UNIVERSITÉ DE NANTES
UFR SCIENCES ET TECHNIQUES

ÉCOLE DOCTORALE

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE MATHÉMATIQUES

2011

Thèse de Doctorat de l'Université de Nantes

Spécialité : INFORMATIQUE

Présentée et soutenue publiquement par

Hugo FOUCHAL

le 20 octobre 2011

LINA

Optimisation de l'intégrale de Choquet pour le calcul de plus courts chemins multi-objectifs préférés

Jury

Président	:	Pr. Éric PINSON, Professeur	Université catholique de l'ouest
Rapporteurs	:	Pr. Patrice PERNY, Professeur	Université Paris 6
		Pr. Bernard FORTZ, Professeur	Université libre de Bruxelles
Examinateurs	:	Mme. Sabine RANDRIAMASY, Ingénieure de Recherche	Alcatel-Lucent
		M. Christophe LABREUCHE, Ingénieur de Recherche	Thales
		M. Anthony PRZYBYLSKI, Maitre de conférences	Université de Nantes

Directeur de thèse : Pr. Xavier GANDIBLEUX
Co-encadrant de thèse : M. Fabien LEHUÉDÉ
Laboratoire : LABORATOIRE D'INFORMATIQUE DE NANTES ATLANTIQUE.
2, rue de la Houssinière, BP 92 208 – 44 322 Nantes, CEDEX 3.

N° ED 0366-...

OPTIMISATION DE L'INTÉGRALE DE CHOQUET POUR LE CALCUL DE PLUS COURTS CHEMINS MULTI-OBJECTIFS PRÉFÉRÉS

Optimisation of the Choquet integral to compute multi-objective preferred shortest paths

Hugo FOUCHAL

⋈

favet neptunus eunti

Université de Nantes

Hugo FOUCHAL

Optimisation de l'intégrale de Choquet pour le calcul de plus courts chemins multi-objectifs préférés

x+166 p.

Ce document a été préparé avec LaTeX2e et la classe `these-LINA` version v. 1.30 de l'association de jeunes chercheurs en informatique LoGiN, Université de Nantes. La classe `these-LINA` est disponible à l'adresse :

`http://login.lina.sciences.univ-nantes.fr/`

Impression : sommaire.tex – 1/12/2011 – 12:56

Révision pour la classe : these-LINA.cls,v 1.30 2005/08/16 17:01:41 mancheron Exp

Résumé

Dans cette thèse nous nous intéressons à la prise en compte des préférences du décideur pour le calcul des plus courts chemins multi-objectif. Les préférences du décideur sont modélisées a priori par une fonction d'utilité basée sur une intégrale de Choquet. Plutôt que de calculer l'ensemble des solutions efficaces puis de sélectionner une solution optimale selon cette fonction d'utilité, nous cherchons à directement construire une solution efficace optimale au regard des préférences. Le travail réalisé développe une règle de coupe, propose une relation de dominance et valide ces propositions sur un problème de routage multi-objectif rencontré dans les réseaux informatiques. La règle de coupe permet de réduire l'espace de recherche en calculant une somme pondérée bornant la fonction d'utilité. Plusieurs méthodes pour le calcul d'une telle somme pondérée sont proposées, analysées et comparées. Nous définissons une nouvelle relation de dominance capable de prendre en compte la fonction d'utilité et raffinant la relation de dominance de Pareto. Des conditions suffisantes de cette dominance sont proposées, elles permettent de réduire le nombre de solutions calculées et les temps de calcul par rapport aux méthodes existantes. Nous montrons que les préférences d'un administrateur réseau, dans le cadre du routage multi-objectif au niveau IP, peuvent être modélisées à l'aide d'une fonction d'utilité basée sur l'intégrale de Choquet. Les résultats précédents sont appliqués pour la résolution de ce problème.

Mots-clés : optimisation combinatoire multi-objectif ; plus courts chemins ; préférences ; intégrale de Choquet ; algorithme d'étiquetage.

Abstract

The purpose of this thesis is to handle decision maker preferences for computing multi-objective shortest paths. Decision maker preferences are modeled a priori with a utility function based on a Choquet integral. Instead of computing the set of efficient solutions and choosing afterward an optimal solution according to the utility function, we aim to directly compute an efficient optimal solution satisfying decision maker preferences. In this thesis we develop a pruning rule, propose a dominance relation and valid these propositions in a multi-objective routing problem for computer networks.

The pruning rule allows to reduce the search space by computing a weighted sum that bounds the utility function. Several methods for computing such weighted sum are proposed, analyzed and compared. We define a new dominance relation that considers the utility function and refines Pareto dominance. Sufficient conditions for this dominance are proposed, they allow to reduce the number of solutions computed and computing time compared to existing methods. We highlight that network administrator preferences, for multi-objectif routing in IP network, can be modeled with a utility function based on the Choquet integral. Previous results are applied for solving this problem.

Keywords: multi-objective combinatorial optimisation; shortest paths; preferences; Choquet integral; labeling algorithm.

Remerciements

Je remercie mes deux encadrants pour m'avoir aidé et formé durant ces trois difficiles années. Je remercie aussi les membres des deux équipes SLP et OPTI pour m'avoir intégré en leur sein et soutenu durant cette période ainsi que les différents doctorants avec qui j'ai eu le plaisir de discuter.

Sommaire

Introduction

Dans le cadre de cette thèse, nous nous somme intéressés à l'optimisation d'une fonction d'utilité dans un problème de plus court chemin multi-objectif. Cette approche est utilisée pour résoudre des problèmes dans le monde réel, notamment pour routage dans les réseaux internet [XN99] ou pour le transport dans les réseaux routiers [SNB11]. Différentes contributions sont proposées dans le cadre de cette problématique et une discussion de l'application de ces contributions à un problème de routage dans les réseaux internet est développée.

Le problème de plus court chemin

Le problème de plus court chemin est un des problèmes d'optimisation combinatoire les plus étudiés. De nombreux problèmes applicatifs peuvent être modélisés sous la forme de ce problème combinatoire. Ce problème consiste à construire un chemin minimisant un objectif dans un graphe modélisant un réseau internet, de transport en commun ou routier. Pour un réseau internet, les routeurs du réseau sont modélisés par les nœuds du graphe et les liens du réseau par les arcs. Il s'agit d'un des premiers problèmes d'optimisation combinatoire étudiés dans la littérature. Ainsi une formalisation de ce problème est proposée par Euler au $18^{\text{ème}}$ siècle [Eul36]. De nombreux travaux ont été réalisés sur ce problème depuis les années 60. Les *algorithmes d'étiquetage* (ou de labeling), dont l'algorithme de Dijkstra [Dij59] est l'exemple le plus connu, sont parmi les algorithmes les plus efficaces pour la résolution du problème de plus court chemin [Ber93]. Les algorithmes d'étiquetage sont basés sur l'utilisation d'un ensemble de labels (étiquettes) qui définit un arbre de chemins d'un nœud source aux autres nœuds du graphe. Cet arbre correspond à un sous-graphe du graphe initial. Un label est associé à un nœud (ou une feuille) de cet arbre, il correspond à un chemin entre le nœud source du graphe et son propre nœud. Les algorithmes d'étiquetage sont des algorithmes itératifs pour lesquels à chaque itération l'arbre des chemins soit s'agrandit d'une branche (i.e. d'un nouveau nœud et d'un nouveau label) soit corrige une branche en améliorant le chemin permettant d'atteindre un nœud. L'arbre obtenu à la terminaison de l'algorithme est un arbre de plus courts chemins du nœud source à tous les nœuds. Deux familles d'algorithmes d'étiquetage existent : les algorithmes de *label correcting* et les algorithmes de *label setting*. L'arbre des chemins peut être corrigé dans les algorithmes de label correcting mais pas dans les algorithmes de label setting.

La prise en compte de plusieurs objectifs

De nombreux problèmes comportent plusieurs objectifs non-redondants à optimiser. Par exemple, on peut considérer l'optimisation simultanée d'un objectif de coût et d'un objectif de qualité. La prise en compte de plusieurs objectifs non-redondants implique en général la non-existence d'une unique solution optimale sur chaque objectif. En revanche, il existe un ensemble de solutions dites efficaces pour lesquelles aucune autre solution n'est meilleure sur chacun des objectifs. Deux problématiques surviennent alors : comment choisir une solution parmi l'ensemble des solutions efficaces et comment construire un ensemble de solutions efficaces comprenant la solution efficace choisie. Pour répondre à la première problématique il est parfois possible de faire intervenir un décideur, qui est un expert du domaine d'application capable d'exprimer ses préférences entre les solutions efficaces. Les deux problématiques impliquent deux étapes dans le processus de prise en compte des objectifs. On distingue

trois méthodologies pour organiser ces deux étapes. La méthodologie *a posteriori* propose de calculer un ensemble complet de solutions efficaces puis d'utiliser une méthode d'aide à la décision pour choisir une solution préférée. La méthodologie *a priori* propose de définir un modèle mathématique définissant les préférences du décideur, puis d'utiliser ce modèle durant la résolution du problème pour construire directement une solution préférée. La méthodologie *interactive* propose de résoudre le problème combinatoire et de choisir une solution de manière itérative via plusieurs interactions avec le décideur. Dans le cadre du routage réseau, il n'est pas possible de demander l'intervention d'un décideur humain pour chaque résolution d'un problème de plus court chemin, dans ce cas il est nécessaire de faire appel à la *décision automatique*. La décision automatique implique une méthodologie comme l'approche a priori. Dans un premier temps un modèle des préférences du décideur est construit. En *aide à la décision multi-critère* (MCDM) il est admit que le modèle doit être créé par un expert MCDM qui pose les bonnes questions au décideur. Ensuite ce modèle est réutilisé pour construire directement une solution préférée selon ce modèle dans un grand nombre de problèmes de plus court chemin. Le premier chapitre de cette thèse aborde les différentes problématiques concernant la prise en compte de plusieurs objectifs dans les problèmes d'optimisation combinatoire.

Modélisation des préférences

La modélisation des préférences consiste à construire un modèle mathématique représentatif des préférences du décideur : un modèle de préférence. Une littérature très abondante [FGE05] présente la modélisation des préférences pour un problème de décision prenant en compte plusieurs critères *. De nombreux modèles de préférence pour représenter les préférences du décideur existent. On considère dans nos travaux que le modèle de préférence est une *fonction d'utilité* agrégeant les valeurs des chemins (ou solutions) sur les objectifs en une unique valeur appelée utilité. L'utilité d'une solution, définie sur une échelle de satisfaction, donne une évaluation globale de la satisfaction du décideur pour chaque solution, ce qui permet ensuite de comparer les solutions et de déterminer la meilleure d'entre elles. La modélisation des préférences avec une fonction d'utilité est fréquemment utilisée dans la littérature [WDF+08] dans le cadre de l'aide à la décision multi-critère. Ce type de modèle est aussi apprécié en optimisation combinatoire multi-objectif.

Si la fonction d'utilité est "simple" (voire simpliste) alors l'optimisation de cette fonction dans un problème multi-objectif est équivalente à un problème mono-objectif. Le cas le plus connu de fonction d'utilité simple est une somme pondérée. Pour optimiser une telle fonction d'utilité, un algorithme de plus court chemin mono-objectif peut être utilisé. Dans le cas contraire, si la fonction d'utilité est "complexe" alors certaines propriétés, utiles à l'optimisation de cette fonction, ne peuvent être exploitées pour réduire ce problème à un problème mono-objectif.

La modélisation des préférences à l'aide d'une somme pondérée, souffre néanmoins de nombreux défauts et limitations : l'impossibilité de modéliser la commensurabilité entre les critères qui permet de comparer la satisfaction d'une solution sur les différents critères ; l'impossibilité de modéliser une complémentarité ou une substituabilité entre les critères.

Pour résoudre les limitations de la somme pondérée, différents modèles de préférence on été proposés dans la littérature. Dans la théorie MAUT [Dye05], une fonction d'utilité est composée d'une fonction d'utilité partielle pour chaque objectif et d'une fonction d'agrégation. Les fonctions d'utilité partielles permettent d'établir la commensurabilité entre les objectifs, les valeurs obtenues grâce à ces fonctions étant comparables car définies sur une même échelle de satisfaction. La fonction d'agrégation agrège les valeurs obtenues par les fonctions d'utilité partielles en une unique valeur définie sur cette échelle

∗. La différence exacte entre le terme critère et le terme objectif est définie dans le chapitre 1, voir la section 1.2.3.

de satisfaction, elle permet d'établir une pondération et/ou une complémentarité et/ou une substituabilité entre chaque ensemble de critères. Nous avons choisi d'utiliser une fonction d'utilité MAUT composée de fonctions d'utilité partielles linéaires par morceaux monotones croissantes et d'une *intégrale de Choquet* pour agréger les objectifs. Les fonctions d'utilité partielles linéaires par morceaux sont largement utilisées dans la littérature, elles sont notamment utilisées dans les méthodes MACBETH [eCCV05] et UTA [SGM05]. L'intégrale de Choquet est une fonction d'agrégation utilisée récemment pour la modélisation des préférences multi-critère [Gra95]. Contrairement aux fonctions d'agrégation habituellement utilisées (la somme pondérée par exemple) l'intégrale de Choquet permet de modéliser à la fois une pondération entre les critères et une substituabilité ou une complémentarité pour des ensembles de critères. Ainsi l'intégrale de Choquet peut être utilisée pour à la fois modéliser les préférences du décideur dans le cadre de l'aide à la décision multi-critère, mais aussi pour modéliser la préférence pour une solution équilibrée en optimisation robuste [GP07]. L'intégrale de Choquet permet aussi de modéliser de nombreuses fonctions d'agrégation [GP99] comme une somme pondérée, une fonction d'agrégation $\max$ ou $\min$, une norme de Tchebychev, une fonction OWA [Yag88],...

Le problème de plus court chemin multi-objectif

En 1980, plusieurs méthodes pour la résolution de différents problèmes de plus court chemin bi-objectif ont été proposées par Hansen [Han80]. Depuis, de nombreuses méthodes pour la résolution du problème de plus court chemin bi-objectif ou multi-objectif ont été publiées dans la littérature. Le deuxième chapitre présente chacune de ces méthodes et propose une nouvelle classification. Les différentes comparaisons des temps de calcul disponibles dans la littérature montrent que les méthodes exactes les plus efficaces sont des algorithmes d'étiquetage. La plupart des algorithmes d'étiquetage pour le problème de plus court chemin multi-objectif utilisent la dominance de Pareto pour comparer deux labels correspondant à deux chemins différents mais associés avec le même nœud du graphe. Un premier label domine un deuxième label selon Pareto si le premier label est meilleur ou égal que le deuxième label sur chacun des objectifs considérés. Les labels dominés selon Pareto sont supprimés dans l'algorithme d'étiquetage, ce qui permet en général de réduire l'explosion combinatoire (i.e. le nombre de labels générés).

Comme dans le cas mono-objectif un algorithme d'étiquetage multi-objectif est dit de *label correcting*, soit dit de *label setting*. Pour certains problèmes de plus court chemin, les algorithmes de label correcting sont plus rapides que les algorithmes de label setting, pour d'autres problèmes de plus court chemin c'est l'inverse [GM01, PRS07]. Pour cette raison nous avons choisi de travailler sur des méthodes générales pouvant être appliquées dans n'importe quel algorithme d'étiquetage.

Dans la classification des méthodes existantes nous proposons de différencier les méthodes pour résoudre des problèmes de plus courts chemins d'un nœud source à tous les nœuds et les méthodes pour des problèmes de plus court chemin d'un nœud source à un (ou quelques) nœud puits. Ces deux problèmes possèdent des applications différentes, ainsi dans le cadre du routage dans un réseau internet si on souhaite établir pour un routeur une table de routage vers l'ensemble des autres routeurs du réseau il est nécessaire de résoudre un problème de plus court chemin de ce routeur vers tous les autres routeurs/nœuds. Par contre si on souhaite établir une route pour un flux de paquets précis entre deux routeurs (routage *on-demand*) alors il est uniquement nécessaire de résoudre le problème de plus court chemin du nœud source au nœud puits. Notons que dans le cadre de problèmes de plus court chemin d'un nœud à un nœud, il est possible de calculer une estimation de la distance restant à parcourir pour atteindre le nœud puits pour chaque label, contrairement aux problèmes d'un nœud à tous les nœuds pour lesquels la

notion de nœud puits n'existe pas. L'exploitation de cette information permet de supprimer un nombre important de labels dans un algorithme d'étiquetage.

Problématique

La problématique abordée dans cette thèse est l'optimisation d'une fonction d'utilité basée sur l'intégrale de Choquet et des fonctions d'utilité linéaires par morceaux dans le cadre de problèmes de plus courts chemins multi-objectifs. Cette fonction d'utilité est complexe, elle ne permet pas en général de réduire un problème multi-objectif en un problème mono-objectif. Pour ce problème en particulier aucune méthode n'était proposée dans la littérature, il existait néanmoins plusieurs méthodes sur des problématiques proches.

Concernant l'optimisation d'une fonction d'utilité complexe, pour le problème de plus court chemin multi-objectif, différentes méthodes ont néanmoins été proposées dans la littérature : pour optimiser une intégrale de Choquet concave [GP07], une norme de Tchebychev [SN10] ou une fonction d'agrégation OWA [GS07]. Ces méthodes sont toutes basées sur des algorithmes d'étiquetage pour le problème de plus court chemin multi-objectif. Afin de prendre en compte ces fonctions d'utilité, ces différents travaux proposent de supprimer des labels dans l'algorithme d'étiquetage en calculant une borne inférieure pour chaque label. Pour calculer ces bornes inférieures, des chemins optimaux selon chaque objectif et/ou des chemins optimaux selon une somme pondérée bornant inférieurement la fonction d'utilité [GPS10] sont calculés préalablement à l'exécution de l'algorithme d'étiquetage.

On note que l'ensemble de la littérature concernant l'optimisation d'une fonction d'utilité pour le problème de plus court chemin multi-objectif s'intéresse uniquement au problème de plus court chemin d'un nœud à un nœud.

Contributions concernant le calcul d'une borne inférieure

Durant cette thèse nous avons dans un premier temps travaillé à l'amélioration et à la généralisation des travaux précédents. Ainsi la première méthode que nous proposons pour optimiser une intégrale de Choquet quelconque est une généralisation des travaux de Galand et Perny (2007) [GP07] concernant l'optimisation d'une intégrale de Choquet concave. Notre méthode repose sur le calcul d'une somme pondérée bornant inférieurement l'intégrale de Choquet. Ensuite entre chaque nœud du graphe et le nœud puits, un chemin optimal selon cette somme pondérée est construit. Ces chemins permettent de définir une borne inférieure pour chaque label. Cette borne inférieure permet, à l'aide d'une règle de coupe, de supprimer des labels durant l'exécution de l'algorithme d'étiquetage. Les temps de calcul montrent que cette méthode permet de réduire de manière significative les temps de calcul par rapport à une méthode calculant l'ensemble des solutions efficaces. Cette méthode souffre néanmoins de temps de calcul importants quand l'intégrale de Choquet n'est pas concave.

Lorsque l'intégrale de Choquet n'est pas concave, la somme pondérée calculée ne permet pas de calculer une borne inférieure satisfaisante pour les labels. La règle de coupe utilisant cette borne inférieure ne permet de supprimer que peu de labels durant l'exécution de l'algorithme d'étiquetage. Nous avons alors proposé d'utiliser une nouvelle méthode utilisant des bornes inférieures et supérieures sur les objectifs pour calculer une meilleure somme pondérée bornant l'intégrale de Choquet. Pour chaque objectif nous calculons une borne inférieure et une borne supérieure qui encadre l'ensemble des solutions efficaces. L'utilisation de ces bornes dans un programme linéaire nous permet de construire une somme pondérée qui borne inférieurement l'intégrale de Choquet uniquement sur l'espace des objectifs situé entre ces bornes. Les expérimentations indiquent que cette méthode permet d'améliorer les temps

de calcul de façon significative par rapport à la méthode précédente quand l'intégrale de Choquet n'est pas concave.

Nous proposons ensuite deux méthodes pour calculer une somme pondérée bornant inférieurement une fonction d'utilité comprenant une intégrale de Choquet et des fonctions d'utilité partielles linéaires par morceaux. Sur chaque objectif nous proposons de calculer une fonction linéaire bornant la fonction d'utilité partielle associée à cet objectif. Ces fonctions linéaires peuvent ensuite être aisément intégrées à la somme pondérée bornant l'intégrale de Choquet, définissant ainsi une nouvelle somme pondérée bornant inférieurement la fonction d'utilité. Dans un deuxième temps, nous proposons une méthode pour calculer pour chaque objectif une fonction affine bornant la fonction d'utilité partielle associée à cet objectif. Les fonctions affines peuvent être utilisées de la même manière que les fonctions linéaires. Ces fonctions peuvent être intégrées à la somme pondérée bornant l'intégrale de Choquet, définissant ainsi une nouvelle somme pondérée augmentée d'une fixe bornant la fonction d'utilité. Cette somme pondérée augmentée peut ensuite être utilisée de la même manière qu'une somme pondérée habituelle.

Dominance selon une fonction d'utilité

Aucune proposition n'existait dans la littérature pour optimiser une fonction utilité dans le cadre du problème de plus court chemin d'un nœud à tous les nœuds ou pour optimiser une fonction d'utilité avec des objectifs non additifs. Nous avons alors proposé des nouvelles méthodes pour ces problèmes.

