intégrale de Choquet en aide multicritère à la décision ». *Newsletter of the European Working Group "Multicriteria Aid for Decisions"*, 3(14):5 – 10, 2006.
- [60] R.W. HALL. « The fastest path through a network with random time-dependent travel times ». *Transportation Science*, 20(3):182 – 188, 1986.
- [61] H.W. HAMACHER, S. RUZIKA et S.A. TJANDRA. « Algorithms for time-dependent bicriteria shortest path problems ». *Discrete optimization*, 3:238 – 254, 2006.
- [62] Y.O. HAMIDOUNE. « On a pursuit game on Cayley digraphs ». *European Journal of Combinatorics*, 8:289 – 295, 1987.
- [63] Sven Ove HANSSON. « Decision Theory : A Brief Introduction ». <http://www.infra.kth.se/soh/decisiontheory.pdf>, 1994.
- [64] M.M. HIZEM, E. CASTELAIN et A. TOGUYENI. « Interception d'un mobile dans un graphe ». Dans *Proceedings of JD-MACS*, 2007.
- [65] M.M. HIZEM, E. CASTELAIN et A. TOGUYENI. « Mobile Interception in a Graph : Different Case Studies ». Dans *Proceedings of 37th International Conference on Computers and Industrial Engineering*, 2007.
- [66] M.M. HIZEM, E. CASTELAIN et A. TOGUYENI. « Interception of a Moving Object in a FIFO Graph ». Dans *Proceedings of the 17th IFAC World Congress*, 2008.

- [67] M.M. HIZEM, E. CASTELAIN et A. TOGUYENI. « Utilisation du GPS pour l'interception d'un mobile dans un graphe FIFO ». Dans *Proceedings of CIFA 08*, 2008.
- [68] J.H. HOLLAND. *Adaptation in Natural and Artificial Systems*. Éditions University of Michigan Press, 1975.
- [69] J. HORN, N. NAFPLIOTIS et D.E. GOLDBERG. « A niched Pareto genetic algorithm for multiobjective optimization ». Dans *Proceedings of First IEEE Conference on Evolutionary Computation*, pages 82 – 87, 1994.
- [70] J.V. HOWARD. « Rendezvous search on the interval and circle ». *Operations Research*, 47(4):550 – 558, 1999.
- [71] T. HSU, O. KORBAA, R. DUPAS et G. GONCALVES. « Genetic algorithm for FMS cyclic scheduling ». Dans *Proceedings of MOSIM 03*, pages 519 – 525, 2003.
- [72] D. HUJIC, G. ZAK, E. CROFT, R.G. FENTON, J.K. MILLS et B. BENHABIB. « An active prediction, planning and execution system for interception of moving objects ». Dans *Proceedings of IEEE International Symposium on Assembly and Task Planning*, pages 347 – 352, 1995.
- [73] M. JAMALI. « Learning to Solve Stochastic Shortest Path Problems ». Rapport Technique, Sharif University of Technology, 2006.
- [74] E.L. JOHNSON. « On shortest paths and sorting ». Dans *Proceedings of 25th ACM Annual Conference*, pages 510 – 517, 1972.
- [75] J. KAMBUROWSKI. « A note on the Stochastic Shortest Route Problem ». *Operations Research*, 33(6):696 – 698, 1985.
- [76] M.A. KAMOUN. « Conception d'un système d'information pour l'aide au déplacement multimodal : Une approche multi-agents pour la re-