Dans le cadre du problème de plus court chemin d'un nœud source à tous les nœuds, il n'est pas possible d'utiliser les travaux précédents sur le calcul d'une borne inférieure. En effet, un label peut être propagé vers un nœud proche ou bien vers un nœud très éloigné, ainsi aucune borne inférieure intéressante ne peut être calculée en général pour un label. La dominance de Pareto était le seul outil à notre disposition pour supprimer des labels. Pour deux labels associés à un même nœud si un label est meilleur que l'autre sur chaque objectif alors ce deuxième label peut être supprimé, autrement dit uniquement les labels efficaces sont conservés.

Nous avons alors cherché à définir une nouvelle forme de dominance capable de prendre en compte une fonction d'utilité. Considérons deux labels, si le premier label est bien meilleur que le deuxième label sur chaque objectif à l'exception d'un seul objectif alors aucun label ne domine selon Pareto l'autre label. Cependant le deuxième label du point de vue de nombreuses fonctions d'utilité semble beaucoup moins intéressant que le premier label. On peut alors se demander si il n'est pas possible, considérant uniquement la fonction d'utilité et le premier label, de supprimer ce deuxième label.

Dans un premier temps, nous proposons la définition de la dominance selon une intégrale de Choquet. Comme il semble difficile de vérifier exactement cette dominance entre deux labels, on propose une condition suffisante. Ainsi pour deux labels si cette condition suffisante est vérifiée alors la dominance selon l'intégrale de Choquet est aussi vérifiée entre ces deux labels. Cette condition suffisante peut alors être utilisée dans un algorithme d'étiquetage de la même manière que la dominance de Pareto et comme elle ne prend pas en compte de nœud puits elle peut être utilisée pour résoudre le problème de plus court chemin d'un nœud à tous les nœuds. De plus cette condition suffisante permet de supprimer plus de labels que la dominance de Pareto. La vérification de cette condition suffisante peut être réalisée rapidement, néanmoins elle nécessite un temps de calcul plus important que la vérification de la dominance de Pareto. Quand l'intégrale de Choquet modélise une forte complémentarité ou une forte substituabilité entre les objectifs peu de labels en plus de ce ceux habituellement supprimés avec la dominance de Pareto peuvent être supprimés. Ainsi pour ces cas de l'intégrale de Choquet on observe un temps de calcul plus important pour un algorithme d'étiquetage utilisant cette condition suffisante que le même algorithme utilisant la dominance de Pareto. Nous avons alors proposé, pour deux labels, d'utiliser cette condition suffisante à la

place de la dominance de Pareto uniquement si une fonction de test sur la différence entre les deux labels est vérifiée. Cette méthode permet d'améliorer de façon significative les temps de calcul de l'algorithme d'étiquetage.

Nous avons ensuite travaillé sur la définition d'une dominance selon une fonction d'utilité composée d'une intégrale de Choquet et de fonctions d'utilité linéaires par morceaux dans un problème de plus court chemin multi-objectif composé non seulement d'objectifs additifs mais aussi d'objectifs bottleneck ou multiplicatifs. On retrouve ces objectifs non additifs dans de nombreux problèmes de routage dans les réseaux internet, par exemple un objectif de maximisation de la bande passante se modélise avec un objectif bottleneck. Comme précédemment nous proposons une condition suffisante pour deux labels qui si elle est vérifiée permet d'affirmer qu'un label est dominé par un autre label selon cette fonction d'utilité et pour ce problème de plus court chemin.

Contribution au routage multi-objectif

Pour établir le routage des paquets dans un réseau internet, chaque routeur construit généralement une table de routage en déterminant l'ensemble des chemins vers les autres routeurs du réseau. En général des algorithmes d'étiquetage sont utilisés pour construire ces chemins [XN99, YF03]. Souvent seul un objectif de coût est pris en compte, le routage est alors dit "*best-effort*". Récemment de nombreuses méthodes pour le routage multi-objectif prenant en compte des objectifs de qualité de service ont été publiées [YF03, GBR06]. Le routage par contraintes est une des approches présentées pour prendre en compte des objectifs de qualité de service [XN99]. Cette approche implique que les chemins empruntés par les paquets doivent respecter des contraintes sur certains objectifs. Pour cela, il est nécessaire de résoudre un problème de plus court chemin multi-objectif sous contraintes. Dans la littérature, pour résoudre ce type de problème, on trouve de nombreux algorithmes d'étiquetage optimisant une fonction d'agrégation : une somme pondérée [Jaf84], une fonction min-max pondérée [NM99, MNK01] ou une norme de Tchebytchev [RGFT04].

Dans le cadre du routage par contraintes, les contraintes sur les objectifs peuvent être vues comme une modélisation (simpliste) des préférences de l'administrateur du réseau (i.e. le décideur) : un chemin vérifiant l'ensemble des contraintes est pleinement satisfaisant, dans le cas contraire le chemin est insatisfaisant. Si un tel modèle de préférence est facile à définir pour l'administrateur, il ne permet pas en général de modéliser fidèlement ses préférences. Une telle modélisation des préférences implique qu'un chemin satisfaisant toutes les contraintes sera toujours préféré à un chemin globalement plus performant mais ne satisfaisant pas une contrainte.

Dans le cadre général du routage multi-objectif, plutôt que de définir des contraintes sur les objectifs, nous proposons de définir une fonction d'utilité composée d'une intégrale de Choquet et de fonctions d'utilité partielles linéaires par morceaux pour modéliser les préférences de l'administrateur. Une telle fonction d'utilité permet non seulement de modéliser des contraintes sur des objectifs à l'aide des fonctions d'utilité partielles, mais aussi de modéliser un grand nombre de fonctions d'agrégation : une somme pondérée, une fonction d'agrégation $\max$ pondérée ou une norme de Tchebychev. Cette fonction d'utilité permet aussi de modéliser une complémentarité ou une substituabilité sur chaque ensemble d'objectifs [GL10]. On peut alors considérer que cette approche est un généralisation de différentes approches existantes dans la littérature comme la satisfaction de contraintes sur les objectifs, ou l'agrégation des objectifs à l'aide d'une somme pondérée.

Afin optimiser cette fonction d'utilité, nous proposons d'utiliser la dominance selon cette fonction d'utilité dans un algorithme d'étiquetage. Dans le cadre de routage multi-objectif, il est parfois nécessaire de modéliser la bande passante maximale des chemins avec un objectif bottleneck. Nous montrons que la

méthode proposée est efficace pour prendre en compte un tel objectif. Ces travaux sur le routage ont été réalisés dans la continuation de travaux antérieurs à cette thèse sur l'algorithme RMC [GBR06, Beu06] produit d'une coopération entre l'université de Valenciennes et ALCATEL.

Plan de la thèse

La thèse est composée de 5 chapitres. Le premier chapitre présente la problématique et le cadre général de cette thèse, notamment le domaine de l'optimisation combinatoire multi-objectif et celui de l'aide à la décision multi-critère. Le deuxième chapitre propose un cadre général, une classification et une présentation de l'ensemble des méthodes non interactives proposées dans la littérature pour la résolution du problème de plus court chemin multi-objectif. Nous présentons les algorithmes permettant le calcul de l'ensemble des solutions efficaces et les algorithmes optimisant une fonction d'utilité (ou fonction d'agrégation). Le troisième chapitre présente nos contributions pour le problème de plus court chemin d'un nœud à un autre nœud. Les méthodes proposées permettent de calculer des bornes inférieures qui sont ensuite utilisées dans une règle de coupe pour supprimer des labels dans les algorithmes d'étiquetage. Le quatrième chapitre présente nos contributions pour le problème de plus court chemin d'un nœud à tous les nœuds. Ce chapitre définit des conditions suffisantes pour déterminer la dominance entre deux labels selon une fonction d'utilité. Le cinquième chapitre présente le routage prenant en compte la qualité de service et notre contribution au routage multi-objectif. Plusieurs perspectives dans la continuation de ces travaux sont également présentées.

Notations

Notation	Signification
$\mathbb{R}$	ensemble des réels
$\mathbb{R}_+ \subset \mathbb{R}$	ensemble des réels positifs ou nul
X	ensemble des solutions
p	nombre d'objectifs
n	nombre de variables
$P = \{1, \ldots, p\}$	ensemble des objectifs
$x \in \{0,1\}^n$	une solution
$X \subset \{0,1\}^n$	ensemble des solutions d'un problème combinatoire en variable binaire
$\Omega_k \subset \mathbb{R}$	espace de définition de l'objectif $k \in P$
$\mathbf{\Omega} = \Omega_1 \times \cdots \times \Omega_p$	espace des objectifs
$z_k : X \to \Omega_k$	fonction objectif de l'objectif $k \in P$
$y \in \Omega$	le vecteur de coûts
$Y = z(X) \subset \Omega$	espace des objectifs
$\xi \subset \mathbb{R}$	échelle commune de satisfaction
$U : \Omega \to \xi$	fonction d'utilité
$V : \xi^p \to \xi$	fonction d'agrégation
$u_k : \Omega_k \to \xi$	fonction d'utilité partielle de l'objectif $k \in P$
Υ_α	norme de Tchebychev
$\|.\|_2$	norme euclidienne
$\lambda_k \in \mathbb{R}_+$, $\alpha_k \in \mathbb{R}_+$	poids de l'objectif $k \in P$ associé à une somme pondérée
$\varphi_\lambda : \mathbb{R}^p \to \mathbb{R}$	somme pondérée
$\prec_{DM}, \sim_{DM}, \precsim_{DM}$	relations de préférence du décideur
$\prec_P$	relation de dominance de Pareto
$\prec_{GBR}$	relation de dominance GBR (pour des objectifs bottleneck)
$\prec_U, \precsim_U$	relations de dominance selon une fonction d'utilité U
$\mathfrak{P}$	ensemble des parties de P
$\mu : \mathfrak{P} \to [0,1]$	fonction de capacité
$m : \mathfrak{P} \to \mathbb{R}$	fonction de Möbius
$\phi_k \in [0,1]$	indice de Shapley du critère $k \in P$
$\epsilon \in \mathbb{R}_+$	une petite valeur positive
$\leq_k, <_k$	relation d'inégalité selon le sens d'optimisation de l'objectif $k \in P$

Notation	Signification
$E_M(X) \subset X$	ensemble maximal complet des solutions efficaces
$E(X) \subset E_M(X)$	un ensemble complet de solutions efficaces
$E_m(X) \subset E_M(X)$	un ensemble minimal complet de solutions efficaces
$E_S(X) \subset E_M(X)$	ensemble des solutions supportées
$y^I \in \mathbf{\Omega}$	point idéal
$y^N \in \mathbf{\Omega}$	point nadir
n	nombre de nœud
$N = \{1, \ldots, n\}$	ensemble des nœuds
$A \subset N \times N$	ensemble des arcs
$c : A \to \mathbb{R}^p$	fonction de coût
$G = (N, A, c)$	un graphe
$s \in N$	le nœud source
$t \in N$	le nœud puits
$\Gamma^+(i) \subset N$	ensemble des nœuds successeurs du nœud $i \in N$
$\Gamma^-(i) \subset N$	ensemble des nœuds prédécesseurs du nœud $i \in N$
$v_i \in N,\ i \in \{1, \ldots, n\}$	un nœud
$r = \langle v_1, \ldots, v_l \rangle$	un chemin de v_1 à v_l, séquence de l nœuds
$R_{\bullet,\bullet}$	ensemble des chemins
$R_{i,\bullet} \subset R_{\bullet,\bullet}$	ensemble des chemins de $i \in N$ à un nœud quelconque
$R_{\bullet,j} \subset R_{\bullet,\bullet}$	ensemble des chemins d'un nœud quelconque à $j \in N$
$R_{i,j} \subset R_{i,\bullet}$	ensemble des chemins de i à j
$\otimes_k : \mathbb{R}^2 \to \mathbb{R}$	opérateur binaire de l'objectif $k \in P$
$\mathbf{0}_k \in \mathbb{R}$	élément neutre de l'opérateur binaire $\otimes_k$
l-$\sum$	l fonctions objectif additives à minimiser
l-max	l fonctions objectif maximales à minimiser
l-min	l fonctions objectif minimales à maximiser
l-$\prod$	l fonctions objectif multiplicatives à minimiser
Q	un nombre indéterminé de fonctions objectif
ℓ_i	un label associé au nœud $i \in N$
L_i	ensemble des labels associés au nœud $i \in N$
$\mathfrak{L}$	ensemble des labels ouverts
$q_k : N \to \mathbb{R}$	fonction d'évaluation de l'objectif $k \in P$

CHAPITRE 1

Préférences du décideur et optimisation combinatoire multi-objectif

Il existe de nombreux problèmes dans le monde réel pouvant être modélisés par des problèmes combinatoires, par exemple la recherche d'un chemin dans un réseau routier ou de télécommunication. Pour ces problèmes, on va en général chercher à optimiser un objectif comme le coût de ce chemin. Parfois il est nécessaire de prendre en compte plusieurs objectifs. Cependant il n'existe pas en général une solution qui soit optimale sur chacun des objectifs. La notion de solution optimale unique à un problème d'optimisation combinatoire multi-objectif (cf. §1.1, de la présente page) quand elle existe n'a plus de sens. Il est possible de calculer un ensemble de solutions de compromis pour lesquelles il n'existe aucune autre solution qui soit meilleure sur chacun des objectifs. Ces solutions sont dites efficaces. L'ensemble des solutions efficaces pouvant être important, il se pose alors la question de cerner la meilleure solution efficace d'un problème multi-objectif. L'aide à la décision multi-critère (cf. §1.2, page 22) propose des réponses à cette question. Il faut pour y répondre collecter des informations sur les préférences d'un expert du domaine capable d'exprimer ses préférences sur les solutions du problème. Cet expert est appelé le décideur. Les informations préférentielles exprimées par le décideur permettent alors de construire un modèle de préférence, qui permet par exemple de choisir la meilleure solution. L'intervention du décideur et la résolution du problème d'optimisation sont les deux étapes principales d'un tel processus. Plusieurs méthodologies existent pour organiser ces deux étapes (cf. §1.3, page 36).

1.1 Optimisation combinatoire multi-objectif

Un *problème d'optimisation* est défini par un ensemble de variables, un ensemble de contraintes et une fonction objectif. Une solution d'un tel problème est une affectation de l'ensemble des variables vérifiant l'ensemble des contraintes. Résoudre un problème d'optimisation consiste à construire une solution qui soit optimale au regards de la fonction objectif. Un *problème d'optimisation discret* est un problème d'optimisation dans lequel les variables sont entières ou réelles. La fonction objectif d'un tel problème peut par exemple être une somme, une fonction maximale, une fonction minimale ou une multiplication. Un *problème d'optimisation combinatoire* est un problème d'optimisation discret en contraintes linéaires qui possède une structure particulière. On sait aussi que le nombre de solutions d'un problème combi-

natoire est fini [Ehr00b] mais souvent trop élevé pour pouvoir les énumérer. Il n'existe cependant pas de définition précise de ce qu'est un problème d'optimisation combinatoire. Parmi ces problèmes, on retrouve le problème de sac à dos, le problème de tri, le problème de plus court chemin, le problème d'affectation (quadratique). Le problème de plus court chemin est détaillé dans le chapitre 2.

Un problème d'optimisation combinatoire mono-objectif avec n variables et m contraintes se modélise donc de la manière suivante :

$$\begin{array}{ll} \text{opt.} & z(x) \\ \text{s.c.} & Ax \leq b \\ & x_i \in \mathbb{X},\ i = 1, \ldots, n \end{array} \tag{1.1}$$

Les variables sont $x_1, \ldots, x_n$, la fonction objectif est $z : \mathbb{X}^n \to \mathbb{R}$ et les contraintes sont définies par $Ax \leq b$. Avec $A \in \mathbb{R}^{(m \times n)}$ une matrice de contraintes, $b \in \mathbb{R}^m$ les limites des contraintes et $\mathbb{X}^n \subset \mathbb{N}_+^n$ l'espace de définition des variables. "opt." indique que la fonction objectif z doit être minimisée ou maximisée. Une *solution réalisable* x est un vecteur de $\mathbb{X}^n$ qui satisfait les m contraintes linéaires du problème. *L'ensemble des solutions réalisables* du problème est alors défini par X. Par abus de langage on considère qu'une *solution* est une *solution réalisable*.

Pour chaque variable $i \in \{1, \ldots, n\}$, une valeur réelle $c(i) \in \mathbb{R}$ représentant *le coût* * de cette variable est définie. $c : \{1, \ldots, n\} \to \mathbb{R}$ est la fonction de coût. Pour une solution $x \in \mathbb{X}^n$, la fonction objectif $z : \mathbb{X}^n \to \mathbb{R}$ agrège les coûts des variables $x_1, \ldots, x_n$. Dans la littérature on peut trouver différentes fonctions objectif :

- $z(x) = \sum_{i=1}^{n} c(i)x_i$: l'objectif est dit *additif*.
- $z(x) = \min_{i=1}^{n} c(i)x_i$: si l'objectif est à maximiser alors il est appelé $\max\min$ ou *bottleneck*. De manière équivalente un objectif $\min\max$ est aussi dit *bottleneck*.
- $z(x) = \Pi_{i=1}^{n} c(i)x_i$: l'objectif est dit multiplicatif (ou *quadratique*).

Un objectif additif correspond par exemple à un coût (monétaire, écologique ou autre), un objectif bottleneck à une capacité et un objectif multiplicatif à un risque.

1.1.1 Optimiser plusieurs objectifs

Un problème d'optimisation combinatoire multi-objectif (repris sous l'acronyme MOCO pour "*Multi-Objective Combinatorial Optimization*") est un problème combinatoire dans lequel plusieurs fonctions objectif doivent être optimisées simultanément. Soient $z_1, \ldots, z_p : \mathbb{X}^n \to \mathbb{R}$ l'ensemble des fonctions objectif avec p le nombre d'objectifs et $P = \{1, \ldots, p\}$ l'ensemble des objectifs. Un problème d'optimisation combinatoire multi-objectif avec n variables et m contraintes se modélise de la manière suivante :

$$\begin{array}{ll} \text{vopt} & z(x) = (z_1(x), \ldots, z_p(x)) \\ \text{s.c.} & Ax \leq b \\ & x \in \mathbb{X}^n \end{array} \tag{1.2}$$

"vopt" indique que chaque fonction objectif doit être minimisée ou maximisée. $z = (z_1, \ldots, z_p) : \mathbb{X}^n \to \mathbb{R}^p$ est le vecteur des fonctions objectif. On considère que sauf indication contraire chaque fonction objectif doit être minimisée.

Soient $c_1, \ldots, c_p : \{1, \ldots, n\} \to \mathbb{R}$ les fonctions de coût sur chaque objectif et $c : \{1, \ldots, n\} \to \mathbb{R}^p$ la fonction vectorielle de coût telle que $\forall i \in N$, $c(i) = (c_1(i), \ldots, c_p(i))$ et $z(i) = (z_1(i), \ldots, z_p(i))$, $\forall i \in \{1, \ldots, n\}$. On définit $y = z(x)$ le *point* de la solution x dans *l'espace des objectifs* $Y = z(X)$.

∗. Ces travaux se situent dans le contexte de la minimisation, ainsi le terme de coût est utilisé pour désigner aussi bien la capacité, la performance, le risque ou même le coût associé à une variable ou à une solution.

1.1.1.1 Solutions efficaces

Il n'existe pas en général une solution qui optimise tous les objectifs simultanément. En l'absence d'une solution optimale sur l'ensemble des objectifs, la *dominance au sens de Pareto* ($\prec_P$) [Par06] peut être utilisée pour comparer les solutions entres elles. Par souci de simplicité nous définissons l'inégalité $\leq_k, \forall k \in P$ telle que $\leq_k \Leftrightarrow \leq$ ($<_k \Leftrightarrow <$) si l'objectif k doit être minimisé et $\leq_k \Leftrightarrow \geq$ ($<_k \Leftrightarrow >$) si l'objectif k doit être maximisé.

Définition 1.1. Dominance au sens de Pareto [Ehr00b].
Soient $x^a, x^b \in X$ deux solutions et $y^a, y^b \in Y$ t.q. *$y^a = z(x^a)$ et $y^b = z(x^b)$ leur points respectifs dans l'espace des objectifs $Y = z(X)$, alors :*

- $y^a \prec_P$-domine y^b *($y^a \prec_P y^b$) si et seulement si $y^a_k \leq_k y^b_k, \forall k \in P$ et $y^a_k \neq y^b_k$.*
- $x^a \prec_P$-domine x^b *si et seulement si $z(x^a) \prec_P$-domine $z(x^b)$.*
- y^a *est* non $\prec_P$-dominé *si et seulement si $\forall y \in Y$, y ne $\prec_P$-domine pas y^a.*
- x^a *est dit* $\prec_P$-efficace *si est seulement si $z(x^a)$ est non $\prec_P$-dominé.*
- y^a faiblement $\prec_P$-domine y^b *si et seulement si $y^a \prec_P$-domine y^b et* $\exists k \in P$, t.q. $y^a_k = y^b_k$.
- x^a *est dit* faiblement $\prec_P$-efficace *si et seulement si il n'existe pas de solution* $x \in X$ t.q. $z_k(x) < z_k(x^a), \forall k \in P$.

Notons aussi que deux solutions x^a et x^b égales sur tous les objectifs sont dites *équivalentes* : $z_k(x^a) = z_k(x^b), \forall k \in P$. D'autres définitions de la notion de dominance existent dans la littérature, citons la *dominance au sens de Lorenz* [GP10].

Définition 1.2. Ensembles de solutions efficaces [Han80].