- cherche et la composition des itinéraires en ligne* ». PhD thesis, École Centrale de Lille, 2007.
- [77] O.E. KARASAN, M.C. PINAR et H. YAMAN. « The robust shortest path problem with interval data ». *Computers & Operations Research*, 2002.
- [78] D.E. KAUFMAN et R.L. SMITH. « Fastest Paths in Time-Dependent Networks for Intelligent Vehicle-Highway Systems Application ». *Journal of Intelligent Transportation Systems*, 1(1):1 – 11, 1993.
- [79] S. KIRKPATRICK, C.D. GELATT et M.P. VECCHI. « Optimization by simulated annealing ». *Science*, 220(4598):671 – 680, 1983.
- [80] E. KLAFSZKY. « Determination of shortest path in a network with time-dependent edge-lengths ». *Optimization*, 3(4):255 – 257, 1972.
- [81] P. KOUVELIS et G. YU. *Robust Discrete Optimization and its Applications*. Éditions Kluwer Academic Publishers, 1997.
- [82] H.W. KUHN. « The Hungarian Method for the assignment problem ». *Naval Research Logistic Quarterly*, 2:83 – 97, 1955.
- [83] H.W. KUHN. « Variants of the Hungarian method for assignment problems ». *Naval Research Logistic Quarterly*, 3:253 – 258, 1956.
- [84] A.H. LAND et A.G. DOIG. « An Automatic Method of Solving Discrete Programming Problems ». *Econometrica*, 28(3):497 – 520, 1960.
- [85] M. LEI et B.K. GHOSH. « Visually guided robotic tracking and grasping of a moving object ». Dans *Proceedings of IEEE 32nd Conference on Decision and Control*, pages 1604 – 1609, 1993.
- [86] W. LIM et S. ALPERN. « Minimax rendezvous on the line ». *SIAM Journal on Control and Optimization*, 34:1650 – 1665, 1996.
- [87] J. LIN, A.S. MORSE et B.D.O. ANDERSON. « The multi-agent rendezvous problem - the asynchronous case ». Dans *Proceedings of 43rd*

- IEEE Conference on Decision and Control*, volume 2, pages 1926 – 1931, 2004.
- [88] R.P. LOUI. « Optimal paths in graphs with stochastic or multidimensional weights ». *Communications of the ACM*, 26(9):670 – 676, 1983.
- [89] M. MAAMOUN et H. MEYNIEL. « On a game of policemen and robber ». *Discrete Applied Mathematics*, 17:307 – 309, 1987.
- [90] M.K. McBEATH, D.M. SHAITER et M.K. KAISER. « How baseball outfielders determine where to run to catch fly balls ». *Science*, 268(5210):569 – 573, 1995.
- [91] P. MCLEOD et Z. DIENES. « Running to catch the ball ». *Nature*, 362(6415), 1993.
- [92] M.D. MIKESELL et R.J. CIPRA. « Development of a real-time intelligent robotic tracking system ». Dans *Proceedings of ASME 23rd Mechanism Conference*, pages 213 – 222, 1994.
- [93] E. MILLER-HOOKS et B. YANG. « The Impact of Travel Time Models on the Quality of Real-Time Routing Instructions ». Dans *Proceedings of 82nd annual meeting of the Transportation Research Board*, pages 1 – 20, 2003.
- [94] E.D. MILLER-HOOKS et H.S. MAHMASSANI. « Least possible time paths in stochastic time-varying networks ». *Computers & Operations Research*, 25:1107 – 1125, 1998.
- [95] E.D. MILLER-HOOKS et H.S. MAHMASSANI. « Least expected time paths in stochastic time-varying transportation networks ». *Transportation Science*, 34(2):198 – 215, 2000.
- [96] P.B. MIRCHANDANI et H. SOROUSH. « Optimal paths in probabilistic networks: A case with temporary preferences ». *Computers & Operations Research*, 12:365 – 383, 1985.

- [97] R. MONTEMANNI et L.M. GAMBARDELLA. « An exact algorithm for the robust shortest path problem with interval data ». *Computers & Operations Research*, 31(10):1667 – 1680, 2004.
- [98] R. MONTEMANNI, L.M. GAMBARDELLA et A.V. DONATI. « A branch and bound algorithm for the robust shortest path problem with interval data ». *Operations Research Letters*, 32(3):225 – 232, 2004.
- [99] T. MUROFUSHI et S. SONEDA. « Techniques for reading fuzzy measures (III) : interaction index ». Dans *Proceedings of 9th Fuzzy System Symposium*, pages 693 – 696, 1993.
- [100] I. MURTHY et S. SARKAR. « A relaxation-based pruning technique for a class of stochastic shortest path problems ». *Transportation science*, 30(3):220 – 236, 1996.
- [101] D.W. NORTH. « A tutorial introduction to decision theory ». *IEEE Transactions on Systems Science and Cybernetics*, 4(3):200 – 210, 1968.
- [102] R. NOWAKOWSKI et P. WINKLER. « Vertex to vertex pursuit in a graph ». *Discrete Mathematics*, 43:235 – 239, 1983.
- [103] A. ORDA et R. ROM. « Shortest Path and Minimum Delay Algorithms in Networks with Time-Dependent Edge Length ». *Journal of the ACM*, 37(3):607 – 625, 1990.
- [104] A. ORDA et R. ROM. « Minimum weight paths in time-dependent network ». *Networks*, 21(3):295 – 320, 1991.
- [105] M.A. OULD-LOULY et A. DOLGUI. « Supply planning optimization under uncertainties ». *International Journal of Agile Manufacturing*, 5:17 – 26, 2002.
- [106] S. PALLOTTINO et M.G. SCUTELLA. « Shortest path algorithms in Transportation models : classical and innovative aspects ». Dans *Pro-*

- ceedings of the Equilibrium and Advanced Transportation Modelling Colloquium*, 1998.
- [107] T.H. PARK et B.H. LEE. « An approach to robot motion analysis and planning for conveyor tracking ». *IEEE Transactions on Systems, Man and Cybernetics*, 22:378 – 384, 1992.
- [108] T.D. PARSONS. « Pursuit-evasion in a graph ». *Lecture Notes in Mathematics*, pages 426 – 441, 1976.
- [109] T.D. PARSONS. « The search number of a connected graph ». Dans *Proceedings of 9th Southeastern Conference on Combinatorics*, pages 549 – 554, 1978.
- [110] D. PRETOLANI. « A directed hypergraph model for random time-dependent shortest paths ». *European Journal of Operational Research*, 123:315 – 324, 2000.
- [111] A. QUILLIOT. « *Études de quelques problèmes sur les graphes et hypergraphes et applications à la théorie des jeux à information complète* ». PhD thesis, Université de Paris VI, 1978.
- [112] A. QUILLIOT. « Discrete pursuit games ». Dans *Proceedings of 13th Conference on Graphs and Combinatorics*, 1982.
- [113] A. QUILLIOT. « A short note about pursuit games played on a graph with a given genus ». *Journal of combinatorial theory Series B*, 38:89 – 92, 1985.
- [114] D.M. RASTEIRO et AB. ANJO. « Metaheuristics for stochastic shortest path problem ». Dans *Proceedings of 4th MetaHeuristics International Conference (MIC2001)*, 2001.
- [115] J.H. REIF. « Universal games of incomplete information ». Dans *Proceedings of 11th Annual ACM Symposium on Theory of Computing*, pages 288 – 308, 1979.

- [116] J.H. REIF. « The complexity of two-player games of incomplete information ». *Journal of Computer and Systems Sciences*, 29(2):274 – 301, 1984.
- [117] T. SCHELLING. *The Strategy of Conflict*. Éditions Oxford University Press, 1960.
- [118] D. SCHULZ, W. BURGARD, D. FOX et A.B. CREMERS. « Tracking Multiple Moving Objects with a Mobile Robot ». Dans *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR' 01)*, volume 1, pages 371 – 377, 2001.
- [119] L.S. SHAPLEY. A value for n-person games. Dans H.W. KUHN et A.W. TUCKER, éditeurs, *Proceedings of Contributions to the Theory of Games*, volume 2, pages 307 – 317. Éditions Princeton University Press, 1953.
- [120] C.E. SIGAL, A.A.B. PRITSKER et J.J. SOLBERG. « Stochastic shortest route problem ». *Operations Research*, 28(5):1122 – 1129, 1980.
- [121] A. SULUH, T. SUGAR et M. MCBEATH. « Spatial navigational principles: Applications to mobile robotics ». Dans *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1689 – 1694, 2001.
- [122] A.A. TAIROU. *Analyse et décisions financières*. Éditions L'Harmattan, 2006.
- [123] P. TETALI et P. WINKLER. « Simultaneous reversible Markov chains ». *Bolyai Society Mathematical Studies*, 1:433 – 451, 1993.
- [124] K.M. THOMAS, G. SUGAR et M.K. MCBEATH. « Perceptual Navigation Strategy: A Unified Approach to Interception of Ground Balls and Fly Balls ». Dans *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 3461 – 3466, 2003.