- $E(X) \subset X$ *désigne* un ensemble complet de solutions efficaces*, i.e. pour chaque point non dominé dans $Y = z(X)$ il existe au moins une solution correspondante dans $E(X)$.*
- $E_M(X)$ *désigne* l'ensemble maximal complet des solutions efficaces*, i.e. toutes les solutions efficaces appartiennent à cet ensemble.*
- $E_m(X) \subset E_M(X)$ *désigne* un ensemble minimal complet des solutions*, i.e. il existe exactement une solution efficace dans $E_m(X)$ pour chaque point non dominé dans Y.*

La taille de l'ensemble maximal complet des solutions efficaces $E_M(X)$ peut être exponentielle par rapport au nombre n de variables [Ser86]. Par exemple considérons un problème MOCO en variables binaires ($\mathbb{X} = \{0, 1\}$), sans contraintes avec deux objectifs à minimiser, les deux objectifs étant contradictoires de tels sorte que $c(i) = (-1, 1)$ pour chaque variable $i \in \{1, \ldots, n\}$. Chaque vecteur de variables dans $\{0, 1\}^n$ représente une solution efficace. Il en est de même pour un ensemble minimal complet des solutions efficaces $E_m(X)$ pour certains problèmes MOCO, voir [SA00] dans le cadre du problème de plus court chemin multi-objectif.

Un ensemble complet de solutions efficaces permet de définir des bornes dans l'espace des objectifs tel que toute solution à l'extérieur de ces bornes soient nécessairement non efficace.

Définition 1.3. Point nadir et idéal.

- *Le* point idéal y^I *d'un problème* MOCO *est défini tel que, $\forall k \in P$:*

$$\forall x \in E(X),\ y^I_k \leq_k z_k(x) \text{ et } \exists x \in E(X),\ y^I_k = z_k(x)$$

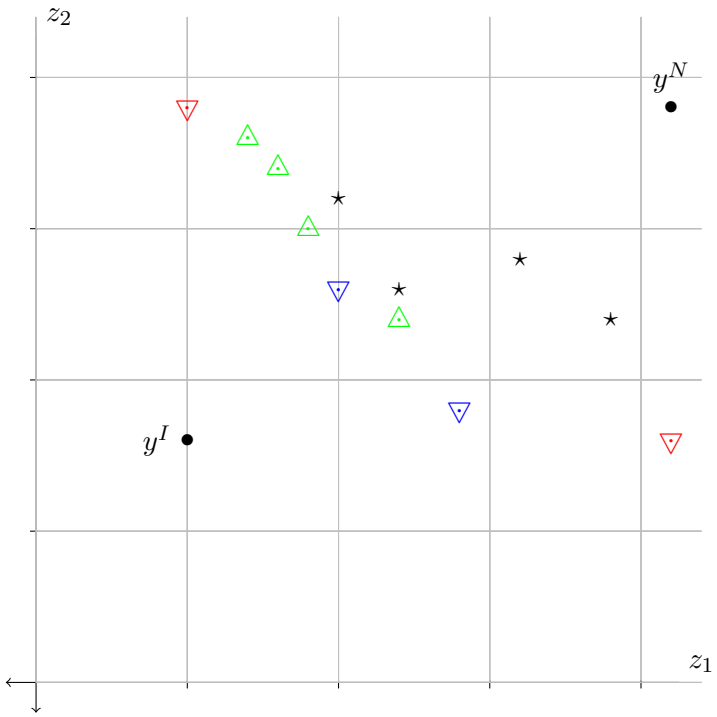

Figure 1.1 – Les étoiles et les triangles représentent les points des solutions d'un problème MOCO. Les solutions lexicographiques optimales (▽), les solutions supportées (▽ et ▽), les solutions efficaces (▽, ▽ et △), les solutions non efficace (⋆) et les points nadir y^N et idéal y^I.

- *Le* point nadir y^N *d'un problème* MOCO *est défini tel que,* $\forall k \in P$ *:*

$$\forall x \in E(X),\ y_k^N \geq_k z_k(x) \text{ et } \exists x \in E(X),\ y_k^N = z_k(x)$$

Considérons que tous les objectifs doivent être minimisés, alors le *point idéal* y^I représente le vecteur des bornes minimales des solutions efficaces ($y_k^I = \min_{y \in E(X)} y_k$) et le *point nadir* y^N représente le vecteur des bornes maximales des solutions efficaces ($y_k^N = \max_{y \in E(X)} y_k$) [Ehr00b].

Avec deux objectifs ($p = 2$), le point idéal et le point nadir peuvent être facilement calculés [Ehr00b]. Avec plus de deux objectifs ($p > 2$), il est difficile de calculer le point idéal et le point nadir [ETP03]. Les différentes définitions 1.2, 1.6, 1.4, 1.3 données ci-dessus sont illustrées par la figure 1.1.

1.1.2 Résolution des problèmes d'optimisation combinatoire multi-objectif

Il existe différentes approches pour la résolution d'un problème d'optimisation (combinatoire) multi-objectif, ces approches sont discutées dans les paragraphes suivants. Une approche qui n'est pas considérée dans cette thèse est le *Goal Programming* [Ign76] pour l'optimisation (combinatoire) multi-objectif. Ces méthodes proposent de définir dans un premier temps des points (*goals*) dans l'espace des objectifs. Dans un deuxième temps ces méthodes cherchent à minimiser la déviation par rapport à ces points [EG00], la solution la *plus proche* possible de ces points dans l'espace des objectifs est calculée. Le *Goal Programming* a été une des premières approches et une des plus étudiées pour résoudre des problèmes multi-objectifs [WDF+08].

1.1.2.1 Optimisation lexicographique

Résoudre un problème multi-objectif par optimisation lexicographique consiste, dans un premier temps, à ranger les objectifs selon leur ordre d'importance. La fonction de permutation $(.) : P \to P$ permet de ranger les objectifs de telle sorte que si $k \in P$ est l'objectif le plus important alors k est considéré comme le premier objectif ($(k) = 1$). Dans un deuxième temps, l'optimisation lexicographique consiste à optimiser les objectifs les uns après les autres dans leur ordre d'importance. Ainsi dans le cas bi-objectif ($(1) = 1$ et $(2) = 2$) il faut construire une solution optimale selon le deuxième objectif parmi les solutions qui sont optimales selon le premier objectif.

Définition 1.4. Optimisation lexicographique
Une solution $x \in X$ est lexicographique optimale *selon une permutation des objectifs $(.) : P \to P$* t.q. *:*

$$\forall x' \in X \;\; \exists l \in P \text{ t.q. } \begin{cases} z_{(k)}(x) = z_{(k)}(x') & \forall k = 1, \ldots, l-1 \\ z_{(l)}(x) <_k z_{(l)}(x') \end{cases}$$

Dans le cas bi-objectif le calcul d'une solution lexicographique optimale pour chaque permutation permet de déterminer le point idéal et le point nadir.

L'optimisation lexicographique suppose qu'il n'existe aucune compensation entre les objectifs. Ainsi l'objectif le plus important sera toujours optimisé en priorité : il n'est pas possible de compenser une légère dégradation sur le plus important des objectifs par une amélioration (même conséquente) sur les autres objectifs.

1.1.2.2 Scalarisation du problème multi-objectif à l'aide d'une somme pondérée

La *scalarisation* d'un problème multi-objectif consiste à agréger les objectifs à l'aide d'une somme pondérée.

Définition 1.5. Somme pondérée [Geo68].
Pour un point $y \in \mathbb{R}^p$ dans l'espace des objectifs, et un vecteur de poids $\lambda \in \mathbb{R}^p_+$, la somme pondérée $\varphi_\lambda : \mathbb{R}^p \to \mathbb{R}$ est définie par :

$$\varphi_\lambda(y) = \sum_{k=1}^{p} \lambda_k . y_k$$

avec $\lambda \in \mathbb{R}^p_+$ un vecteur de poids défini a priori, tel que $\sum_{k \in P} \lambda_k > 0$.

Une solution optimale d'un problème MOCO selon une somme pondérée φ_λ (ou une fonction d'utilité) est une solution (réalisable) $x \in X$ du problème MOCO qui minimise la somme pondérée : $z(x) = \min\{\varphi_\lambda(z(x')) : x' \in X\}$. Si au moins un poids est non nul ($\sum_{k \in P} \lambda_k > 0$ ou $(0, \ldots, 0) \prec_P \lambda$) alors toutes les solutions optimales selon φ_λ sont faiblement efficaces et au moins une solution optimale est efficace [Geo68]. Si tous les poids sont non nuls ($\lambda_k > 0, \forall k \in P$) alors toutes les solutions optimales selon φ_λ sont efficaces [Geo68]. La minimisation d'une somme pondérée est illustrée sur la figure 1.2.

Si chaque objectif est additif, alors la scalarisation du problème MOCO produit un problème mono-objectif équivalent [EG00]. Ce problème mono-objectif est défini par l'objectif (additif) $z'(x) = \varphi_\lambda(z(x))$, la fonction de coût $c'(i) = \varphi_\lambda(c(i))$ et les mêmes contraintes que le problème MOCO original (cf. Équation 1.2, page 12). Ainsi un problème de plus court chemin multi-objectif devient un problème de plus court chemin mono-objectif pour lequel il existe des algorithmes d'une complexité polynomiale [Ber93].

Néanmoins toutes les solutions efficaces d'un problème MOCO ne peuvent être obtenues en scalarisant ce problème, quels que soient les poids $\lambda \in \mathbb{R}^p_+$.

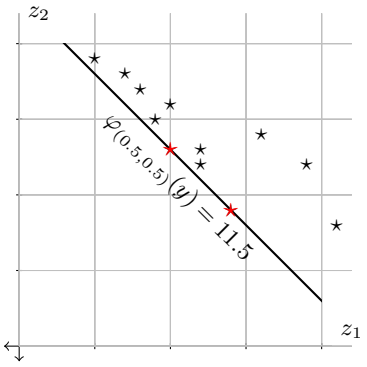

Figure 1.2 – Minimisation d'une somme pondérée $\varphi_{(0.5,0.5)}$. Le trait en gras représente la courbe de niveau de la somme pondérée. Les étoiles ($\star$) représentent les points des solutions du problème.

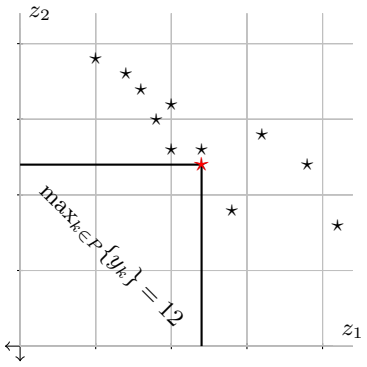

Figure 1.3 – Minimisation de la fonction d'agrégation max. Le trait en gras représente la courbe de niveau de la fonction d'agrégation max. Les étoiles ($\star$) représentent les points des solutions du problème.

Définition 1.6. Solutions supportées [Ehr00b].
Une solution $x \in X$ d'un problème MOCO *est dite* supportée *s.s.i. il existe un vecteur de poids strictement positifs $\lambda \in \mathbb{R}^p_+$, $\forall k \in P$, $\lambda_k > 0$* t.q. *x est une solution optimale selon la somme pondérée correspondante φ_λ. Une solution supportée est par définition efficace, puisque le vecteur de poids est strictement positif. En général il existe des solutions efficaces qui ne sont pas supportées, on dit que ces solutions sont* non-supportées.
Une solution $x \in X$ d'un problème MOCO *est dite* supportée extrême *s.s.i. il existe au moins deux vecteurs de poids strictement positifs $\lambda^a, \lambda^b \in \mathbb{R}^p_+$ ($\lambda^a_k > 0, \lambda^b_k > 0, \forall k \in P$) et différents $\lambda^a \neq \lambda^b$, tels que la solution soit optimale selon les deux sommes pondérées correspondantes φ_{λ^a}, φ_{λ^b}.*

En général il existe beaucoup plus de solutions non-supportées que de solutions supportées [EG00].

1.1.2.3 Recherche de solutions de compromis

L'approche dite *recherche de solutions de compromis* consiste à construire des solutions dont le vecteur de coûts représente sur chaque objectif un compromis par rapport aux coûts des autres solutions efficaces. Supposons que chaque objectif soient très important et qu'une solution mauvaise sur au moins un objectif ne puisse être considérée comme une solution optimale si il existe une solution qui n'est mauvaise sur aucun objectif, alors cette approche doit être préférée à l'approche précédente consistante à optimiser une somme pondérée. Cette approche est notamment utilisée en optimisation robuste [GP06b], ou pour le routage QoS (cf. Chapitre 4.5, page 117).

Définition 1.7. Fonctions d'agrégation min ou max pondérées.
Soit $\lambda \in \mathbb{R}^p_+$ un vecteur de poids permettant la pondération des objectifs, la fonction d'agrégation $V : \mathbb{R}^p \to \mathbb{R}$ est :

- *une fonction d'agrégation max pondérée si : $V(y) = \max_{k \in P} \lambda_k \times y_k$, $\forall y \in \mathbb{R}^p$;*
- *une fonction d'agrégation min pondérée si : $V(y) = \min_{k \in P} \lambda_k \times y_k$, $\forall y \in \mathbb{R}^p$.*

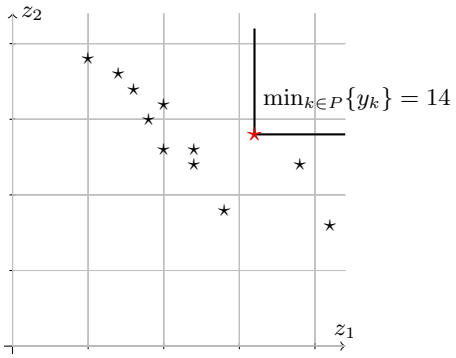

Figure 1.4 – Maximisation de la fonction d'agrégation $\min$. Le sens d'optimisation des deux objectifs a été inversé en maximisation afin d'illustrer la recherche d'une solution de compromis dans le cadre de la maximisation. Le trait en gras représente la courbe de niveau de la fonction d'agrégation $\min$. Les étoiles ($\star$) représentent les points des solutions du problème.

La minimisation d'une fonction d'agrégation $\min$ pondérée permet de construire des solutions de compromis (cf. Figure 1.4, de la présente page). Si tous les objectifs sont à maximiser alors la maximisation d'une fonction d'agrégation $\max$ pondérée permet de construire des solutions de compromis (cf. Figure 1.3, page ci-contre).

Définition 1.8. Norme de Tchebychev [EG00].
Pour un point $y \in \mathbb{R}^p$ dans l'espace des objectifs, la norme de Tchebychev $\Upsilon_\alpha : \mathbb{R}^p \to \mathbb{R}$ est définie comme une mesure de distance par rapport à un point $y^\star \in \mathbb{R}^p$ qui peut être le point idéal du problème (y^I) ou un point défini a priori :

$$\max_{k\in P} \alpha_k \times |y_k - y_k^\star|$$

avec $\alpha \in \mathbb{R}^p$ un vecteur de poids défini a priori.
La norme de Tchebychev augmentée Υ_α^+ est :

$$\max_{k\in P} \alpha_k \times |y_k - y_k^\star| + \epsilon \times \sum_{k=1}^{p} \alpha_k \times y_k$$

avec $\epsilon \in \mathbb{R}_+$ une petite valeur positive.

Même si les objectifs ont des sens d'optimisation différents, la minimisation de la norme de Tchebychev par rapport au point idéal des solutions efficaces permet de construire des solutions de compromis (cf. Figure 1.5, page suivante) (grâce à l'utilisation de la valeur absolue). De plus le vecteur de poids α permet de prendre en compte des objectifs définis sur des échelles différentes. Ainsi la norme Tchebychev peut être utilisée dans un plus grand nombre de problèmes que les fonctions d'agrégation min ou max. Cette approche est une des plus utilisées pour construire des solutions de compromis [EG00, GP06a, SN10].

Une solution optimale selon une norme de Tchebychev Υ_α est nécessairement faiblement efficace et il existe au moins une solution optimale selon Υ_α qui est efficace. Cette propriété est aussi vérifiée pour les fonctions d'agrégation $\min$ et $\max$. Si $\lambda_k > 0, \forall k \in P$ alors les solutions optimales selon la norme de Tchebychev augmentée Υ_α^+ sont nécessairement efficaces. L'optimisation de la norme de Tchebychev

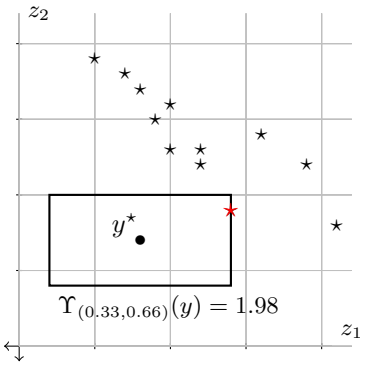

Figure 1.5 – Minimisation d'une norme de Tchebychev $\Upsilon_{(0.33,0.66)}$ par rapport au point y^*. Le trait en gras représente la courbe de niveau de la norme de Tchebychev. Les étoiles ($\star$) représentent les points des solutions du problème.

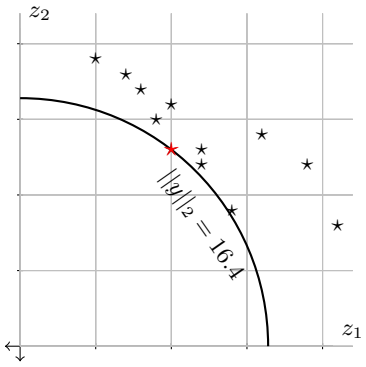

Figure 1.6 – Minimisation de la norme euclidienne $||.||_2$. Le trait en gras représente la courbe de niveau de la norme euclidienne. Les étoiles ($\star$) représentent les points des solutions du problème.

dans un problème MOCO avec des objectifs additifs est en général un problème d'une complexité NP-complet [EG00], cela est notamment le cas dans le cadre du problème de plus court chemin multi-objectif [MH92]. La minimisation de la norme euclidienne $||y||_2 = \sqrt{\sum_{k \in P}(y_k)^2}$ permet de construire des solutions de compromis [PMRS03, NM99] (cf. Figure 1.6, de la présente page). D'autres normes, $||.||_3$ ou $||.||_4$ par exemple, peuvent être utilisées pour construire des solutions de compromis.

1.1.2.4 Optimisation d'une fonction d'utilité

Dans le chapitre suivant (cf. §1.2.3.2, page 28) nous définissons précisément la notion de *fonction d'utilité*. Considérons pour l'instant qu'une fonction d'utilité $U : \mathbb{R}^p \to \mathbb{R}$ est une fonction d'agrégation des objectifs qui généralise la notion de somme pondérée, de fonction d'agrégation $\max$, de fonction d'agrégation $\min$ et de norme de Tchebychev.

Définition 1.9. Fonction d'utilité croissante.
Une fonction d'utilité $U : \mathbb{R}^p \to \mathbb{R}$ est croissante *si pour deux points quelconques $y^a, y^b \in \mathbb{R}^p_+$ dans l'espace des objectifs :*

$$\forall k \in P,\ y^a_k \leq_k y^b_k \Rightarrow U(y^a) \leq U(y^a)$$

Une fonction d'utilité $U : \mathbb{R}^p \to \mathbb{R}$ est strictement croissante *si pour deux points $y^a, y^b \in \mathbb{R}^p_+$ dans l'espace des objectifs :*

$$y^a \prec_P y^b \Rightarrow U(y^a) < U(y^a)$$

Théorème 1.1. Fonction d'utilité croissante et solutions efficaces.
Pour un problème MOCO *donné : si la fonction d'utilité U est* croissante *alors il existe au moins une solution optimale selon U qui est efficace. Si la fonction d'utilité est* strictement croissante *alors toutes les solutions optimales selon U sont efficaces.*

Le théorème 1.1 permet de montrer qu'une solution optimale selon une somme pondérée φ_λ avec un vecteur de poids strictement positifs ($\lambda_k > 0, \forall k \in P$) est une solution efficace. En effet une telle somme pondérée est une fonction strictement croissante. De manière identique, si les poids de la somme pondérée φ_λ sont positifs ou nuls alors au moins une solution optimale selon φ_λ est efficace puisque une telle somme pondérée est croissante. Il en est de même pour la norme de Tchebychev et la norme de Tchebychev augmentée.

1.1.2.5 Construction d'un ensemble représentatif de l'ensemble des solutions efficaces

Pour chacune des approches pour résoudre les problèmes MOCO, il existe au moins une solution efficace parmi les solutions optimales. Ainsi la construction d'un ensemble complet de solutions efficaces permet d'assurer qu'une solution optimale soit déterminée quelle que soit l'approche multi-objectif utilisée (cf. §1.3.1, page 37). La construction d'un ensemble complet de solutions efficaces est l'approche la plus communément adoptée pour la résolution d'un problème MOCO. Dans ce cas la notion d'optimalité est équivalent à la notion d'efficacité (cf. Définition 1.1, page 13).

La construction d'un ensemble complet de solutions efficaces est en général un problème "*intractable*" puisque le nombre de solutions efficaces de nombreux problèmes MOCO est exponentiel [EG00, Ehr00b]. C'est notamment le cas du problème de plus court chemin multi-objectif [Han80, Ser86].

1.1.2.6 Méthodes de résolution pour les différentes approches de l'optimisation multi-objectif

L'optimisation lexicographique et l'optimisation d'une somme pondérée peuvent en général être réalisées avec un algorithme mono-objectif. L'optimisation d'une fonction d'utilité ou d'une norme de Tchebychev est en général basée sur des algorithmes pour la construction d'un ensemble représentatif de solutions efficaces. Dans le cadre du problème de plus court chemin multi-objectif, l'optimisation d'une fonction d'utilité ou d'une norme de Tchebychev est détaillée dans la section 2.4. Les méthodes pour construire de façon approchée ou exacte un ensemble complet (ou représentatif) de solutions efficaces sont discutées dans les sous-sections suivantes (cf. §1.1.3, de la présente page) et (cf. §1.1.4, page suivante).