- [125] L. THOMAS. « Finding your kids when they are lost ». *The Journal of the Operational Research Society*, 43(3):637 – 639, 1992.
- [126] D. Van VLIET. « Improved shortest path algorithms for transport networks ». *Transportation Research*, 12:7 – 20, 1978.
- [127] B.M. WAXMAN. « Routing of Multipoint Connections ». *IEEE Journal on Selected Areas in Communications*, 6(9):1617 – 1622, 1988.
- [128] A. WREN et D.O. WREN. « A genetic algorithm for public transport driver scheduling ». *Computers & Operations Research*, 22(1):101 – 110, 1995.
- [129] R.R. YAGER. « On weighted median aggregation operators in multi-criteria decision making ». *IEEE Transaction on Systems, Man and Cybernetics*, 18:183 – 190, 1988.
- [130] G. YU et J. YANG. « On the robust shortest path problem ». *Computers & Operations Research*, 25(6):457 – 468, 1998.
- [131] H. ZGAYA. « Conception et optimisation distribué d'un système d'information d'aide à la mobilité urbaine : Une approche multi-agent pour la recherche et la composition des services liés au transport ». PhD thesis, École Centrale de Lille, 2007.
- [132] K. ZIDI. « Système Interactif d'Aide au Déplacement Multimodal ». PhD thesis, École Centrale de Lille, 2006.
- [133] P. ZIELINSKI. « The computational complexity of the relative robust shortest path problem with interval data ». *European Journal of Operational Research*, 158:570 – 576, 2004.
- [134] E. ZITZLER, M. LAUMANN et L. THIELE. « SPEA2: Improving the Strength Pareto Evolutionary Algorithm ». Rapport Technique, Swiss Federal Institute of Technology (ETH) Zurich, 2001.

Annexe A

Étude d'un exemple d'optimisation multicritère par l'intégrale de Choquet

Pour les besoins de production, le gérant d'une PME doit rédiger un appel d'offre pour l'acquisition de machines. Pour départager les entreprises candidates, trois critères sont retenus : la fiabilité des machines, le degré de compétence technique de l'entreprise et le prix. À cause des contraintes de production auxquelles il est soumis, le gérant doit favoriser l'entreprise qui propose les machines les plus fiables et possédant la meilleure compétence technique. Néanmoins, il souhaite que le prix ne soit pas exorbitant. Pour classer les entreprises candidates, il opte pour la technique d'agrégation des critères par somme pondérée. Par conséquent, il attribue la pondération 0.4 au critère fiabilité et au critère compétence technique et il attribue la pondération 0.2 au critère prix. Trois entreprises répondent à l'appel d'offre. Le gérant assigne un score à chaque entreprise suivant chaque critère. Plus le prix est faible plus le score est important et plus la fiabilité ou le degré de

compétence est élevé, plus son score est important. Les scores de chaque entreprise et le résultat de l'agrégation par somme pondérée sont indiqués dans le tableau A.1. Dans ce tableau, le critère fiabilité est noté F, le critère compétence technique est noté CT, le critère prix est noté P et le score de l'entreprise suivant la somme pondérée est noté F_{sp} .

	F	CT	P	F_{sp}
Entreprise 1	18	16	10	15.6
Entreprise 2	10	12	18	12.4
Entreprise 3	14	15	15	14.6

TAB. A.1 – *Scores des entreprises et résultat de l'agrégation par somme pondérée*

Suivant la méthode utilisée par le gérant, l'entreprise 1 remporte l'appel d'offre. Cependant le gérant n'est pas tout à fait satisfait des résultats. En effet, l'entreprise 1 propose la meilleure fiabilité et possède la meilleure compétence technique mais ses prix sont excessifs. Par contre, l'entreprise 3 propose une fiabilité correcte, elle possède une bonne compétence technique et ses prix sont très abordables. Par conséquent, le gérant choisit d'utiliser l'intégrale de Choquet pour départager les entreprises. Il formalise ses préférences comme suit :

1. Les critères fiabilité et compétence technique sont les plus importants ;
2. Une entreprise qui possède une bonne compétence technique propose généralement des machines assez fiables et inversement. D'où, il ne faut pas favoriser excessivement les entreprises qui proposent une bonne fiabilité et qui possèdent de bonnes compétences techniques simultanément ;

3. Il est rare de trouver des entreprises qui proposent des machines fiables à un prix abordable. D'où il faut favoriser ce type d'entreprises ;
4. Il est rare de trouver des entreprises qui possèdent de bonnes compétences techniques et proposent des prix abordables. D'où il faut favoriser ce type d'entreprises.

Les trois dernières préférences peuvent être reformulées en termes d'interactions entre les critères comme suit :

- Il existe une synergie positive entre la fiabilité et la compétence technique ;
- Il existe une synergie négative entre le prix et la fiabilité ;
- Il existe une synergie négative entre le prix et la compétence technique ;

Par suite, l'ensemble des préférences du gérant est modélisé à l'aide de la mesure floue μ comme suit :

1. $\mu(\{F\}) = \mu(\{CT\}) = 0.45$, $\mu(\{P\}) = 0.3$ (importance relative des critères)
2. $\mu(\{F, CT\}) = 0.5 > \mu(\{F\}) + \mu(\{CT\})$ (synergie positive ou redondance) ;
3. $\mu(\{P, F\}) = 0.9 > \mu(\{P\}) + \mu(\{F\})$ (synergie négative ou complémentarité) ;
4. $\mu(\{P, CT\}) = \mu(\{P\}) + \mu(\{CT\}) = 0.55$ (synergie négative ou complémentarité).

Le calcul de la fonction objectif en utilisant l'intégrale de Choquet, noté F_μ , pour les différentes entreprises est réalisé comme suit :

Entreprise 1 D'abord, il faut classer les scores suivant un ordre croissant

$$10 < 16 < 18 \implies \{P, CT, F\}$$

la fonction objectif pour l'entreprise 1 est :

$$\begin{aligned} F_{\mu}(E1) &= (10 - 0) \mu(\{P, CT, F\}) + (16 - 10) \mu(\{CT, F\}) + (18 - 16) \mu(\{F\}) \\ &= 10 + 6 \times 0.5 + 2 \times 0.45 \\ &= 13.9 \end{aligned}$$

Entreprise 2 Le classement des critères est le suivant :

$$10 < 12 < 18 \implies \{F, CT, P\}$$

D'où la fonction objectif pour l'entreprise 2 est :

$$\begin{aligned} F_{\mu}(E2) &= (10 - 0) \mu(\{F, CT, P\}) + (12 - 10) \mu(\{CT, P\}) + (18 - 12) \mu(\{P\}) \\ &= 10 + 2 \times 0.9 + 6 \times 0.3 \\ &= 13.6 \end{aligned}$$

Entreprise 3 Le classement des critères est le suivant :

$$14 < 15 < 15 \implies \{F, CT, P\}$$

D'où la fonction objectif pour l'entreprise 3 est :

$$\begin{aligned} F_{\mu}(E3) &= (14 - 0) \mu(\{F, CT, P\}) + (15 - 14) \mu(\{CT, P\}) + (15 - 15) \mu(\{P\}) \\ &= 14 + 1 \times 0.9 + 0 \times 0.3 \\ &= 14.9 \end{aligned}$$

Par suite, l'entreprise 3 se classe première et remporte l'appel d'offre. Les résultats sont réunis dans le tableau A.2.