1.1.3 Méthodes de résolution approchées

Les méthodes de résolutions approchées calculent un ensemble de solutions $\tilde{E}(X)$ du problème MOCO qui "approxime" (ou approche) l'ensemble des solutions efficaces $E(X)$. Le terme "approxime" possède plusieurs significations. Il existe plusieurs méthodes (par exemple, l'hypervolume [ZBT07]) pour calculer la *proximité* de l'ensemble approché $\tilde{E}(X)$ avec l'ensemble exact $E(X)$. Il existe deux familles d'algorithmes MOCO approchés, les algorithmes avec garantie de performance qui garantissent a priori la proximité de l'ensemble approché et les métaheuristiques (et heuristique) qui ne garantissent pas la proximité de l'ensemble approché. Ces algorithmes sont particulièrement utiles quand le problème MOCO est *intractable*.

Les algorithmes avec garantie de performance proposent des algorithmes polynomiaux pour résoudre de façon approché les problèmes NP-complet. La proximité de $\tilde{E}(X)$ par rapport à $E(X)$ est définie telle que $\forall x \in \tilde{E}(X), \exists x' \in E(X), \forall k \in P, z_k(x) \leq (1+\epsilon) z_k(x')$ avec $\epsilon \in \mathbb{R}_+$ la variable d'approximation. Ces algorithmes utilisent des schémas d'approximation polynomiaux, qui sont en général dérivés d'algorithmes mono-objectif. Pour de nombreux problèmes MOCO il existe des algorithmes avec garantie de performance : le problème de plus court chemin multi-objectif [War87, TZ09] ; le problème d'affectation multi-objectif [PY00] ; le problème d'arbre couvrant multi-objectif [RG96] ; le problème de voyageur de commerce multi-objectif [Ehr00a]. Pour un état de l'art général, on réfère à [RW05].

Les métaheuristiques multi-objectif sont plus populaires que les algorithmes avec garantie de performance MOCO. Elles ne garantissent pas la proximité de l'ensemble $\tilde{E}(X)$ avec l'ensemble $E(X)$. Ehrgott et Gandibleux [EG04] proposent un état de l'art des métaheuristiques pour les problèmes MOCO, une bibliométrie plus générale des métaheuristiques multi-objectif est proposée par Jones et al [JMT02], Il existe deux familles de métaheuristiques : les algorithmes de recherche locale et les algorithmes à population.

- Parmi les algorithmes de recherche locale on cite les algorithmes tabou [GMF97] et les algorithmes GRASP [GVT98].
- Parmi les algorithmes à population on cite les algorithmes génétiques [Deb01, DPAM02, ZLT02], les algorithmes d'optimisation par essaims particulaires SMPSO [DGN+09] et les algorithmes de colonie de fourmis [ASG07].

Notons que les algorithmes génétiques sont les métaheuristiques multi-objectif les plus populaires [JMT02, WDF+08].

Dans ce cadre du problème de plus court chemin multi-objectif, la littérature rapporte quelques métaheuristiques spécifiques dérivées, par exemple, d'algorithmes génétiques [STW04, MPJ07, HQF07] ou d'algorithmes de colonie de fourmis [GN10].

1.1.4 Méthodes de résolution exactes

Ehrgott et Gandibleux [EG00] proposent un état de l'art de l'ensemble des méthodes (exactes et approchés) spécifiques aux problèmes MOCO.

Parmi les méthodes pour résoudre exactement un problème MOCO on différencie deux approches : les méthodes de résolution habituellement employées dans le cadre mono-objectif et qui sont généralisée au cadre multi-objectif (cf. §1.1.4.1, de la présente page), les méthodes de résolution spécifiquement multi-objectif qui utilisent dans des sous-routines des méthodes de résolutions mono-objectif (cf. §1.1.4.2, page suivante).

1.1.4.1 Généraliser des méthodes mono-objectif au cas multi-objectif

Concernant les méthodes mono-objectif généralisée au cas multi-objectif on notera : les méthodes séparation et évaluation (B&B) et les méthodes de programmation dynamique (DP).

Les méthodes B&B consistent à énumérer toutes les solutions guidé par une stratégie de recherche en profondeur ou en largeur sur un arbre de profondeur p défini sur $\{0,1\}^p$ (avec des variables binaires). Les nœuds de l'arbre correspondent à des vecteurs $x \in \{0,1\}^l$, avec l le niveau de l'arbre. Afin d'éviter de construire toutes les feuilles de l'arbre, on évalue les nœuds de l'arbre afin de déterminer si il existe une solution intéressante qui peut être construite à partir de ce nœud. Ces méthodes ont montré leur efficacité pour certains problèmes d'optimisation combinatoire mono-objectif, tel que le problème de sac à dos [MT90]. Néanmoins ces méthodes ne s'adaptent pas facilement au cas multi-objectif [TP02] car l'évaluation d'un nœud de l'arbre n'est pas triviale. En effet on ne cherche plus à construire une unique solution mais un ensemble de solutions efficaces. Un nœud de l'arbre peut par exemple être évalué en calculant le point idéal de l'ensemble des nœuds de l'arbre sous le nœud évalué. Si ce point idéal est dominé au sens de Pareto par une solution connue alors l'ensemble des solutions qui peuvent être construites à partir du nœud ne contient aucune solution intéressante. Une manière ensuite d'améliorer l'évaluation d'un nœud peut être de calculer un ensemble de points (plutôt que seulement un point idéal) qui domine l'ensemble des solutions qui peuvent être construites à partir du nœud [EG07]. Des méthodes

de séparation et évaluation ont été proposés pour le problème d'arbre couvrant multi-objectif [SS08] ou pour le sac à dos [DS10, Jor10].

Les méthodes de programmation dynamique [Bel57] (DP) proposent de résoudre une suite de sous-problèmes du problème d'optimisation combinatoire en augmentant la taille des sous-problèmes résolus au fur et à mesure afin de résoudre le problème original in fine. Contrairement aux méthodes B&B les méthodes DP sont totalement adaptées au type de problème d'optimisation combinatoire traité. Néanmoins une évaluation similaire à celle employée dans les algorithmes B&B peut parfois être utilisée. Ces méthodes s'appliquent très bien sur les problèmes de plus court chemin mono-objectif [Dij59, Bel58]. Pour certains problèmes les méthodes de programmation dynamique peuvent s'étendre au cas multi-objectif. C'est par exemple le cas du problème de plus court chemin multi-objectif [Han80, Mar84a] (cf. §2.2, page 50) mais aussi du problème de sac à dos multi-objectif [CCF+03, Jor10].

1.1.4.2 Méthodes spécifiques au multi-objectif

En plus de la généralisation des méthodes mono-objectif il existe plusieurs méthodes exploitant spécifiquement la structure d'un problème bi-objectif. Certaines de ces méthodes ont été adaptées pour optimiser plus de 2 objectifs.

- Les méthodes dites ϵ-contrainte [HLW71] proposent de supprimer le deuxième objectif (ou le premier) et d'ajouter une borne supérieure sur l'objectif supprimé afin d'obtenir un problème mono-objectif sur-contraint (cf. Figure 1.7, page suivante). À chaque itération ce problème est résolu et la borne supérieure est diminuée. Ce qui permet d'obtenir in fine un ensemble complet de solutions efficaces. Une solution x^0 lexicographique optimale ($(1) = 1$ et $(2) = 2$) est calculée dans un premier temps. Puis une procédure itérative permet de construire les autres solutions efficaces : à une itération t, la solution optimale x^t de cette itération est construite en résolvant le problème d'optimisation combinatoire mono-objectif (le premier objectif) où une contrainte sur le second objectif est ajoutée : $z_2(x) \leq z_2(x^{t-1}) + \epsilon$, avec $\epsilon \in [0, 1]$ une petite valeur positive (le pas) et x^{t-1} la solution optimale de l'itération $t-1$. Le pas ϵ doit être suffisamment petit de telle sorte qu'aucune solution efficace soit oubliée, ainsi ces méthodes s'adaptent mieux quand les objectifs sont définis sur l'ensemble des entiers. Pour que ces méthodes fonctionnent efficacement, il faut aussi pouvoir ajouter une contrainte sur un objectif au problème mono-objectif sans augmenter de façon significative la difficulté de résolution du problème mono-objectif. Cette méthode est utilisée dans le cadre du plus court chemin avec un objectif additif associé à un objectif bottleneck [Han80] ou à deux objectifs bottleneck [PBM09, PP10].
- Les méthodes dites "ranking" [CM81, CM82] proposent d'utiliser un algorithme de ranking (mono-objectif) afin de générer l'ensemble maximal complet des solutions efficaces. Un algorithme de ranking permet de construire toute les solutions du problème une à une selon leur valeur sur l'objectif considéré, de la meilleure à la moins bonne solution. L'algorithme de ranking permet de construire toutes les solutions selon le premier objectif jusqu'à construire une solution $x \in X$, $z_1(x) > y_1^N$ qui dépasse la borne supérieure des solutions efficaces sur le premier objectif (cf. Figure 1.8, page suivante). Toutes les solutions efficaces sont donc construites de cette manière. L'efficacité de cette méthode dépend de l'efficacité de l'algorithme de ranking mono-objectif utilisé. Notons que cette méthode construit clairement un nombre de solutions beaucoup plus grand que l'ensemble des solutions efficaces.
- Les méthodes dites dichotomie [Geo67, AN79] proposent de construire plusieurs sommes pondérées pour construire l'ensemble de solutions supportées extrêmes. La figure 1.9 montre que les sommes pondérées définies par leurs vecteurs de poids $\lambda^1, \ldots, \lambda^5$ permettent de trouver tous points supportés

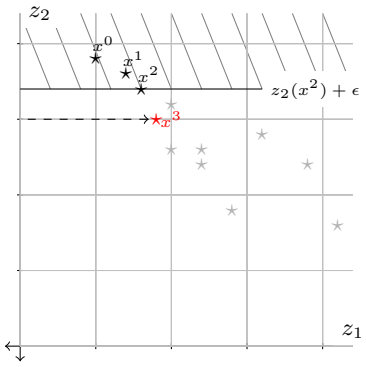

Figure 1.7 – Méthode ϵ-constraint. La solution lexicographique optimale x^0 est calculée, puis durant les trois itérations suivantes les solutions x^1, x^2 et x^3 sont calculées. La borne $z_2(x) < z_2(x^2) + \epsilon$ est utilisée pour calculer x^3 lors de la troisième itération.

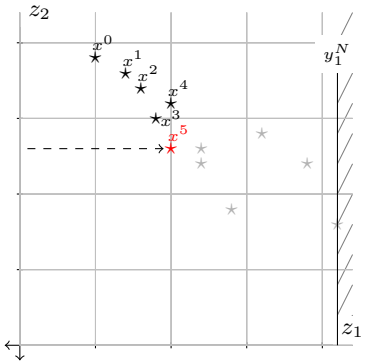

Figure 1.8 – Méthode de ranking. Les solutions x^1, x^2, x^3, x^4 et x^5 sont calculées après 5 itérations respectivement. Une borne $z_1(x) \leq y_1^N$ est utilisée, avec y^N le point nadir.

extrêmes. Ces méthodes sont en général efficaces puisque l'utilisation de somme pondérée permet de transformer le problème multi-objectif en un problème mono-objectif. Une extension pour le cas tri-objectif est donnée dans [PGE10a].

- La méthode dite 2-phase [UT95] propose de construire un ensemble complet de solutions supportées extrêmes pendant la première phase, puis de construire le reste des solutions efficaces pendant la deuxième phase (cf. Figure 1.10, page 23). L'ensemble des solutions supportées extrêmes sont généralement construites avec la méthode de dichotomie. Ensuite, à l'aide de l'ensemble des solutions supportées extrêmes, le reste des solutions efficaces peuvent être construites à l'aide d'un algorithme de DP multi-objectif [MMO91], d'un algorithme de B&B multi-objectif [Jor10] ou d'un algorithme de ranking mono-objectif [PGE08]. Przybylski et al proposent une généralisation pour des problèmes MOCO comportant plus de deux objectifs [PGE10b, PGE10a].
- Mentionnons encore la méthode de Degoutin et Gandibleux [DG02] (voir [TP03] pour une généralisation au multi-objectif) et la méthode de Sylva et Crema [SC04]. Ces méthodes reposent sur l'ajout de contraintes sur les objectifs et sur l'optimisation (à plusieurs reprises) d'une somme pondérée afin calculer un ensemble complet de solutions efficaces.

1.2 Modélisation des préférences en aide à la décision multi-critère

La première section 1.2.1 donne le cadre général de l'aide à la décision multi-critère, les sections 1.2.2,1.2.3,1.2.4,1.2.5 suivantes définissent le modèle de préférence utilisé dans le cadre de cette thèse. Dans cette section on utilise des notations spécifiques à l'aide à la décision multi-critère. Les notations qui sont utilisés dans le reste de la thèse sont définies dans la section suivante.

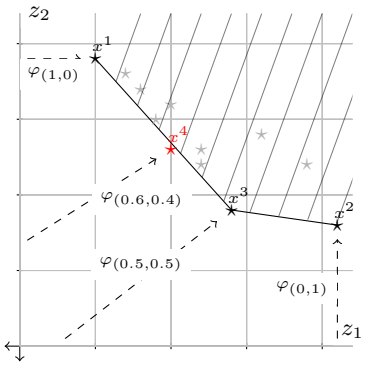

Figure 1.9 – Méthode de dichotomie. Les solutions x^1, x^2, x^3 et x^4 sont calculées après 4 itérations, grâce à l'optimisation des sommes pondérées $\varphi_{(1,0)}$, $\varphi_{(0,1)}$, $\varphi_{(0.5,0.5)}$ et $\varphi_{(0.6,0.4)}$ respectivement. La zone barrée ne sera pas explorée par la méthode de dichotomie.

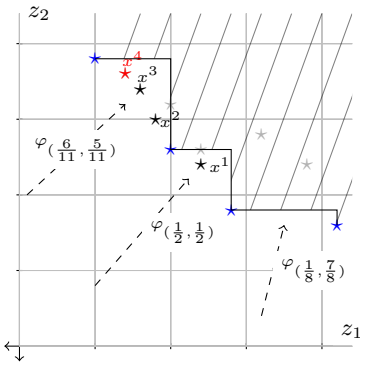

Figure 1.10 – Méthode en 2-phase, deuxième phase, calcul des solutions non supportées. ($\star$) : les solutions obtenues durant la première phase. Durant la deuxième phase : la solution x^1 a été obtenue par le ranking utilisant la somme pondérée $\varphi_{(\frac{1}{2},\frac{1}{2})}$. les solutions x^2, x^3 et x^4 ont été obtenues par le ranking utilisant la somme pondérée $\varphi_{(\frac{6}{11},\frac{5}{11})}$. La solution x^4 est la dernière solution obtenue. La zone barrée ne sera pas explorée durant la deuxième phase.

1.2.1 Aide à la décision multi-critère

Selon Roy [Roy05], l'aide à la décision peut être définie comme l'activité de personnes qui utilisent des modèles (mathématiques) afin d'aider le décideur durant le processus de décision. Des méthodes d'aide à la décision existent dans plusieurs domaines comme l'informatique décisionnelle et la recherche opérationnelle.

En général le décideur doit prendre en compte lors de la prise de décision différents points de vues, parfois contradictoires comme le coût et la qualité [Roy05]. Pour ces raisons il existe rarement un seul critère qui satisfasse à tout point de vue. Plusieurs critères de décision sont alors pris en compte. Le domaine de l'aide à la décision multi-critère (MCDA) [KR76, RB93, PBR00] propose de modéliser les préférences du décideur sur les différents critères de décision [GL05]. Ce domaine de recherche est en pleine expansion depuis les années 80 [WDF+08]. On notera des domaines proches qui partagent parfois des outils avec l'aide à la décision multi-critère : la décision dans l'incertain [KY97] ou les critères modélisent les conséquences des scénarios impliqués par la prise de décision et la décision collective [Mun04] où les critères modélisent les avis des personnes responsables de la prise de décision.

Dans une perspective d'aide à la décision multi-critère, un ensemble d'*alternatives* X et un ensemble d'attributs $P = \{1, \ldots, p\}$ sont définis. L'ensemble des alternatives X peut être défini en extension (chaque alternative de X est définie a priori) ou en compréhension (les alternatives doivent être construites, par exemple les alternatives sont les solutions d'un problème combinatoire). Chaque alternative $x \in X$ est définie par un nombre p d'attributs. Les attributs représentent l'information brute que le décideur possède au début du processus de décision. À chaque attribut $k \in P$ correspond un ensemble Ω_k tel que la fonction $z_k : X \to \Omega_k$ associe chaque alternative $x \in X$ sa valeur sur l'attribut k. Soit $\mathbf{\Omega} = \Omega_1 \times \cdots \times \Omega_p$ l'espace sur lequel sont définis les attributs, i.e. l'espace des attributs. Par abus de langage un critère est parfois utilisé comme équivalent à un attribut puisque chaque critère correspond un

attribut. Dans notre cadre de travail, on considère qu'un critère doit plus précisément permettre d'évaluer et comparer les alternatives selon un point de vue particulier [Roy05]. Ces propriétés sont détaillées dans la section 1.2.3. Dans le cadre de cette thèse on considère que pour chaque attribut $k \in P$, l'ensemble Ω_k est un sous-ensemble de $\mathbb{R}$. Par analogie avec l'optimisation multi-objectif on note $Y = z(X) \subset \mathbf{\Omega}$ l'ensemble des alternatives dans l'espace des attributs. Un vecteur d'attributs $y \in Y$ est appelé *point* car il correspond à une alternative dans l'espace des attribut. On dit que deux alternatives x^a et x^b sont équivalentes s.s.i. $z_k(x^a) = z_k(x^b)$ sur l'ensemble des attributs $k \in P$.
Il existe trois problématiques de décision [Roy85] : la *problématique du choix* où la "meilleure" alternative ; la *problématique du tri* où l'ensemble des alternatives est totalement ordonné selon les préférences du décideur ; la *problématique du rangement* où une classe est associée à chaque alternative. Dans le cadre du problème de choix la meilleure alternative peut être considérée comme la solution optimale au regard des préférences du décideur.

Dans le cadre de l'aide à la décision multi-critère, le processus de décision peut être divisé en trois étapes distinctes. Les travaux sur l'aide à la décision multi-critère s'intéressent en général à une seule de ces étapes. On définit ces trois étapes : (1) structuration multi-attribut du problème de décision ; (2) modélisation des préférences ; (3) recherche d'alternatives préférées (résolution du problème de décision).

1.2.1.1 Structuration multi-attribut du problème de décision

Dans un premier temps, il est nécessaire d'identifier l'ensemble des points de vue qui influent sur la qualité (dans le sens des préférences du décideur) d'une alternative. Ces points de vue permettent de définir différents attributs sur les alternatives. Ensuite un ensemble cohérent (P) de ces attributs [RB93] est construit de tel sorte que cet ensemble soit (1) exhaustif, i.e. il n'existe pas d'attributs en dehors de cet ensemble permettant de différencier les alternatives entre elles, et (2) non-redondant, i.e. tous les attributs pour différencier les alternatives entre elles. Les attributs peuvent ensuite être hiérarchisés de telle sorte que des sous-ensembles d'attributs définissent chacun un unique attribut. Par exemple un attribut de coût monétaire et un attribut de coût écologique permettent de définir un attribut de coût global. Cette étape du processus de décision n'est pas d'avantage détaillée dans cette thèse, pour plus d'information on reporte à la littérature MCDA [Kee96, Roy85].

1.2.1.2 Modélisation des préférences du décideur

La modélisation des préférences du décideur consiste à construire, à partir d'*informations préférentielles* du décideur, un modèle (mathématique) de préférence. Ce modèle de préférence doit permettre de comparer les alternatives entre elles, ce qui permet de proposer une recommandation au décideur.

1.2.1.3 Recherche d'alternatives préférées

Dans le cadre de la problématique du choix cette recommandation est un petit sous-ensemble d'alternatives de X composé d'alternatives choisies selon le modèle de préférence : des alternatives préférées. Quand l'ensemble des alternatives X est définie en extension et que cet ensemble X n'est pas trop grand, alors il suffit de comparer les alternatives les unes avec les autres à l'aide du modèle de préférence afin de sélectionner des alternatives préférées. La recherche d'alternatives préférées quand l'ensemble d'alternatives est défini en compréhension par un problème MOCO est discuté dans la section 1.3.

1.2.2 Modélisation des préférences en aide à la décision multi-critère

La modélisation des préférences consiste à construire un modèle de préférence à partir d'*informations préférentielles* du décideur. L'information préférentielle peut définir des préférences sur les critères/attributs ou sur les alternatives. L'information préférentielle peut être de type cardinal (évaluation d'une alternative (resp. critère) sur une échelle de satisfaction (resp. importance)) ou de type ordinal (comparaison entre les critères ou entre les alternatives). Dans la littérature [SGM05] en général on considère qu'il est difficile pour le décideur d'exprimer une information (fiable) de type cardinal ou une information (fiable) sur les critères. Ainsi l'information ordinale sur les alternatives peut être considérée comme l'information la plus facile à exprimer pour le décideur.