	F	CT	P	F_{sp}	F_{μ}
Entreprise 1	18	16	10	15.6	13.9
Entreprise 2	10	12	18	12.4	13.6
Entreprise 3	14	15	15	14.6	14.9

TAB. A.2 – Scores des entreprises et résultat de l'agrégation par somme pondérée et par l'intégrale de Choquet

Annexe B

Étude d'un exemple d'utilisation des critères de décision dans un environnement incertain

Soit un vendeur ambulant qui, au début de la journée, doit choisir les produits avec lesquels il va remplir son chariot. Il a le choix entre quatre types de produits : glaces (produit 1), boissons (produit 2), journaux (produit 3) ou jouets (produit 4). Supposons que l'unique facteur influant sur ses ventes est le facteur climatique. Les états qui peuvent se produire sont : beau temps (état 1), temps couvert (état 2) ou temps pluvieux (état 3). Dans la suite, le choix du type de produits i est représenté par l'action a_i et l'occurrence de l'état climatique j est représentée par l'état e_j . Les résultats des actions suivant les états sont présentés dans le tableau B.1.

Pour déterminer la décision à prendre, les critères quantitatifs et probabilistes sont appliqués à ce cas.

Action \ Etat	e_1 (Beau temps)	e_2 (Temps couvert)	e_3 (Temps pluvieux)
a_1 (Glaces)	800	200	-100
a_2 (Boissons)	250	700	-50
a_3 (Journaux)	150	400	500
a_4 (Jouets)	600	100	290

TAB. B.1 – *Résultats des actions suivant les états*

B.1 Critères quantitatifs

B.1.1 Critère de Laplace

Pour ce critère, il faut calculer la moyenne des résultats de chaque action :

- $V(a_1) = \frac{1}{3}(800 + 200 - 100) = \frac{900}{3} = 300$
- $V(a_2) = \frac{1}{3}(250 + 700 - 50) = \frac{900}{3} = 300$
- $V(a_3) = \frac{1}{3}(150 + 400 + 500) = \frac{1050}{3} = 350$
- $V(a_4) = \frac{1}{3}(600 + 100 + 290) = \frac{990}{3} = 330$

Par suite, $V(a_3) > V(a_4) > V(a_2) \geq V(a_1)$. En conclusion, d'après le critère de Laplace, c'est la décision a_3 (Journaux) qui est choisie.

B.1.2 Critère de Wald ou du MaxiMin

Pour ce critère, il faut déterminer le résultat minimal pour chaque action :

- $V(a_1) = -100$
- $V(a_2) = -50$
- $V(a_3) = 150$
- $V(a_4) = 100$

Par suite, $V(a_3) > V(a_4) > V(a_2) > V(a_1)$. En conclusion, d'après le critère de Wald, c'est la décision a_3 (Journaux) qui est choisie.

B.1.3 Critère du MaxiMax

Pour ce critère, il faut déterminer le résultat maximal pour chaque action :

- $V(a_1) = 800$
- $V(a_2) = 700$
- $V(a_3) = 500$
- $V(a_4) = 600$

Par suite, $V(a_1) > V(a_2) > V(a_4) > V(a_3)$. En conclusion, d'après le critère du MaxiMax, c'est la décision a_1 (Glaces) qui est choisie.

B.1.4 Critère d'Hurwitz

Pour ce critère un facteur d'optimisme α est considéré. Supposons que $\alpha = 0.7$. D'où les valeurs des actions sont les suivantes :

- $V(a_1) = 0.7 \times 800 + 0.3 \times (-100) = 560 - 30 = 530$
- $V(a_2) = 0.7 \times 700 + 0.3 \times (-50) = 490 - 15 = 475$
- $V(a_3) = 0.7 \times 500 + 0.3 \times 150 = 350 + 45 = 395$
- $V(a_4) = 0.7 \times 600 + 0.3 \times 100 = 420 + 30 = 450$

Par suite, $V(a_1) > V(a_2) > V(a_4) > V(a_3)$. En conclusion, d'après le critère d'Hurwitz, c'est la décision a_1 (Glaces) qui est choisie.

B.1.5 Critère de Savage ou du MiniMax

Pour ce critère, il faut calculer le regret maximal pour chaque action. Dans ce but, il faut d'abord déterminer le résultat maximal pour chaque état :

- $\max_{k \in \{1,2,\dots,n\}} \{R(a_k, e_1)\} = 800$

- $\max_{k \in \{1,2,\dots,n\}} \{R(a_k, e_2)\} = 700$
- $\max_{k \in \{1,2,\dots,n\}} \{R(a_k, e_3)\} = 500$

Ensuite une matrice appelée matrice des regrets est établie. Elle est décrite par le tableau B.2

Action \ Etat	e_1	e_2	e_3
a_1	$800 - 800 = 0$	$700 - 200 = 500$	$500 - (-100) = 600$
a_2	$800 - 250 = 550$	$700 - 700 = 0$	$500 - (-50) = 550$
a_3	$800 - 150 = 650$	$700 - 400 = 300$	$500 - 500 = 0$
a_4	$800 - 600 = 200$	$700 - 100 = 600$	$500 - 290 = 210$

TAB. B.2 – Matrice des regrets

Puis, pour chaque action, le regret maximal est déterminé :

- $V(a_1) = 600$
- $V(a_2) = 550$
- $V(a_3) = 650$
- $V(a_4) = 600$

Par suite, $V(a_3) > V(a_1) \geq V(a_4) > V(a_2)$. En conclusion, d'après le critère de Savage, c'est la décision a_2 (Boissons) qui est choisie.

B.2 Critères probabilistes

Pour cette partie, nous supposons que la probabilité de chacun des événements est définie comme suit :

- $P(e_1) = 0.5$
- $P(e_2) = 0.1$
- $P(e_3) = 0.4$

B.2.1 Critère de Pascal

Pour ce critère il faut calculer l'espérance de chaque action :

- $V(a_1) = E(a_1) = 800 \times 0.5 + 200 \times 0.1 + (-100) \times 0.4 = 380$
- $V(a_2) = E(a_2) = 250 \times 0.5 + 700 \times 0.1 + (-50) \times 0.4 = 175$
- $V(a_3) = E(a_3) = 150 \times 0.5 + 400 \times 0.1 + 500 \times 0.4 = 315$
- $V(a_4) = E(a_4) = 600 \times 0.5 + 100 \times 0.1 + 290 \times 0.4 = 426$

Par suite, $V(a_4) > V(a_1) > V(a_3) > V(a_2)$. En conclusion, d'après le critère de Pascal, c'est la décision a_4 (Jouets) qui est choisie.

B.2.2 Critère de Markowitz

Pour ce critère, il faut, en plus de l'espérance, calculer la variance pour chaque action :

- $\sigma^2(a_1) = (800 - 380)^2 \times 0.5 + (200 - 380)^2 \times 0.1 + (-100 - 380)^2 \times 0.4$
 $\Rightarrow \sigma(a_1) = 428.48$
- $\sigma^2(a_2) = (250 - 175)^2 \times 0.5 + (700 - 175)^2 \times 0.1 + (-50 - 175)^2 \times 0.4$
 $\Rightarrow \sigma(a_1) = 225$
- $\sigma^2(a_3) = (150 - 315)^2 \times 0.5 + (400 - 315)^2 \times 0.1 + (500 - 315)^2 \times 0.4$
 $\Rightarrow \sigma(a_1) = 167.4$
- $\sigma^2(a_4) = (600 - 426)^2 \times 0.5 + (100 - 426)^2 \times 0.1 + (290 - 426)^2 \times 0.4$
 $\Rightarrow \sigma(a_1) = 182.1$

Ensuite, les $V(a_i)$ est obtenu comme suit :

- $V(a_1) = \frac{E(a_1)}{\sigma a_1} = 0.88$
- $V(a_2) = \frac{E(a_2)}{\sigma a_2} = 0.77$
- $V(a_3) = \frac{E(a_3)}{\sigma a_3} = 1.88$

$$- V(a_4) = \frac{E(a_4)}{\sigma a_4} = 2.33$$

Par suite, $V(a_4) > V(a_3) > V(a_1) > V(a_2)$. En conclusion, d'après le critère de Markowitz, c'est la décision a_4 (Jouets) qui est choisie.