1.2.2.1 Ordre et préordre

Afin de définir l'information préférentielle de type ordinal des définitions sur les ordres sont données.

Définition 1.10. Ordre et Préordre [PBR00].
Soient deux relations binaires $\succ, \sim$ *et la relation binaire* $\succsim$ *définie comme l'union des deux relations* $\succ$ *et* $\sim$ *telle que,* $\forall x^a, x^b \in X : x^a \succ x^b \iff x^a \succsim x^b$ et $\neg(x^b \succsim x^a)$*;* $x^a \sim x^b \iff x^a \succsim x^b$ et $x^b \succsim x^a$.
Une relation $\succ$ *sur l'ensemble* X *est un* ordre *s.s.i.* $\forall x^a, x^b, x^c \in X$ *:*

- $\neg(x^a \succ x^a)$ *(irréflexivité)*
- $x^a \succ x^b$ et $x^b \succ x^c \Longrightarrow x^a \succ x^c$ *(transitivité)*
- $x^a \succ x^b \Longrightarrow \neg(x^b \succ x^a)$ *(asymétrie)*

La relation $\succ$ *est un* ordre complet *si la relation est définie pour chaque paire d'alternatives* $x^a, x^b \in X$ *:* $x^a \succ x^b$ *ou* $x^b \succ x^a$*. Dans le cas contraire* $\succ$ *est un* ordre partiel.
Une relation $\succsim$ *sur l'ensemble* X *est un* préordre *s.s.i.* $\forall x^a, x^b, x^c \in X$ *:*

- $x^a \succsim x^a$ *(réflexivité)*
- $x^a \succsim x^b$ et $x^b \succsim x^c \Longrightarrow x^a \succsim x^c$ *(transitivité)*

La relation $\succsim$ *est un* préordre complet *si la relation est définie pour chaque paire d'alternatives* $x^a, x^b \in X$ *:* $x^a \succ x^b$*,* $x^a \sim x^b$ *ou* $x^b \succ x^a$*. Dans le cas contraire* $\succsim$ *est un* préordre partiel.

La relation de dominance de Pareto (cf. Définition 1.1, page 13) entre les solutions est un ordre non complet.

1.2.2.2 Préférences du décideur

Supposons que le décideur exprime une information de type ordinal sur les alternatives du problème de décision, alors l'information préférentielle obtenue est une relation de préférences $\succ_{DM}, \sim_{DM}, \succsim_{DM}$ définie sur l'ensemble des alternatives X telle que pour deux alternatives $x^a, x^b \in X$:

- le décideur préfère x^a à x^b, $x^a \succ_{DM} x^b$;
- le décideur est indifférent entre x^a et x^b, $x^a \sim_{DM} x^b$.

La relation $\succsim_{DM}$ est définie comme l'union de $\succ_{DM}$ et $\sim_{DM}$ (cf. Définition 1.10). Par souci de simplicité on considère que les relations binaires $\succ_{DM}, \sim_{DM}, \succsim_{DM}$ sont aussi définies sur $\mathbf{\Omega}$ de telle sorte que $z(x^a) \succsim_{DM} z(x^b)$ s.s.i. $x^a \succsim_{DM} x^b$. En supposant la rationalité du décideur, la relation de préférence $\succsim_{DM}$ correspond à un préordre sur X et $\succ_{DM}$ correspond à un ordre sur X [PBR00].

Si l'ensemble des alternatives X n'est pas trop grand, alors il est possible de demander au décideur de construire une relation de préférence (préordre) $\succsim_{DM}$ complète. À l'aide du préordre $\succsim_{DM}$ complet les alternatives préférées peuvent être déduites directement. Dans le cas contraire un modèle de préférence doit être construit afin de comparer les alternatives de manière automatique.

1.2.2.3 Modèle de préférence

Un modèle de préférence doit permettre de construire un préordre $\succsim$ complet qui vérifie la relation de préférence $\succsim_{DM}$ du décideur : $\forall x^a, x^b \in X$, $x^a \succsim_{DM} x^b$ alors le modèle de préférence doit préférer x^a à x^b. De nombreuses méthodes existent pour construire un modèle de préférence [FGE05]. La plupart de ces méthodes peuvent être rangées dans une des deux familles suivantes :

- Les méthodes de *surclassement* proposent de construire un modèle de préférence permettant uniquement de comparer les alternatives deux à deux. Les alternatives sont comparées sur chacun des critères puis une agrégation de ces comparaisons permet d'établir si une alternative est préférée à une autre. Ces méthodes constituent l'école française [WDF+08] de l'aide à décision multi-critère fondée par B. Roy. Parmi ces méthodes citons : la méthode ELECTRE [Roy68, FMR05, FGRS10] et la méthode PROMETHEE [BV85, FMR05].
- Les méthodes de *critère unique de synthèse* proposent de synthétiser (ou agréger) tous les critères en un unique critère. Grâce au critère de synthèse obtenu il est possible de de comparer toutes les solutions entre elles puisque ce critère unique défini un préordre complet. Parmi ces méthodes citons : la méthode *Analytic Hierarchy Process* (AHP) [Saa77, eCCV05, OS06] et les méthodes issues du cadre *Multi-Attribute Utility Theory* (MAUT) [KR76] : MACBETH [eCV94], UTA [JLS82] (cf. §1.2.3, de la présente page).

La famille des méthodes à critère unique de synthèse est la famille la plus populaire dans la littérature [WDF+08]. Néanmoins, ces deux familles de méthodes pour construire un modèle de préférence ont chacune des avantages et des inconvénients. Ainsi les méthodes de surclassement sont particulièrement utiles quand les critères ne peuvent pas être rendus commensurables entre eux, i.e. quand il est difficile de comparer les performances d'une alternative sur les différents critères. Les méthodes à critère unique de synthèse permettent d'obtenir une évaluation pour chaque alternative qui ne dépend pas des autres alternatives. Dans le cadre de la problématique du choix, une méthode à critère unique de synthèse permet de savoir rapidement si une alternative est meilleure (préférée) à toutes les autres. Pour cela il suffit de comparer cette alternative avec la meilleure alternative connue. Ceci n'est pas toujours possible avec les méthodes de surclassement pour lesquelles l'alternative doit être comparée à l'ensemble des autres alternative. Dans le cadre de l'optimisation combinatoire multi-objectif qui correspond à la problématique du choix, il semble raisonnable d'utiliser un modèle de préférence construit avec une méthode à critère unique de synthèse.

Notons aussi qu'une fois construit, un modèle de préférence permet alors non seulement de reproduire les préférences du décideur sur un ensemble d'alternatives X très grand, mais aussi de reproduire les préférences du décideur sur une instance (i.e. un ensemble d'alternatives) du problème MCDA comparable à l'ensemble d'alternatives sur lequel a été défini le modèle de préférence.

1.2.3 Théorie de l'utilité multi-attribut

La théorie MAUT [KR76, Dye05] propose un cadre pour la modélisation multi-critère. Les méthodes MAUT sont des méthodes à critère unique de synthèse. Le critère unique de synthèse est défini sur une échelle de satisfaction ξ qui, sans perte de généralité, est définie comme un sous-ensemble de $\mathbb{R}$. Une

fonction d'utilité (ou de valeur) $U : \mathbf{\Omega} \rightarrow \xi \subset \mathbb{R}$ est construite. Cette fonction d'utilité est le modèle de préférence. Cette fonction associe à chaque vecteur dans l'espace des attributs $\mathbf{\Omega}$ un niveau de satisfaction défini dans ξ.

Définition 1.11. Le modèle MAUT [Dye05].
Soit $U : \mathbf{\Omega} \rightarrow \xi \subset \mathbb{R}$ *une* fonction d'utilité MAUT. *Pour un point* $y \in \mathbf{\Omega}$ *défini dans l'espace des attributs,* l'utilité $U(y)$ *de ce point est définie par :*

$$U(y) = V(u_1(y_1), \ldots, u_k(y_k), \ldots, u_p(y_p)) \tag{1.3}$$

Avec,

- $u_1, \ldots, u_p$ *les* fonctions d'utilité partielles *;* $u_k : \Omega_k \rightarrow \xi$ *donne le niveau de satisfaction (défini sur* ξ*) de chaque valeur de* Ω_k *;*
- $V : \xi^p \rightarrow \xi$ *une* fonction d'agrégation.

Dans le cadre de la théorie MAUT, un critère $k \in P$ est constitué d'un attribut $k \in P$ et d'une fonction d'utilité partielle u_k. Chaque fonction d'utilité partielle u_k, $k \in P$ permet d'établir une relation de préférence entre les valeurs prises par l'attribut sur Ω_k. Ces fonctions d'utilité partielles permettent ainsi d'obtenir la *commensurabilité* entre les critères.

Il existe plusieurs méthodes dans la littérature pour construire une fonction d'utilité MAUT, notons la méthode UTA [JLS82, SGM05] et la méthode MACBETH [eCV94, eCCV05]. La méthode UTA est décrite dans le paragraphe §1.2.3.3. D'autres méthodes telles, que la méthode GRIP [FGS09], permettent d'utiliser sans la construire une fonction d'utilité MAUT.

1.2.3.1 Fonctions d'utilité partielles linéaires par morceaux

Dans le cadre de cette thèse on considère que chaque attribut est défini sur un sous-ensemble de réels : $\Omega_k \subset \mathbb{R}$. Chaque fonction d'utilité partielle u_k, $k \in P$ est donc une fonction arithmétique définie sur le sous-ensemble de réels Ω_k.

Des méthodes à critère unique de synthèse telles que AHP associent un poids (ou importance) $\lambda_k \in \mathbb{R}$ à chaque critère $k \in P$. Le poids associé à chaque critère définit alors une fonction d'utilité partielle u_k linéaire : $u_k(y_k) = \lambda_k \times y_k$, $\forall y_k \in \Omega_k \subset \mathbb{R}$. Considérer que chaque fonction d'utilité partielle est une fonction linéaire peut poser problème puisque la perception de préférence du décideur n'est pas nécessairement linéaire par rapport à l'échelle des attributs (cf. Exemple 1.1, de la présente page).

Exemple 1.1. Fonctions d'utilité partielles.
Pour un attribut $k \in P$ *définissant la distance à parcourir pour un chemin (alternative), mesuré en kilomètres* $\Omega_k = \{0km, \ldots, 1000km\}$*, le critère équivalent sera défini sur une échelle* $\xi = [0, \ldots, 1]$ *évaluant la satisfaction du décideur quant à la distance de l'alternative. Une fonction* $u_k : \Omega_k \rightarrow \xi$ *telle que* $u_k(y) = \frac{y}{1000}$ *permet d'assurer que le critère* k *soit commensurable par rapport aux autres critères eux-mêmes définis sur* ξ*. Néanmoins la fonction* u_k *ne modélise pas nécessairement la satisfaction du décideur concernant les distances. Ainsi considérons que le décideur perçoit peu de différence de satisfaction entre une distance de* $200km$ *et une distance de* $210km$*, par contre il perçoit une importante différente de satisfaction entre une distance de* $190km$ *et une distance de* $200km$*. Alors une fonction d'utilité partielle modélisant précisément les préférences du décideur sur l'attribut* k *est nécessairement non linéaire, ce qui n'est pas le cas de la fonction* u_k *définie auparavant.*

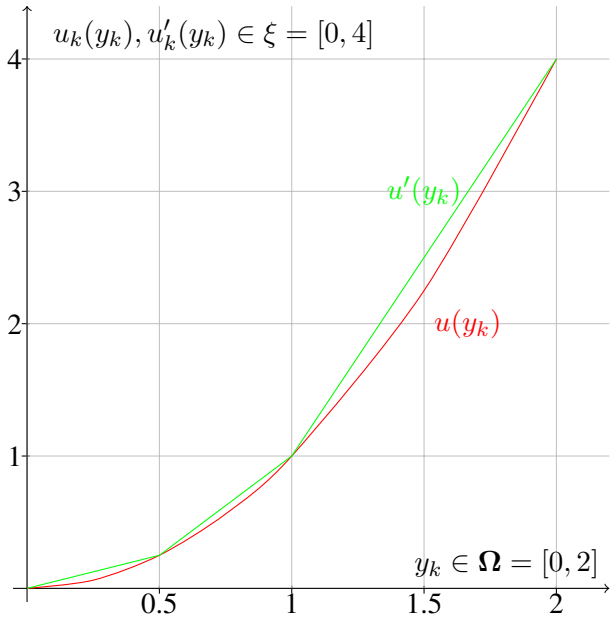

Figure 1.11 – Soient $k \in P$ un critère et y_k la valeur d'une alternative sur l'attribut k. $u_k, u'_k : [0, 2] \rightarrow [0, 4]$ sont deux fonctions d'utilité partielles, telles que $u_k(y_k) = y_k \times y_k$ est une fonction continue et u'_k une fonction linéaire par morceaux qui approxime u_k.

Afin de poser un cadre formel permettant la construction des fonctions d'utilité partielles $u_1, \ldots, u_p$ à partir d'une relation de préférence $\succsim_{DM}$ établie par le décideur, on considère que les fonctions d'utilité partielles sont continues et *linéaires par morceaux* (cf. Définition 1.12). Ce type de fonction est notamment utilisé dans les méthodes UTA [JLS82] et MACBETH [eCV94], ces fonctions permettent d'approximer n'importe quelle autre fonction arithmétique (cf. Figure 1.11).

Afin de définir une fonction d'utilité linéaire par morceaux les espaces de définitions des attributs $\Omega_1, \ldots, \Omega_p$ doivent être séparés en intervalles. Soient u_k la fonction d'utilité partielle définie sur l'attribut $k \in P$, et $\Omega_k \subseteq \mathbb{R}$ l'espace de définition de l'attribut k. Ω_k est séparé en τ_k intervalles, sur chacun de ces intervalles la fonction d'utilité partielle u_k est linéaire. Par exemple pour la fonction d'utilité partielle de la figure 1.11 l'espace de définition Ω_k de l'attribut k est $[0, 2]$, cet espace est séparé en trois ($\tau_k = 3$) intervalles $[0, 0.5]$, $]0.5, 1]$ et $]1, 2]$. $z_k^1 = 0, \ldots, z_k^{\tau_k+1} \leq +\infty$ sont les bornes qui définissent ces intervalles. Dans notre exemple ces bornes sont : $z_k^1 = 0$, $z_k^2 = 0.5$, $z_k^3 = 1$ et $z_k^4 = 2$. L'utilité de chacune de ces bornes est : $u_k(0) = 0$, $u_k(0.5) = 0.25$ $u_k(1) = 1$ et $u_k(2) = 4$. Les méthodes UTA (cf. §1.2.3.3, page suivante) et MACBETH permettent de calculer l'utilité pour chacune de ces bornes $u_k(z_k^1), \ldots, u_k(z_k^{\tau_k})$. En connaissant l'utilité de chacune de ces bornes il est possible de calculer l'utilité de n'importe quelle valeur de Ω_k (cf. Définition 1.12, de la présente page).

Définition 1.12. Fonction d'utilité partielle linéaire par morceaux [SGM05].
Soit $u_k : \Omega_k \rightarrow \xi$ la fonction d'utilité partielle définie sur l'attribut $k \in P$, avec $\Omega_k \subseteq \mathbb{R}$ l'espace de définition de l'attribut k et une échelle de satisfaction commune $\xi \subseteq \mathbb{R}$. Ω_k est divisé en τ_k intervalles. $z_k^1 = 0, \ldots, z_k^{\tau_k+1} \leq +\infty$ sont les bornes qui définissent ces intervalles : $\forall j \in \{1, \ldots, \tau_k\} : z_k^j < z_k^{j+1}$.
L'utilité pour chacune de ces bornes $u_k(z_k^1)$, …, $u_k(z_k^{\tau_k})$ est définie a priori.
L'utilité d'une alternative $x \in X$ sur le k-ème critère est définie par une interpolation linéaire : $z_k(x) \in$

$[z_k^j, z_k^{j+1}]$ *avec* $j \in \{1, \ldots, \tau_k\}$,

$$u_k(y_k) = u_k(z_k^j) + \theta_k^j \times (z_k(x) - z_k^j) \tag{1.4}$$

Avec $\theta_k^j = \frac{u_k(z_k^{j+1}) - u_k(z_k^j)}{z_k^{j+1} - z_k^j}$ *le coefficient d'inclination de* u_k *sur le* j*-ème intervalle.* $\Theta_k = \{\theta_k^1, \ldots, \theta_k^{\tau_k}\}$ *est l'ensemble des coefficients d'inclination* u_k.

1.2.3.2 Fonction d'agrégation et fonction d'utilité

La fonction d'agrégation $V : \xi^p \to \xi$ permet d'agréger les utilités des critères pour chaque alternative. Ainsi chaque alternative est évaluée par une valeur sur une échelle commune de satisfaction, i.e. un critère de synthèse. Les fonctions d'agrégation présentées précédemment peuvent être utilisée : somme pondérée (cf. Définition 1.5, page 15), fonction min ou max (cf. Définition 1.7, page 16). Les fonctions d'agrégation sont discutées dans la sous-section 1.2.4 suivante.

La fonction d'utilité U obtenue doit permettre de reproduire aussi bien que possible les préférences du décideur sur l'ensemble des alternatives : pour deux alternatives $x^a, x^b \in X$, $U(z(x^a)) > U(z(x^b)) \Longrightarrow x^a \succsim_{DM} x^b$, x^a préférée à x^b. Ainsi, dans le cadre de la problématique du choix, l'alternative préférée est l'alternative $x^a \in X$ ayant la plus grande utilité $U(z(x^a))$ parmi l'ensemble des alternatives : $U(z(x^a)) \geq U(z(x^b)), \forall x^b \in X$.

1.2.3.3 Construire de la fonction d'utilité : exemple de la méthode UTA

La méthode *UTilités Additives* (UTA) [JLS82, SGM05] est une des premières méthodes MAUT proposées dans la littérature. Cette méthode propose de construire une fonction d'utilité U : $U(y) = \sum_{k=1}^{p} u_k(y_k)$, $\forall y \in \mathbf{\Omega}$, avec $\forall k \in P$, $u_1, \ldots, u_p$ des fonctions d'utilité partielles linéaires par morceaux croissantes (cf. Définition 1.12). Notons que la fonction d'agrégation de U est la somme. L'échelle commune de satisfaction $\xi = [0, 1]$ est définie entre 0 et 1. Soit X l'ensemble des alternatives. Pour construire cette fonction d'utilité, le décideur doit définir un ordre non complet $\succ_{DM}$ (et une équivalence $\sim_{DM}$) de ses préférences entre des alternatives de X (cf. §1.2.2, page 24). Cet ordre non complet $\succ_{DM}$ sur X est l'information préférentielle du décideur qui est exploitée pour construire la fonction d'utilité. Cette méthode propose une régression depuis l'ordre non complet $\succ_{DM}$ vers l'ordre complet $>_U$ induit par la fonction d'utilité U : $x^a >_U x^b$ s.s.i. $U(z(x^a)) > U(z(x^b))$. Cette régression implique que pour deux alternatives $x^a, x^b \in X$, si x^a est préférée à x^b par le décideur ($x^a \succ_{DM} x^b$) alors l'utilité de x^a est plus importante que l'utilité de x^b ($U(x^a) > U(x^b)$).

Cette régression peut être modélisée avec problème linéaire (cf. Équation 1.5). La résolution de ce problème linéaire permet alors de construire une fonction d'utilité U. Il n'est pas toujours possible de construire une fonction d'utilité U pour laquelle $>_U$ constitue une régression exacte de $\succ_{DM}$ (cf. Exemple 1.2, page suivante), on parle alors d'*incohérences*. Dans le cas de la méthode UTA cela signifie qu'il n'existe pas de fonctions partielles linéaires par morceaux croissantes $u_1, \ldots, u_p$ telles que : $\forall x^a, x^b \in X$, $x^a \succ_{DM} x^b \Rightarrow \sum_{k=1}^{p} u_k(z_k(x^a)) > \sum_{k=1}^{p} u_k(z_k(x^b))$. La méthode UTA prend en compte ces incohérences par une erreur d'évaluation $\sigma(x) \in \mathbb{R}_+$ pour chaque alternative $x \in X$. Le problème linéaire pour construire cette fonction d'utilité minimise la somme des erreurs d'évaluation des alternatives. Une petite valeur strictement positive $\delta \in \mathbb{R}$, $\delta > 0$ permet de modéliser une préférence $\succ_{DM}$ entre deux alternatives.

$$
\begin{array}{lll}
\min & \sum_{x\in X}\sigma(x) & (a)\\
\text{s.c.} & (U(z(x^a))+\sigma(x^a))-(U(z(x^b))+\sigma(x^b))\geq\delta \quad \forall x^a,x^b\in X \text{ t.q. } x^a\succ_{DM}x^b & (b)\\
& (U(z(x^a))+\sigma(x^a))-(U(z(x^b))+\sigma(x^b))=0 \quad \forall x^a,x^b\in X \text{ t.q. } x^a\sim_{DM}x^b & (c)\\
& u_k(z_k^j)\leq u_k(z_k^{j+1}) \quad \forall k\in P,\ \forall j\in\{1,\ldots,\tau_k-1\} & (d)\\
& \sum_{k\in P}u_k(z_k^{\tau_k})=1 & (e)\\
& u_k(z_k^1)=0 \quad \forall k\in P & (f)\\
& \sigma(x^a)\in\mathbb{R}_+ \quad \forall x\in X & (g)\\
& u_k(z_k^j)\in[0,1] \quad \forall k\in P,\ \forall j\in\{1,\ldots,\tau_k-1\} & (h)
\end{array}
\tag{1.5}
$$

Le programme linéaire (1.5) est défini de la manière suivante : la fonction objectif (a) minimise les incohérences ; les équations (b) et (c) définissent la régression entre $\succ_{DM}$, $\sim_{DM}$ et U ; l'équation (d) implique que les fonctions d'utilité linéaires par morceaux sont croissantes ; les équations (e) et (f) permettent de normaliser la fonction d'utilité U entre 0 et 1 ; les équations (g) et (h) définissent les variables du problème linéaire.

Concernant la méthode MACBETH [eCCV05], la fonction d'utilité est une somme pondérée avec des fonctions d'utilité partielles linéaires par morceaux : $U=\sum_{k=1}^{p}\lambda_k\times u_k(y_k)$. La méthode MACBETH demande dans un premier temps au décideur d'exprimer ses préférences séparément sur chaque critère, i.e. d'exprimer ses préférences sur des valeurs $y_k\in\Omega_k$ de chaque attribut. Grâce à cette information préférentielle les fonctions d'utilité partielles linéaires par morceaux sont construites. Puis dans un deuxième temps cette méthode demande au décideur d'exprimer ses préférences sur des alternatives $x\in X$. Cette information préférentielle permet de construire les poids de chaque critère : $\lambda_1,\ldots,\lambda_p$. La méthode MACBETH nécessite une quantité d'information préférentielle moindre par rapport à la méthode UTA.

1.2.4 Indépendance préférentielle et interactions entre critères

La plupart des méthodes MAUT (cf. §1.2.3, page 26) telles que les méthodes UTA, MACBETH proposent d'utiliser comme fonction d'agrégation une somme pondérée : φ_λ (cf. Définition 1.5, page 15). L'agrégation des critères par une somme pondérée implique que les critères respectent la propriété d'*indépendance préférentielle* (cf. Définition 1.13 et Exemple 1.2).

Définition 1.13. Indépendance préférentielle [BP05]
Soient $y^c,y^d\in\xi^p$ des vecteurs d'utilités et $A,B\subset P$ deux ensembles de critères t.q. $A\bigcup B=P$ *et* $A\bigcap B=\emptyset$. *On note (y^c_A,y^d_B) le vecteur composé des utilités y^c_k pour chaque critère $k\in A$ et des utilités y^d_k pour chaque critère $k\in B$.*
La relation de préférence $\precsim_{DM}$ vérifie l'indépendance préférentielle *s.s.i.* $\forall y^c,y^d\in\xi^p$, $\forall A\subset P$:

$$\exists y^e\in\xi^p,\ (y^c_A,y^e_{P\setminus A})\precsim_{DM}(y^d_A,y^e_{P\setminus A})\Longrightarrow\forall y^f\in\xi^p,\ (y^c_A,y^f_{P\setminus A})\precsim_{DM}(y^d_A,y^f_{P\setminus A})$$

Selon Bouyssou et Pirlot [BP05], une fonction d'utilité construite dans le cadre MCDA doit vérifier l'indépendance préférentielle, i.e. la relation de préférence induite par la fonction d'utilité vérifie l'indépendance préférentielle. Il existe néanmoins de nombreuses applications dans le cadre MCDA qui n'utilisent pas cette hypothèse [Gra96]. L'exemple 1.2 illustre les limites de l'indépendance préférentielle.

Exemple 1.2. Limites de l'indépendance préférentielle [GL05].
Un directeur pédagogique souhaite construire une fonction d'utilité U afin de sélectionner les meilleurs

étudiants qui postulent pour un cursus en management. Les étudiants sont évalués sur 3 matières, mathématiques, statistiques et langues. Les évaluations des étudiants sur les 3 matières sont données sur une échelle de 1 à 10.
Les postulants ont généralement un bagage scientifique important, alors, pour le directeur les mathématiques et les statistiques sont importantes comparées aux langues. Néanmoins il ne souhaite pas favoriser les étudiants avec un bagage scientifique important et des manquements importants en langage. De plus, les mathématiques et les statistiques sont considérées comme étant redondantes puisque habituellement un étudiant bon en statistiques est aussi bon en mathématiques et inversement. Par conséquent, pour des étudiants bons en mathématiques le directeur préféra un étudiant bon en langues à un étudiant bon en statistiques.
Considérons l'étudiant x^a :

	mathématiques	*statistiques*	*langues*
étudiant x^a	*16*	*13*	*7*

L'étudiant x^a est fortement pénalisé par ses faibles performances en langues. Ainsi le directeur préfère un élève moins bon en statistiques mais meilleur en langues. Le décideur préfère donc l'étudiant suivant :

	mathématiques	*statistiques*	*langues*
étudiant x^b	*16*	*11*	*9*

Ainsi le directeur préfère x^b à x^a : $x^b \succ_{DM} x^a$. Considérons maintenant des étudiants mauvais en mathématiques. Dans ce cas, comme les postulants sont supposés avoir un bagage scientifique important, un étudiant bon en statistiques est préféré à un étudiant bon en langues. Soient x^c, x^d deux étudiants mauvais en mathématiques :

	mathématiques	*statistiques*	*langues*
étudiant x^c	*6*	*13*	*7*
étudiant x^d	*6*	*11*	*9*

Considérons les arguments donnés précédemment alors x^c est préféré à x^d ($x^c \succ_{DM} x^d$), malgré les faibles performances de x^c en langues. Soit un vecteur de poids $\lambda = (\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}^p_+$ avec λ_1 le poids des mathématiques, λ_2 le poids des statistiques et λ_3 le poids des langues. φ_λ est la somme pondérée définie par le vecteur de poids λ qui aggrége les notes des élèves en une note globale (i.e. un critère unique de synthèse).
On cherche alors à construire une somme pondérée φ_λ respectant les préférences du directeur. Le directeur préfère l'élève x^b à l'élève x^a alors $\varphi_\lambda((16, 13, 7)) < \varphi_\lambda((16, 11, 9))$, ce implique que les langues sont plus importantes que les statistiques, i.e. $\lambda_3 > \lambda_2$. Le directeur préfère l'élève x^c à l'élève x^d alors $\varphi_\lambda((6, 11, 9)) < \varphi_\lambda((6, 13, 7))$, ce implique que les statistiques sont plus importantes que les langues, i.e. $\lambda_2 > \lambda_3$. Alors n'existe alors pas de vecteur de poids $\lambda \in \mathbb{R}^p_+$ tel que la somme pondérée φ_λ respecte les préférences du directeur : $\varphi_\lambda((16, 13, 7)) < \varphi_\lambda((16, 11, 9))$ et $\varphi_\lambda((6, 11, 9)) < \varphi_\lambda((6, 13, 7))$. De telles préférences sont un exemple typique d'interaction entre le critère de mathématiques et le critère de statistiques.

De plus si l'aide à la décision est prise dans un cadre large, englobant certaines méthodes d'optimisation robuste [KY97] où d'intelligence artificielle [PS05], alors l'indépendance préférentielle n'est pas vérifiée de manière générale.

Dans la littérature MCDA on trouve trois comportements pour agréger les critères :

- *Pondération* des critères : pour le décideur les critères ont une importance différente les uns des autres. La pondération peut se faire à l'aide d'une somme pondérée ou une moyenne arithmétique pondérée.
- *Complémentarité* des critères : pour le décideur il existe une synergie entre les critères. Ainsi il préfère une alternative moyenne sur tous les critères (équilibrée) plutôt qu'une alternative bonne sur certains critères mais mauvaise sur les autres. La complémentarité peut se modéliser avec la fonction $\min$ ou la norme de Tchebychev (avec $y^\star = y^N$ le point nadir) (cf. §1.1.2.3, page 16). La notion de complémentarité entre les critères peut être associée avec la notion d'*interaction positive* entre les critères et la notion de *compromis*. Dans l'exemple 1.2 les critères mathématiques et langues sont complémentaire de même que les critères statistiques et langues.
- *Substituabilité* des critères : pour le décideur les critères peuvent se compenser, ainsi il préfère une alternative bonne sur un seul critère et mauvaise sur les autres à une alternative moyenne sur tous les critères. La substituabilité peut se modéliser à l'aide d'une fonction $\max$, d'une norme de Tchebychev (avec $y^\star = y^I$ le point idéal) et d'une norme euclidienne (cf. §1.1.2.3, page 16). On associe à la notion d'*interaction négative* avec la notion de substituabilité. Dans l'exemple 1.2 les critères mathématiques et statistiques sont substituables.

Certaines fonctions d'agrégation permettent de modéliser plusieurs décisions habituelles simultanément. La fonction *Ordered Weighted Averaging* OWA (cf. Définition 1.14) permet de modéliser la complémentarité et la substituabilité en même temps (cf. Figure 1.12) L'intégrale de Choquet (cf. §1.2.5, de la présente page) permet de modéliser la pondération, la complémentarité et la substituabilité en même temps (cf. Figure 1.13).

Définition 1.14. Fonction OWA [Yag88, GS07].
Pour un point $y \in \xi^p$ représentant une alternative, la fonction OWA $\Psi_\alpha^{owa} : \xi^p \to \xi$ *avec $\alpha \in \mathbb{R}_+^p$ un vecteur de poids est définie par :*

$$\Psi_\alpha^{owa}(y) = \sum_{k \in P} \alpha_k \times y_{(k)}$$

avec $(.)$ une permutation telle que $y_{(1)} \leq \cdots \leq y_{(p)}$.

Définition 1.15. Fonction d'aggrégation concave ou convexe ou linéaire
Soit $U : \xi^p \to \xi$ une fonction d'aggrégation :

- *U est convexe s.s.i. $\forall y^a, y^b \in \xi^p$ et $\forall \lambda \in [0,1]$:*

$$U(\lambda.y^a + (1-\lambda).y^b) \leq \lambda.U(y^a) + (1-\lambda).U(y^b)$$

- *U est concave s.s.i. $\forall y^a, y^b \in \xi^p$ et $\forall \lambda \in [0,1]$:*

$$U(\lambda.y^a + (1-\lambda).y^b) \geq \lambda.U(y^a) + (1-\lambda).U(y^b)$$

- *U est linéaire s.s.i. $\forall y^a, y^b \in \xi^p$ et $\forall \lambda \in [0,1]$:*

$$U(\lambda.y^a + (1-\lambda).y^b) = \lambda.U(y^a) + (1-\lambda).U(y^b)$$

Dans le cadre de la maximisation d'une fonction d'agrégation V, si V est concave alors aucune interaction négative n'est modélisée, si V est convexe alors aucune interaction positive n'est modélisée, Dans le cadre de la minimisation d'une fonction d'agrégation V, si V est concave alors aucune interaction positive n'est modélisée, si V est convexe alors aucune interaction négative n'est modélisée. Dans tous les cas une fonction d'agrégation V linéaire ne modélise aucune interaction positive ou négative.

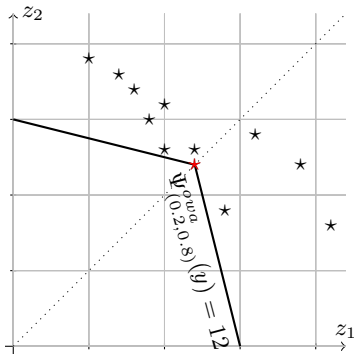

Figure 1.12 – Minimisation d'une fonction d'agrégation OWA $\Psi^{owa}_{(0.2,0.8)}$. Le trait en gras représente la courbe de niveau de la fonction owa. Les étoiles ($\star$) représentent les points des solutions du problème.

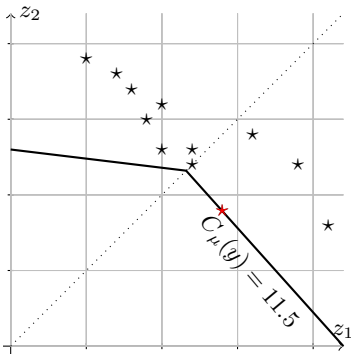

Figure 1.13 – Minimisation d'une intégrale de Choquet C_μ : $\mu(\{1\}) = 0.5$, $\mu(\{2\}) = 0.9$, $\mu(\{1,2\}) = 1$. Le trait en gras représente la courbe de niveau de l'intégrale de Choquet. Les étoiles ($\star$) représentent les points des solutions du problème.

1.2.5 Intégrale de Choquet

L'*intégrale de Choquet* est un outil mathématique défini par Choquet [Cho53]. Elle est utilisée dans le cadre de la logique floue [Sug74, Sug77, Gra96] et dans le cadre de l'aide à la décision multi-critère [Gra95, GL10]. Grabisch [Gra95] propose une extension de MAUT où l'intégrale de Choquet est utilisée comme fonction d'agrégation (cf. Figure 1.13, page ci-contre). L'intégrale de Choquet permet de modéliser la pondération, la complémentarité et la substituabilité de chaque ensemble de critères. Elle est définie à l'aide d'une *fonction de capacité* [Cho53] aussi appelée *mesure floue* [Sug74]. La fonction de capacité μ (cf. Définition 1.16, de la présente page) est une généralisation de l'idée de poids sur les critères utilisé dans la somme pondérée (λ_k, $k \in P$), telle qu'un poids est non seulement défini pour chaque critère mais aussi pour chaque ensemble de critères. Ainsi pour un ensemble de critères $A \subseteq P$, $\mu(A)$ donne le *poids* de l'ensemble des critères A. On définit l'ensemble des ensembles de critères par $\mathfrak{P} = \{A : A \subseteq P\}$, i.e. l'*ensemble des parties* de P.

Définition 1.16. Fonction de capacité [GL10].
Une fonction de capacité $\mu : \mathfrak{P}(P) \to [0,1]$ est une fonction d'ensemble satisfaisant :
(i) $\mu(\emptyset) = 0, \mu(P) = 1$ *(la fonction de capacité est normalisée) ;*
(ii) $\forall A, B \subseteq P, A \subset B \Longrightarrow \mu(A) \leq \mu(B)$ *(la fonction de capacité est croissante).*

Par souci de simplicité $\mu(k)$ désigne le poids du critère $k \in P$: $\mu(k) = \mu(\{k\})$. Pour un vecteur d'utilités $y \in \mathbb{R}^p_+$ l'intégrale de Choquet est définie comme la somme pondérée des composantes ordonnées du vecteur y. La pondération de chaque composante y_k du vecteur y est définie par rapport à l'ordre des composantes.

Définition 1.17. Intégrale de Choquet [GL10].
Soit μ une fonction de capacité définie sur P. L'intégrale de Choquet $C_\mu : \xi^p \to \xi \subseteq \mathbb{R}_+$, d'une alternative $y \in \mathbf{\Omega}$ est définie ainsi :

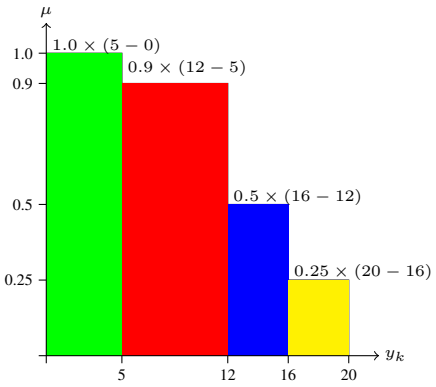

Figure 1.14 – Représentation d'une intégrale de Choquet C_μ. La surface en couleur est égale à la valeur de l'intégrale de Choquet pour le vecteur d'attributs $(5, 20, 12, 16)$ selon l'expression (1.7) : $C_\mu((5, 20, 12, 16)) = 1.0 \times (5 - 0) + 0.9 \times (12 - 5) + 0.5 \times (16 - 12) + 0.25 \times (20 - 16)$. Chaque couleur représente une partie de cette somme.

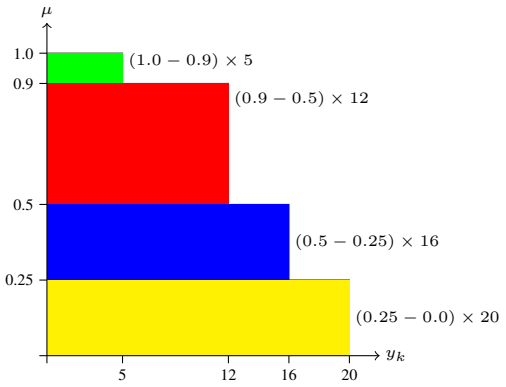

Figure 1.15 – Représentation d'une intégrale de Choquet C_μ. La surface en couleur est égale à la valeur de l'intégrale de Choquet pour le vecteur d'attributs $(5, 20, 12, 16)$ selon l'expression (1.6) : $C_\mu((5, 20, 12, 16)) = (1.0 - 0.9) \times 5 + (0.9 - 0.5) \times 12 + (0.5 - 0.25) \times 16 + (0.25 - 0.0) \times 20$. Chaque couleur représente une partie cette somme.

$$C_\mu(y) = \sum_{k=1}^{p} [\mu(Y_{(k)}) - \mu(Y_{(k+1)})] \times y_{(k)} \tag{1.6}$$

avec $(.)$ *est une fonction de permutation telle que* $y_{(1)} \leq \cdots \leq y_{(p)}$, $Y_{(k)} = \{(k), \ldots, (p)\}$ *et* $Y_{(p+1)} = \emptyset$. *Une autre expression de l'intégrale de Choquet est :*

$$C_\mu(y) = \sum_{k=1}^{p} \mu(Y_{(k)}) \times [y_{(k)} - y_{(k-1)}] \tag{1.7}$$

avec $y_{(0)} = 0$.

Les deux expressions de l'intégrale de Choquet, définies par les équations (1.6) et (1.7), sont illustrées par les figures 1.15 et 1.14 respectivement. Comme le montre ces figures, pour toute alternative $y \in \mathbf{\Omega}$ ces deux expressions sont strictement équivalentes.

L'intégrale de Choquet généralise la plupart des fonctions d'agrégation utilisées en MCDA [GP99], min,max,somme pondérée, OWA (cf. §1.1.2, page 14). Comme l'intégrale de Choquet peut modéliser une fonction max ou min alors elle peut être convexe ou concave (cf. Définition 1.15, page précédente) selon la fonction de capacité μ.

Exemple 1.3. Intégrale de Choquet et indépendance préférentielle.
Suite de l'exemple 1.2. Il existe au moins une intégrale de Choquet C_μ qui permet de modéliser fidèlement les préférences du directeur pédagogique. Cette intégrale de Choquet est définie par la fonction de capacité μ telle que : $\mu(\emptyset) = 0$, $\mu(\{1\}) = 0.5$, $\mu(\{2\}) = 0.5$, $\mu(\{3\}) = 0.2$, $\mu(\{1, 2\}) = 0.6$, $\mu(\{1, 3\}) = 0.8$, $\mu(\{2, 3\}) = 0.8$, $\mu(\{1, 2, 3\}) = 1$. Cette intégrale de Choquet modélise non seulement une pondération entre les objectifs comme la somme pondérée φ_λ mais aussi une substituabilité entre le critère de mathématiques et le critère de statistiques.

Le directeur préfère l'élève x^a à l'élève x^b alors $C_\mu((16,13,7)) < C_\mu((16,11,9))$, ce qui est vérifié par l'inéquation suivante : $C_\mu((16,13,7)) = 7 \times 0.4 + 13 \times 0.1 + 16 \times 0.5 = 12.1 < C_\mu((16,11,9)) = 9 \times 0.4 + 11 \times 0.1 + 16 \times 0.5 = 12.7$ Le directeur préfère l'élève x^c à l'élève x^d alors $C_\mu((6,11,9)) < C_\mu((6,13,7))$, ce qui est vérifié par l'inéquation suivante : $C_\mu((6,11,9)) = 9.4 < C_\mu((6,13,7)) = 9.8$ L'intégrale de Choquet C_μ modélise donc bien fidèlement les préférences du directeur contrairement à n'importe quelle somme pondérée φ_λ.

Définition 1.18. Cas particuliers de l'intégrale de Choquet [Gra96].

- *L'intégrale de Choquet C_μ définit une somme pondérée quand la fonction de capacité μ est* additive *:*

$$\forall A \subset P,\ \mu(A) = \textstyle\sum_{k \in A} \mu(k).$$

- *L'intégrale de Choquet C_μ définie une fonction* OWA *quand la fonction de capacité μ est* symétrique *:*

$$\forall A, B \in P,\ |A| = |B| \Leftrightarrow \mu(A) = \mu(B),$$

  de plus si $\mu(A) = 0, \forall A \subsetneq P$ alors C_μ définit une fonction min *et si $\mu(A) = 1, \forall A \subset P$ alors C_μ définit une fonction* max*.*

- *L'intégrale de Choquet C_μ est dite convexe quand la fonction de capacité μ est* submodulaire*. μ est submodulaire s.s.i. :*

$$\forall A, B \subset P,\ \mu(A \bigcup B) + \mu(A \bigcap B) \leq \mu(A) + \mu(B).$$

- *L'intégrale de Choquet C_μ est dite concave quand la fonction de capacité μ est* supermodulaire*. μ est supermodulaire s.s.i. :*

$$\forall A, B \subset P,\ \mu(A \bigcup B) + \mu(A \bigcap B) \geq \mu(A) + \mu(B).$$

  Il existe différents états de l'art sur l'intégrale de Choquet [GR00, GMSK00, GL05, GL10].

1.2.5.1 Autres représentations de l'intégrale de Choquet

En utilisant une fonction d'ensemble autre que la fonction de capacité il est possible de représenter différemment l'intégrale de Choquet [Gra97a, MG99]. La *fonction de Möbius* $m : \mathfrak{P} \to \mathbb{R}$ est une de ces fonctions d'ensemble.