Annexe C

Algorithme d'interception de référence

L'objectif de cette annexe est de présenter l'algorithme d'interception de référence qui est utilisé dans le chapitre 2. Pour cet algorithme, l'ensemble *Intercep* est défini comme étant l'ensemble des nœuds d'interception. Pour calculer le nœud d'interception optimal, il suffit de trouver l'ensemble *Intercep* et les $PCCh_{MP}(u)$ avec $u \in Intercep$. Ensuite, le nœud d'interception optimal *OPT* est le nœud de l'ensemble *Intercep* tel que $C_o(It(MO, u))$ est minimal. Dans cette optique, l'algorithme de Dijkstra est utilisé. En effet, l'algorithme de Dijkstra permet de calculer le chemin de coût minimal d'un nœud source vers tous les autres nœuds du graphe. La figure C.1 décrit cet algorithme.

```

1  //Initialiser l'ensemble Intercep
2  Intercep =  $\emptyset$ 
3  Calculer les plus courts chemins de InitMP vers tous les nœuds de  $G_p$ 
4  Pour  $\forall u \in It(MO) \cap G_p$  faire
5      Si  $C_p(PCC_{MP}(u)) \leq C_o(It(MO, u))$  Alors
6          Intercep = Intercep  $\cup \{u\}$ 
7      Fin Si
8  Fin Pour
9  Si Intercep =  $\emptyset$  Alors ÉCHEC
10 Sinon Déterminer le nœud OPT  $\in$  Intercep tel que  $C_o(It(MO, OPT))$ 
11 est minimal

```

FIG. C.1 – *Algorithme d'interception de référence*

Résumé : L'objectif de cette thèse est le développement d'algorithmes et de modèles permettant l'optimisation d'itinéraires dans les réseaux routiers. Dans un premier temps, ce travail de recherche étudie le problème de l'interception d'un mobile dans un graphe. Dans ce contexte, l'objectif est de calculer un itinéraire optimal permettant de rejoindre une cible mobile dont la trajectoire est connue. Cette problématique est traitée pour plusieurs situations (un poursuivant/un objectif et plusieurs poursuivants/plusieurs objectifs) et pour plusieurs types de graphes (graphes statiques et graphes FIFO). Pour chaque cas, un algorithme de résolution est proposé et l'optimalité du résultat qu'il retourne est démontrée. De plus, un ensemble de simulations est réalisé afin de vérifier l'efficacité des algorithmes en termes de temps de calcul. Dans un deuxième temps, une nouvelle classe de graphes dynamiques est définie : les graphes dynamiques avec intervalles. La particularité de ces graphes est que le poids de chaque arc dépende du temps et qu'il est représenté par un intervalle. Pour ce nouveau type de graphes, le problème du plus court chemin est étudié. Ce problème peut être vu soit en tant que problème d'optimisation monocritère soit en tant que problème d'optimisation multicritère. Pour chaque cas, le problème est formulé et des approches pour la résolution sont proposées.

Mots-clés : Transport, Problème du plus court chemin, Interception, Graphes dynamiques, Graphes avec des intervalles.

Abstract : This thesis aims to develop algorithms and models for optimizing itineraries in road networks. The first part of this work treats the problem of intercepting a mobile in a graph. In this context, the goal is to compute the optimal path to reach a moving target with a known itinerary. This problem is studied for different situations (one pursuer/one target and several pursuers/several targets) and for different types of graphs (time-independent graphs and FIFO graphs). For each case, an algorithm is suggested and its optimality is proven. Moreover, simulations are conducted to check the algorithms efficiency in terms of execution time. In the second part of this thesis, a new class of time-dependent graphs is defined : the time-dependent interval graphs. The distinctive feature of these graphs is that the edge weight is time-dependent and it is defined by an interval. For this new class of graphs, the shortest path problem is studied. This problem can be viewed either as mono-objective optimization problem or as a multi-objective optimization problem. For each case, the problem is formulated and approaches for resolution are proposed.

Key-words : Transport, Shortest path problem, Interception, Interval graphs, Time-dependent graphs.