Définition 1.19. Représentation de Möbius [Rot64, Gra97a].

- La transformation de Möbius $m : \mathfrak{P}(P) \to \mathbb{R}$ *de la fonction de capacité μ est une fonction d'ensemble définie par :*

$$m(A) = \textstyle\sum_{B \subseteq A} (-1)^{|A \setminus B|} \mu(B), \quad \forall A \subseteq P$$

  La transformation peut être inversée et μ peut être retrouvé depuis m par : $\mu(A) = \sum_{B \subseteq A} m(B), \quad \forall A \subset P$.

- *Une fonction d'ensemble $m : \mathfrak{P}(P) \to \mathbb{R}$ est la transformation de Möbius d'une fonction de capacité si et seulement si :*

(i) $m(\emptyset) = 0, \sum_{A \subseteq P} m(A) = 1$;

(ii) $\sum_{B \subseteq A \setminus \{k\}} m(B \bigcup \{k\}) \geq 0, \forall A \subseteq P, \forall k \in A$.

- *Soit* m *la transformation de Möbius d'une fonction de capacité* μ. $\forall y \in \xi^p$, *avec* $\xi \subseteq \mathbb{R}_+$, la représentation de Möbius C_m *de l'intégrale de Choquet* C_μ *est donnée par :*

$$C_m(y) = \sum_{A \subseteq B} m(A) \times \min_{k \in A} y_k$$

L'*indice de Shapley* [Sha53] défini l'importance qu'une fonction de capacité μ attribue à chaque critère. Ainsi l'importance d'un critère n'est pas seulement déterminée par son poids seul $\mu(k)$ mais aussi par le poids $\mu(A)$ de tous les ensembles de critères $A \subset P$ contenant k, $k \in A$.

Définition 1.20. Indice de Shapley [Sha53]
Soit μ *une fonction de capacité. Pour un critère* $k \in P$ l'indice de Shapley $\phi_k(\mu)$ *est défini par :*

$$\phi_k(\mu) = \sum_{A \subset P \setminus \{k\}} \frac{(p-|A|-1)!|A|!}{p!} . [\mu(A \bigcup \{k\}) - \mu(A)]$$

L'*indice d'interaction* généralise l'indice de Shapley au-delà d'un seul critère [Gra97a]. Grâce aux indices de Shapley et aux indices d'interaction il est possible de donner une autre représentation de l'intégrale de Choquet [Gra97a].

1.2.5.2 Construction d'un modèle de préférence basé sur l'intégrale de Choquet

À notre connaissance il existe deux logiciels : MYRIAD [LL05, GL10] et KAPPALAB [GKM08] pour construire, dans le cadre MCDA, une fonction de capacité μ et donc l'intégrale de Choquet C_μ correspondante. Pour construire la fonction de capacité, ces méthodes résolvent un problème d'optimisation avec des contraintes linéaires (telle le problème 1.5, page 29) défini à partir de l'information préférentielle du décideur. L'intégrale de Choquet est une fonction plus riche que la plupart des fonctions d'agrégation habituellement utilisées dans le cadre MAUT. Ainsi la construction d'une intégrale de Choquet à partir d'information préférentielle provoque moins d'incohérences (cf. §1.2.3.3, page 29) que les autres fonctions d'agrégation telle que la somme pondérée. Des incohérences peuvent néanmoins exister [LL06]. En contrepartie de la richesse de l'intégrale de Choquet, la définition précise d'une fonction de capacité demande plus d'information préférentielle de la part du décideur qu'une somme pondérée par exemple. Il est aussi plus difficile de construire les fonctions d'utilité partielles d'un modèle MAUT quand l'intégrale de Choquet est utilisée comme fonction d'agrégation [LG03], car on ne suppose pas l'indépendance préférentielle.

Pour finir notons l'existence d'intégrales de Choquet *bi-polaire* [GL02]. Ces intégrales de Choquet utilisent deux fonctions de capacité afin de modéliser plus fidèlement les préférences du décideur en différenciant le cas où le décideur trouve une alternative bonne et celui où il trouve une alternative mauvaise. Le décideur doit fournir plus d'informations préférentielles pour determiner une intégrale de Choquet qu'un intégrale de Choquet bipolaire [GL10]. Notons aussi l'existence des fonctions de capacité K-*additive* [Gra97b] où seules les interactions avec moins de k critères sont modélisées. La transformation de Möbius m de la fonction de capacité μ K-additive vérifie $\forall A \subset P$, $|A| > k \Longrightarrow m(A) = 0$. Les intégrales de Choquet basées sur des fonctions de capacité K-additives sont moins riches que l'intégrale de Choquet usuelle, mais permettent un compromis entre richesse, difficulté de construction et interprétation du modèle de préférence.

1.2.6 Paradigme de l'optimisation et aide à la décision multi-critère

Le cadre de nos travaux est l'optimisation où l'on considère que chaque attribut correspond à un objectif. Ainsi chaque attribut $k \in P$ (défini sur Ω_k) est naturellement ordonné de façon croissante ou décroissante par rapport aux préférences du décideur. Un attribut correspond donc à un objectif avec un sens d'optimisation : maximisation ou minimisation. On va par exemple naturellement chercher à minimiser un attribut de coût et à maximiser un attribut de qualité. Ainsi dans le cadre de la problématique du choix le problème de décision peut être considéré comme un problème d'optimisation (cf. §1.1.2.4, page 18).

On considère que pour chaque attribut $k \in P$ la fonction d'utilité partielle u_k est croissante si l'attribut k doit être minimisé et u_k est décroissante si l'attribut k doit être maximisé. Ainsi chaque critère $k \in P$ (i.e. l'association de l'attribut k et de u_k) doit être minimisé. De plus la fonction d'agrégation V est croissante (cf. Définition 1.9, page 18). Ainsi la fonction d'utilité U est aussi croissante et donc vérifie la dominance de Pareto. La fonction d'utilité U doit être minimisée.

1.3 Articulation entre un algorithme d'optimisation combinatoire multi-objectif et une méthode d'aide à la décision multi-critère

Le domaine de recherche de l'optimisation combinatoire multi-objectif (MOCO) et le domaine de recherche l'aide à la décision multi-critère (MCDA) ont été présentés dans les sections §1.2 et §1.1 respectivement. La problématique de cette thèse se situe à la conjonction de ces deux domaines. On considère un problème de décision multi-critère où l'ensemble des alternatives est défini en compréhension par un problème d'optimisation combinatoire multi-objectif. Alors une méthode de résolution MOCO doit est utilisée en association avec une méthode MCDA.

Afin d'obtenir un socle commun aux définitions MCDA et MOCO, nous utilisons des notations identiques pour ces deux domaines. Ainsi une solution correspond à une alternative, on emploie le terme de solution par la suite. L'ensemble des alternatives X est l'ensemble des solutions X. Sans perte de généralité l'ensemble $\mathbf{\Omega} = \Omega_1 \times \cdots \times \Omega_p$ est défini comme l'ensemble $\mathbb{R}^p$ (ou $\mathbb{R}^p_+$). On considère que le modèle de préférence est une fonction d'utilité respectant le paradigme de l'optimisation (cf. §1.2.6, page ci-contre).

Considérons que la phase de structuration multi-attribut du problème MCDA est déjà réalisée (cf. §1.2, page 22) : les attributs sont définis ainsi que le problème d'optimisation combinatoire multi-objectif définissant l'ensemble des alternatives en compréhension. Le problème de décision consiste à : (a) construire un modèle de préférence, (b) résoudre le problème d'optimisation combinatoire multi-objectif. L'ordre selon lequel se déroule ces deux phases permet de catégoriser les méthodes d'articulation entre un algorithme MOCO et une méthode MCDA en trois méthodologies différentes [Eva84]. Une méthode *a posteriori* détermine le modèle de préférence a posteriori de la résolution du problème MOCO. Une méthode *a priori* détermine le modèle de préférence a priori de la résolution du problème MOCO. Durant la résolution du problème MOCO le modèle de préférence est exploité afin construire le plus directement possible la solution préférée. Une méthode *interactive* détermine le modèle de préférence et résout le problème MOCO de manière simultanée. La figure 1.16 illustre ces trois différentes méthodologies qui sont présentées plus en détail dans la suite de cette section.

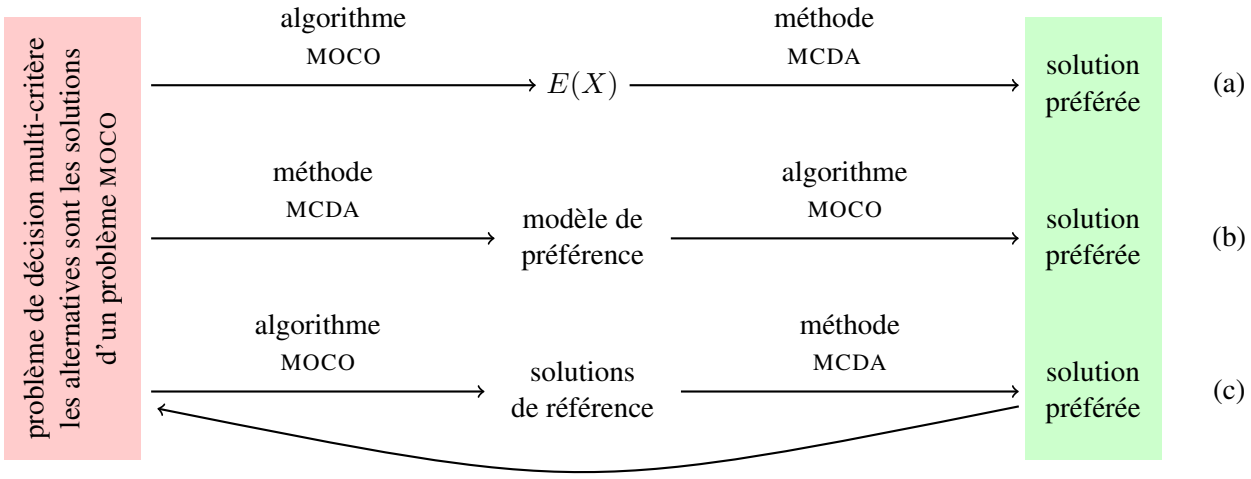

Figure 1.16 – Articulation d'un algorithme MOCO et d'une méthode d'aide à la décision multi-critère. Les trois méthodologies a posteriori (a), a priori (b) et interactive (c) sont représentées.

1.3.1 Méthodes a posteriori

La méthodologie a posteriori consiste à, dans un premier temps calculer un ensemble complet de solutions efficaces $E(X)$ (ou une approximation $\tilde{E}(X)$, cf. §1.1.3), puis dans un deuxième temps d'utiliser une méthode d'aide à la décision multi-critère pour sélectionner une solution préférée de $E(X)$. Cette méthode peut être basée sur une représentation graphique des solutions efficaces ou sur la construction d'un modèle de préférence sur l'ensemble $E(X)$. Un exemple d'application de la méthodologie a posteriori dans le cadre de la conception d'un avion est donné par Chiba et al [CMT11].

Dans la méthodologie a posteriori, n'importe quelle méthode MCDA peut être utilisée avec n'importe quel algorithme MOCO qui calcule un ensemble complet de solutions efficaces. Par exemple on peut générer les solutions efficaces de façon approché avec un algorithme à garantie de performance puis utiliser une méthode de surclassement pour sélectionner une solution préférée. Le modèle de préférence étant alors construit "sur mesure" pour l'instance du problème considéré. Un des avantages de la méthodologie a posteriori est que la méthode MCDA et l'algorithme MOCO peuvent être développés séparément.

1.3.2 Méthodes interactives

La méthodologie interactive propose de résoudre le problème d'optimisation (combinatoire) multi-objectif et le problème d'aide à la décision multi-critère de manière simultané. Une méthode interactive est présentée habituellement comme une procédure itérative en deux étapes [Van89] :

1. *l'étape de calcul* : considérant l'information préférentielle connue (ou un modèle de préférence partiellement déterminé), des solutions du problème sont construites. Ces solutions dites *de référence* peuvent être des solutions efficaces.
2. *l'étape de dialogue* : le décideur réagit aux solutions de référence, de l'information préférentielle est alors recueillie ce qui permet d'orienter la recherche vers des nouvelles solutions de référence et de faire progresser la détermination du modèle de préférence.

En général les méthodes interactives permettent avant tout au décideur d'explorer l'ensemble des solutions efficaces plutôt que de construire un modèle de préférence. Ainsi le modèle de préférence est généralement très incomplet même à la fin de la procédure. En revanche, ce modèle incomplet suffit à identifier rapidement une solution préférée de manière à minimiser le nombre d'itération et donc le temps nécessaire à l'élicitation des préférences. Pour cette raison, en général, le modèle de préférence ne peut pas être réutilisé pour d'autres instances du problème MOCO.

Les méthodes interactives ont suscitées un grand intérêt durant les années 70 et 80 [WDF+08] et connaissent à l'heure actuel un important regain d'intérêt pour leur utilisation en collaboration avec des métaheuristiques [BDMS08, PPK03, TMKM09, BGSZ09]. Parmi la littérature on note : la méthode STEM [BdMTL71] qui est une des premières méthodes interactives de la littérature ; une méthode interactive basée sur la méthode UTA (cf. §1.2.3.3, page 29) [JLMS87] ; des méthodes interactives basées sur la méthode GRIP [FGMS08, BGSZ09] ; un état de l'art [VV89] ; plusieurs méthodes interactives basées sur la norme de Tchebychev (cf. Définition 1.8, page 17) [Roy76, SC83, GG03]. Un exemple d'application de la méthodologie interactive pour choisir des contremesures après un accident nucléaire est donné par Perny et Vanderpooten [PV98].

Il existe dans la littérature plus d'une dizaine de méthodes interactives pour des problèmes de plus court chemin multi-objectif. Ces méthodes permettent, entre autre, de résoudre des problèmes de transport [CAA93], de planification [GV02] et de recherche de solutions de compromis [CRC90, GG03, GP06a].

1.3.3 Méthodes a priori

La méthodologie a priori consiste à, dans un premier temps déterminer un modèle de préférence à l'aide d'une méthode MCDA, puis dans un deuxième temps construire une solution préférée (ou optimale) selon ce modèle de préférence à l'aide d'une méthode de résolution MOCO.

Dans le cadre de la méthodologie a priori, le modèle de préférence est déterminé sans connaitre les solutions de l'instance du problème MOCO que l'on souhaite résoudre. Le modèle de préférence peut par exemple être déterminé avec de l'information préférentielle obtenue par rapport aux solutions d'une instance similaire de ce problème MOCO. Contrairement à une méthode interactive ou une méthode a posteriori, le modèle de préférence utilisé dans une méthode a priori doit nécessairement être défini de manière précise, car le décideur ne peut pas intervenir après la résolution du problème MOCO.

L'utilisation d'une méthodologie interactive ou a posteriori implique l'intervention d'un décideur à posteriori ou durant la résolution du problème MOCO considéré. Alors ces deux méthodologies ne peuvent s'appliquer qu'à des problèmes de décision suffisamment importants pour que le décideur puisse consacrer du temps à leurs résolutions. Si il est légitime de demander à un décideur de dépenser du temps pour exprimer ses préférences concernant la conception d'un avion [CMT11], il est moins légitime de lui demander de dépenser du temps pour exprimer ses préférences pour choisir le chemin emprunté par une voiture dans un réseau routier [SN10]. Soit une instance d'un problème MOCO pour lequel un modèle préférence est connu, une méthode a priori peut alors être appliquée pour résoudre cette instance sans l'intervention du décideur. On peut alors parler de décision automatique.

La plupart des méthodes MCDA peuvent être utilisées pour construire le modèle de préférence dans une méthode a priori : par exemple une méthode de surclassement [FLB+10] ou une méthode à critère unique de synthèse [Leh03]. L'algorithme MOCO utilisé doit permettre de construire une solution optimale selon le modèle de préférence. Pour cela l'algorithme MOCO doit être capable de prendre en compte le modèle de préférence durant la résolution du problème MOCO. Contrairement aux algorithmes MOCO pour le calcul de l'ensemble des solutions efficaces pour lesquels il existe une littérature abondante

depuis les années 70, ce n'est que récemment que de nombreux algorithmes MOCO prenant en compte un modèle de préférences ont été proposés dans la littérature. Ces algorithmes sont en général basés sur des algorithmes pour le calcul de l'ensemble des solutions efficaces (cf. Sections 1.1.4 et 1.1.3) auxquels des heuristiques, des règles de coupe ou des bornes ont été ajoutées afin de réduire les temps de calcul. Il peut s'agir d'algorithmes exacts ou d'algorithmes approchés. 1.1.4

Concernant les méthodes approchées on cite les travaux de Branke et Deb [BD04] qui proposent de prendre en compte une pondération des critères dans un algorithme génétique. Fernandez et al [FLB+10] proposent d'utiliser un modèle de préférence déterminé à l'aide d'une méthode de surclassement dans un algorithme génétique. Saad et al [SHBB08] proposent d'agréger les objectifs avec une intégrale de Choquet dans un algorithme génétique.

Concernant les méthodes exactes, différents types d'algorithmes pour les problèmes MOCO ont été adaptés pour prendre en compte un modèle de préférence. Lehuédé et al [Leh03, LGLS06a, LGLS06b] proposent d'optimiser une fonction d'utilité MAUT composée de fonctions d'utilité partielles linéaires par morceaux et d'une intégrale de Choquet dans un algorithme de programmation par contraintes. Junker [Jun04] propose de prendre en compte un préordre dans un algorithme de programmation par contraintes. Perny et Spanjaard [PS05] proposent des algorithmes pour prendre en compte un préordre quelconque dans un problème d'arbre couvrant multi-objectif. Pour ce même problème Galand et al [GPS10] proposent un algorithme de séparation et évaluation (B&B) minimisant une fonction d'utilité composée d'une intégrale de Choquet concave et d'une fonction d'utilité partielle convexe (la même fonction d'utilité partielle est appliquée à tous les critères). Galand et al [GPS09] proposent pour le problème de sac à dos multi-objectif un algorithme de séparation et évaluation maximisant une intégrale de Choquet convexe. Dans la littérature il existe aussi plusieurs algorithmes d'étiquetage qui exploitent un modèle de préférence pour le problème de plus court chemin multi-objectif. Ces algorithmes sont présentés en détail dans la section 2.4.1.

1.4 Problématique

Nous avons présenté dans ce chapitre les domaines de l'aide à la décision multi-critère (MCDA) et de l'optimisation combinatoire multi-objectif (MOCO), ainsi que les méthodologies pour articuler un algorithme MOCO avec une méthode d'aide à la décision multi-critère.

La méthodologie a priori, dans le cadre de la décision automatique, constitue une approche intéressante pour la résolution de problèmes multi-objectif pour lesquels il n'est pas possible de faire intervenir le décideur pour la résolution de chacune des instances du problème.

La mise en place d'une méthode a priori pour un problème dans le monde réel est liée à deux problématiques :

(1) la définition d'un modèle de préférence adapté à ce problème,

(2) la proposition d'un algorithme capable, en un temps raisonnable, de construire une solution optimale selon ce modèle de préférence.

La définition d'un modèle de préférence (1) ne fait pas partie de la portée de cette thèse. Pour cette problématique de nombreuses méthodes proposées dans la littérature MCDA pour la construction d'un modèle de préférences (cf. §1.2.2.3, page 25) peuvent être utilisées. La proposition d'algorithmes d'efficaces pour optimiser un modèle de préférence (2) est la problématique qui nous intéresse dans cette thèse. Notons que les points (1) et (2) sont parfois contradictoires. Par exemple si le modèle de préférence est une somme pondérée alors il est possible de scalariser (cf. §1.1.2.2, page 15) le problème MOCO. Alors un algorithme mono-objectif peut être utilisé pour résoudre la scalarisation de ce problème en un temps

raisonnable, du point de vue du (2) cet algorithme est très satisfaisant. Par contre une somme pondérée ne permet pas en général de modéliser les préférences du décideur de manière satisfaisante (cf. Exemples 1.1 et 1.2), ce qui du point de vue du (1) n'est pas satisfaisant. Afin de disposer d'un modèle de préférence, qui puisse s'adapter pour de nombreux problèmes du monde réel, on s'intéresse à l'optimisation d'un modèle de préférence complexe. Par conséquent on suppose que le modèle de préférence ne permet pas de réduire le problème MOCO en un problème mono-objectif équivalent.

Quand aucune méthode efficace n'est connue pour prendre en compte le modèle de préférence directement dans la résolution du problème MOCO, des algorithmes MOCO construisant l'ensemble des solutions efficaces (cf. §1.2, page 13) peuvent être utilisés. À la suite de la construction de l'ensemble des solutions efficaces une solution préférée selon le modèle de préférence est sélectionnée dans cet ensemble. Par exemple, dans le cadre du routage dans les réseaux, l'algorithme RMC demande dans un premier temps au décideur de définir des poids sur chacun des critères. Ces poids permettent de déterminer une norme de Tchebychev (cf. Définition 1.8, page 17) qui constitue alors le modèle de préférence. Ensuite l'ensemble maximal complet des solutions efficaces est construit par un algorithme d'étiquetage, puis la solution optimale selon cette norme de Tchebychev est sélectionnée dans cet ensemble de solutions. L'algorithme RMC est détaillé dans la sous-section 4.6.3.2.

Le nombre de solutions efficaces pouvant être exponentiel [Ser86], les temps de calcul pour un algorithme construisant l'ensemble maximal complet des solutions efficaces peut être important (ce qui du point de vue du (2) n'est pas satisfaisant). Pour cette raison il est nécessaire autant que possible de prendre en compte le modèle de préférences directement dans l'algorithme MOCO afin d'éviter le calcul de l'ensemble des solutions efficaces et ainsi réduire les temps de calcul. Un des challenges actuels de l'optimisation combinatoire multi-objectif [WDF+08] est de proposer ce type d'algorithmes qui permettent d'optimiser directement un modèle de préférence. En effet relativement peu de travaux ont été réalisés sur ce type d'algorithmes contrairement aux algorithmes permettant la construction d'un ensemble de solutions efficaces [EG00]. Des algorithmes efficaces pour optimiser un modèle de préférence (i.e. avec des temps de calcul proches de ceux d'algorithmes mono-objectif équivalents) permettrait (dans le cadre de la décision automatique) de considérer l'approche multi-critère pour des problèmes pour lesquels actuellement uniquement l'approche mono-critère est considérée.

Le routage dans les réseaux internet (cf. §4.6, page 117) constitue un domaine d'application pour les méthodes a priori. Pour router les paquets de données dans un réseau, chaque routeur résout des problèmes de plus courts chemins. En général uniquement un objectif de coût est pris en compte pour calculer ces chemins sur lesquels sont ensuite routés les paquets. Néanmoins pour le calcul de ces chemins il peut être intéressant de prendre en compte aussi des objectifs de qualité de service, par exemple la bande passante maximale du chemin ou le délai de transmission du chemin. On parle alors de routage multi-objectif. La prise en compte de plusieurs objectifs nécessite alors la définition d'un modèle de préférence permettant de différencier les solutions efficaces entres elles. Dans le cadre du problème de routage dans un réseau le décideur est l'administrateur de ce réseau. Comme chaque routeur construit et réactualise régulièrement ses plus courts chemins, il semble difficile de demander à l'administrateur du réseau de participer de manière interactive ou a posteriori à la résolution de chaque problème de plus court chemin. La méthodologie a priori semble alors être la plus appropriée pour résoudre ce type de problèmes. Le routage par contraintes (cf. §4.6.3, page 121) constitue un exemple de méthodologie a priori appliquée au routage (multi-objectif) dans les réseaux. Des contraintes sur les objectifs sont déterminées par l'administrateur, elles sont définies par des bornes inférieures ou supérieures sur les objectifs. Ces contraintes sont équivalentes à un modèle de préférence : une solutions satisfaisant l'ensemble des contraintes est nécessairement préférée à une solution ne satisfaisant pas toutes les contraintes. Ainsi les solutions préférées du problème sont celles qui satisfont toutes ces contraintes. De nombreux algorithmes prenant en compte

directement ces contraintes durant la résolution de problème de plus court chemin multi-objectif ont été proposés dans la littérature [NM99, MNK01].

Les travaux de cette thèse se situent dans le cadre d'une méthodologie a priori basée sur un modèle de préférences défini comme une fonction d'utilité MAUT composée d'une intégrale de Choquet et de fonctions d'utilité partielles linéaires par morceaux. Dans le chapitre 4.5 nous étudions l'application de cette méthodologie dans le cadre du routage dans les réseaux internet. Cette fonction d'utilité est particulièrement appropriée pour la représentation des préférences multi-critère. En effet cette fonction d'utilité permet de modéliser la grande majorité des fonctions d'utilité utilisées en MCDA comme la norme de Tchebychev ou les fonctions d'utilité construites par les méthodes UTA et MACBETH (cf. §1.2.3, page 26). Elle permet de modéliser une commensurabilité entres les critères et les trois comportements usuels pour agréger les préférences : pondération, substituabilité et complémentarité entre les critères (cf. §1.2.4, page 30). Cette fonction d'utilité permet aussi de modéliser des contraintes sur les objectifs ou de modéliser une préférence pour des solutions de compromis.

La problématique qui nous intéresse dans cette thèse est de proposer des algorithmes efficaces pour l'optimisation de cette fonction d'utilité dans des problèmes de plus courts chemins multi-objectifs. Compte tenu des possibilités de modélisation d'une telle fonction d'utilité, ces algorithmes peuvent être utilisés non seulement dans le cadre du routage dans les réseaux internet, mais aussi par exemple pour le calcul de plus courts chemins multi-objectifs dans des réseaux routiers.

Il existe de nombreuses variantes du problème de plus courts chemins multi-objectifs, dans cette thèse différentes méthodes sont proposées pour ces différentes variantes. Parmi les différentes variantes de ce problème on cite le problème de plus court chemin d'un nœud du graphe à un autre nœud du graphe et le problème de plus courts chemins d'un nœud du graphe à tous les autres nœuds du graphe. Ces deux variantes correspondent à des problématiques de routage différentes (cf. §4.7.2, page 133). Dans le chapitre 3 on propose des méthodes pour prendre en compte cette fonction d'utilité dans un algorithme d'étiquetage pour le problème de plus court chemin d'un nœud à un nœud avec des objectifs additifs. Dans le chapitre 4 on propose des méthodes pour prendre en compte cette fonction d'utilité dans un algorithme d'étiquetage pour le problème de plus courts chemins d'un nœud à tous les nœuds avec des objectifs additifs et bottleneck.

CHAPITRE 2

Le problème de plus court chemin multi-objectif

Le problème de plus court chemin multi-objectif (MOSP) est un des problèmes d'optimisation combinatoire multi-objectif les plus étudiés [EG00]. Le problème MOSP consiste à calculer un (ou plusieurs) chemin efficace dans un graphe modélisant par exemple un réseau de télécommunications. Ce problème est particulièrement étudié dans le cadre du transport urbain [RE09, SN10, RP11] et du routage réseau [NM99, MNK01]. Ainsi le routage de paquet de données, dans les réseaux de télécommunications, prenant en compte plusieurs métriques (objectifs), nécessite la résolution de problèmes MOSP (cf. Chapitre 4.5, page 117).

De nombreux méthodes exactes existent pour le problème MOSP certaines de ces méthodes sont efficaces (en temps de calcul et en espace mémoire) même pour les instances de grandes tailles [SN10]. Deux algorithmes à garantie de performance ont été proposés dans la littérature, un algorithme [War87] ne permet de résoudre que des instances particulières (graphes acycliques), un autre [TZ09] est plus lent à exécuter que les algorithmes exacts, voir [Gra10]. Des algorithmes génétiques sont proposés, pour certains [STW04, HQF07] les temps de calcul ne sont pas comparés avec des algorithmes exacts, pour un autre [MPJ07] les temps de calcul se révèlent plus importants que pour un algorithme exact. Pour finir un algorithme de colonie de fourmi [GN10] est proposé, cet algorithme ne fonctionne que sur des instances particulières (graphes acycliques). Pour Müller et Weihe [MHW06] il n'est pas nécessaire d'utiliser des approximations afin de résoudre le problème MOSP en un temps raisonnable. En effet, ces auteurs montrent que pour un grand nombre d'instances, le problème MOSP (bi-objectif) n'est pas *intractable* [MHW06]. Pour toutes ces raisons nous ne considérons que la résolution exacte du problème MOSP.

Ce chapitre propose un état de l'art à la fois exhaustif et détaillé de la littérature MOSP, ce qui n'est pas le cas des précédents états de l'art de la littérature [CM86, CM93, HPS96, Skr00, EG02, Tar07, CP10]. Ce chapitre s'attarde particulièrement sur les algorithmes d'étiquetage qui représentent une majorité des algorithmes pour le problème MOSP proposés dans la littérature. Ces algorithmes sont une généralisation au cas multi-objectif d'algorithmes usuels de résolution de problème de plus court chemin mono-objectif : algorithmes de Dijkstra, de Bellman et A*. Nous proposons un algorithme d'étiquetage (cf. Algorithme 2.1, page 51) générique pour le problème MOSP, ensuite chaque algorithme d'étiquetage de la littérature est défini comme une variante de cet algorithme générique.

La section 2.1 définit le problème de plus court chemin multi-objectif. La section 2.2 présente les algorithmes d'étiquetage permettant de construire un ensemble complet de solutions efficaces. La section 2.3 présente les méthodes spécifiques au multi-objectif (cf. §1.1.4, page 20) pour construire un ensemble

complet de solutions efficaces. La section 2.4 présente les algorithmes capables d'exploiter un modèle de préférence pour construire directement une solution préférée. Ces algorithmes peuvent être utilisés dans le cadre de méthodes a priori.

2.1 Le problème de plus court chemin

Le problème de plus court chemin consiste à construire un ou plusieurs plus courts (ou meilleurs) chemins dans un graphe. Les graphes sont utilisés dans la littérature pour modéliser des réseaux de nature diverse :

- des réseaux de transport [SNB11], des notions de précédence ou des contraintes en RO de transport.
- des réseaux de télécommunication [MNK01], une application des travaux réalisés pendant cette thèse est proposée dans le cadre du routage dans les réseaux internet (cf. Chapitre 4.5, page 117).
- des réseaux génétiques, en *bio-informatique*.

Dans tous les domaines mentionnés ci-dessus, il existe de nombreuses applications utilisant des algorithmes pour résoudre divers problèmes de plus court chemin.

Un graphe est défini comme un ensemble de nœuds, et un ensemble d'arcs entre certaines paires nœuds. Dans le cadre du problème de plus court chemin, à chaque arc est associé un *coût*, parfois une *performance*.

Définition 2.1. Graphe [PRS07].
Un graphe $G = (N, A, c)$ *est défini tel que :*

- $N = \{1, \ldots, n\}$ *est l'ensemble des* nœuds, $n = |N|$ *le nombre de nœuds.*
- $A \subset N \times N$ *est l'ensemble des* arcs *orientés. Chaque arc représente un lien entre deux nœuds.*
- $c : A \to \mathbb{R}^p$ *est une fonction qui associe à chaque arc un vecteur de* p *valeurs réelles positives (*coûts*).*

Le graphe ne possède pas d'arc multiple, i.e. il n'existe pas deux arcs différents reliant la même paire de nœuds. Ainsi un arc a *reliant deux nœuds* i *et* j *est défini tel que* $a = (i, j)$.

Par mesure de simplicité on utilise la notation $c(i, j)$ à la place de la notation $c((i, j))$, pour l'arc entre des nœuds $i, j \in N$. Les graphes non orientés peuvent être considérés comme un sous-ensemble des graphes orientés, où pour chaque arc $(i, j) \in A$ il existe un arc $(j, i) \in A$ avec $c(i, j) = c(j, i)$. Sans perte de généralité nous considérons que les graphes sont orientés. Pour chaque nœud $i \in N$, $\Gamma^+(i) = \{j : (i, j) \in A\}$ désigne son ensemble de nœuds *successeurs* et $\Gamma^-(i) = \{j : (j, i) \in A\}$ désigne son ensemble de nœuds *prédécesseurs*. La *densité* d'un graphe est définie par le rapport entre le nombre d'arc et le nombre de nœuds $\frac{|A|}{|N|}$. Un graphe est *complet* s.s.i. $\frac{|A|}{|N|} = |N|$. On considère que le graphe ne possède pas d'arc multiple, ainsi chaque chemin est uniquement défini par sa séquence de nœuds (cf. Définition 2.2, de la présente page).

Définition 2.2. Chemin [PRS07].

- *Un* chemin *(ou route)* $r = \langle v_1, v_2, \ldots, v_{t-1}, v_t \rangle$ *est une séquence alternative de* t *nœuds* $(v_1, \ldots, v_t \in N)$ *et de* $t - 1$ *arcs* $((v_1, v_2), \ldots, (v_{t-1}, v_t) \in A)$.
- *Un* sous-chemin r' *de* r *est une sous-séquence de* r *telle que* $r' = \langle v_l, v_{l+1}, \ldots, v_d \rangle$, *avec* $1 \leq l \leq d \leq t$.
- *Le* chemin nul $r = \langle v_1 \rangle$ *contient un unique sommet* $(v_1 \in N)$ *et aucun arc.*

- *Un chemin* $r = \langle v_1, \ldots, v_t \rangle$ *est un* cycle *si et seulement si* $v_1 = v_t$.
- *Soit* $\Diamond$ *l'*opérateur binaire *d'agrégation des chemins tel que :*

$$\langle v_1, \ldots, v_t \rangle \Diamond \langle v_{t+1}, \ldots, v_l \rangle = \langle v_1, \ldots, v_t, v_{t+1}, \ldots, v_l \rangle$$

Un graphe est dit *acyclique* si il n'existe aucun cycle dans le graphe. Habituellement on considère qu'un algorithme de plus court chemin calcule des chemins entre un nœud source et un nœud puit, néanmoins de nombreux algorithmes de plus court chemin permettent en réalité de calculer un chemin optimal entre un nœud source et chaque nœud du graphe. Afin de catégoriser ces différents algorithmes et de mettre en avant leurs différences, nous définissons plusieurs ensembles de chemins.

Définition 2.3. Ensembles de chemins [GBR06].
Soit un graphe $G = (N, A, c)$ *et* $i, j \in N$ *deux nœuds, on définit les ensembles de chemins suivants :*

- $R_{i,j}$ *est l'ensemble des chemins du nœud* i *au nœud* j.
- $R_{i,\bullet} = \bigcup_{v_2 \in N} R_{i,v_2}$ *est l'ensemble des chemins du nœud* i *à n'importe quel nœud* $v_2 \in N$.
- $R_{\bullet,j} = \bigcup_{v_1 \in N} R_{v_1,j}$ *est l'ensemble des chemins de n'importe quel nœud* $v_1 \in N$ *au nœud* j.
- $R_{\bullet,\bullet} = \bigcup_{v_1 \in N} R_{v_1,\bullet} = \bigcup_{v_2 \in N} R_{\bullet,v_2}$ *est l'ensemble des chemins de n'importe quel nœud* $v_1 \in N$ *à n'importe quel nœud* $v_2 \in N$.

2.1.1 Le problème de plus court chemin mono-objectif

Pour un graphe $G = (N, A, c)$, considérant un nœud source $s \in N$ et un nœud puits $t \in N$, la résolution du problème de plus court chemin ("*shortest path*", SP) est un chemin de coût (ou de longueur) minimal entre une source et un nœud puits. Le coût d'un chemin est défini par la fonction objectif z qui agrège les coûts des arcs composant le chemin. La fonction objectif z est associée avec un *opérateur binaire* $\otimes : \mathbb{R}_+^2 \to \mathbb{R}_+$ (voir aussi [GM08]), tel que le coût du chemin $r = \langle v_1, \ldots, v_{l+1} \rangle$ est défini par $z(r) = c(v_1, v_2) \otimes c(v_2, v_3) \otimes \cdots \otimes c(v_l, v_{l+1}) = \otimes_{a \in r} c(a)$. L'opérateur binaire est associatif ($(y^a \otimes y^b) \otimes y^c = y^a \otimes (y^b \otimes y^c)$; $y^a, y^b \in \mathbb{R}_+$), commutatif ($y^a \otimes y^b = y^b \otimes y^a$; $y^a, y^b \in \mathbb{R}_+$) et possède un élément idempotent ($\exists y^a \in \mathbb{R}_+$ t.q. $y^a \otimes y^a = y^a$). Cette élément idempotent est 0 si l'opérateur binaire $\otimes_k = +$ est une addition et 1 si l'opérateur binaire $\otimes_k = \times$ est une multiplication. Ainsi le coût d'un chemin ne dépend pas de l'ordre des arcs dans la séquence définissant le chemin. Les fonctions objectif peuvent être une :

- somme : $z(r) = \sum_{a \in r} c(a)$, alors l'opérateur binaire est l'addition $\otimes = +$;
- maximum : $z(r) = \max_{a \in r} c(a)$, alors l'opérateur binaire est le maximum $\otimes = \max$;
- minimum : $z(r) = \min_{a \in r} c(a)$, alors l'opérateur binaire est le minimum $\otimes = \min$;
- multiplication : $z(r) = \prod_{a \in r} c(a)$, alors l'opérateur binaire est la multiplication $\otimes = \times$.

Notons qu'une fonction objectif minimale doit être en générale maximisée contrairement aux autres fonctions objectif. Un problème SP avec comme objectif la maximisation d'un opérateur minimum peut facilement être transformé en un problème équivalent où l'objectif est de minimiser un opérateur max [Han80]. Ainsi ces deux objectifs sont équivalents, ils sont dits "*bottleneck*".

2.1.1.1 Différents problèmes de plus court chemin

La résolution d'un problème de plus court chemin peut désigner un chemin ou un ensemble de chemins. Ainsi plutôt que de calculer un plus court chemin entre deux nœuds du graphe, le décideur peut aussi vouloir calculer un plus court chemin entre un nœud et tous les autres nœuds. C'est pourquoi on associe un problème SP distinct à chaque ensemble de chemins $R_{s,t}$, $R_{s,\bullet}$, $R_{\bullet,\bullet}$:

- le problème SP $R_{s,t}$ de plus court chemin d'un nœud source s à un nœud puits t, la résolution de ce problème est un plus court chemin de s à t, une solution de ce problème est un chemin $r \in R_{s,t}$;
- le problème SP $R_{s,\bullet}$ de plus court chemin d'un nœud source s à n'importe quel nœud $j \in N$, la résolution de ce problème est un ensemble de chemins contenant, pour tout nœud $j \in N$, un plus court chemin allant de s à j, une solution est un chemin $r \in R_{s,\bullet}$;
- le problème SP $R_{\bullet,\bullet}$ de plus court chemin de n'importe quel nœud $i \in N$ à n'importe quel nœud $j \in N$, la résolution de ce problème est un ensemble des chemins contenant, pour toute paire de nœuds $i, j \in N$, un plus court chemin allant de i à j, une solution est un chemin $r \in R_{\bullet,\bullet}$.

Pour chacun de ces problèmes il existe des algorithmes d'une complexité polynomiale. Pour le problème $R_{s,t}$ on peut utiliser l'algorithme A* [HNR68]. Pour le problème de plus court chemin $R_{s,\bullet}$, on peut utiliser l'algorithme de Dijkstra [Dij59] ou l'algorithme de Bellman [Bel58]. Pour le problème $R_{\bullet,\bullet}$ on peut utiliser l'algorithme de Roy-Floyd-Warshall [Roy59]. Ces trois algorithmes minimisent une fonction objectif additive, Gondran et Minoux [GM08] proposent des généralisations de ces algorithmes quand l'objectif est défini sur un semi-anneau (la fonction objectif est additive, multiplicative, maximum, minimum ou autre). Le problème $R_{\bullet,\bullet}$ n'est pas considéré dans cette thèse. On note que le problème $R_{\bullet,t}$ peut être résolu facilement en inversant le sens des arcs puis en utilisant un algorithme pour le problème $R_{s,\bullet}$. Dans la pratique une telle modification du graphe ne nécessite pas de temps de calcul supplémentaire.

2.1.1.2 Algorithmes d'étiquetage

Les algorithmes de Dijkstra, de Bellman et A* sont considérés dans la littérature comme des algorithmes d'étiquetage (ou "*labeling*") [Ber93]. Les algorithmes d'étiquetage sont des algorithmes de programmation dynamique qui construisent le(s) chemins optimaux en associant à chaque chemin $r \in R_{s,i}$ (avec $i \in N$ un nœud quelconque) un label (étiquette) ℓ. On dit que le label ℓ est associé avec le nœud i. L'ensemble des labels représente une structure de donnée efficace pour stocker l'ensemble des chemins $R_{s,\bullet}$. Les algorithmes d'étiquetage sont parmi les plus efficaces pour résoudre le problème de plus court chemin [DP84]. Par mesure de simplicité on considère que la fonction objectif est redéfinie pour les labels telle que $z(\ell) = z(r)$. Ces algorithmes suivent le schéma suivant [GP86, Ber93] : durant l'étape d'*initialisation* un label ℓ^a associé au nœud source s, modélisant le chemin $\langle s \rangle$ et avec un coût nul est construit puis ajouté à la liste de labels à propager $\mathfrak{L} = \{\ell^a\}$. Ensuite durant l'étape de *sélection* un label associé à un nœud $i \in N$ est sélectionné et extrait de $\mathfrak{L}$. Durant l'étape de *propagation* ce label est propagé vers chaque nœud adjacent $j \in \Gamma^+(i)$. Ainsi un nouveau label est créé à chaque nœud adjacent, si ce label a un coût moindre que le label déjà associé au nœud adjacent alors ce dernier est supprimé et le nouveau label est ajouté à $\mathfrak{L}$. Dans le cas contraire, si le nouveau label a un coût supérieur au label déjà associé au nœud adjacent, alors il est supprimé. Si ce nœud adjacent n'a jamais été associé à un label alors le nouveau label est directement ajouté à $\mathfrak{L}$. L'étape de propagation est répétée jusqu'à ce que l'ensemble $\mathfrak{L}$ soit vide. Notons que dans le cadre mono-objectif, à chaque itération de l'étape de propagation, chaque nœud est associé avec au plus un label, ce qui n'est plus vrai dans le cadre multi-objectif. La plupart des algorithmes de plus court chemin proposés dans la littérature sont des algorithmes d'étiquetage

[Ber93]. Ainsi la plupart des réseaux informatique reposent sur l'utilisation d'algorithmes d'étiquetage (cf. Chapitre 4.5, page 117).

Les différents algorithmes d'étiquetage se distinguent par la structure de données utilisée pour stocker les labels ainsi que la stratégie employée pour sélectionner un label à propager à chaque itération. Parmi les algorithmes d'étiquetage, on différencie habituellement deux sous-familles d'algorithmes : les algorithmes dits de "*label setting*" (LS) proposent de sélectionner le label de coût minimal à chaque itération [Ber93]. Supposons que le label sélectionné soit associé à un nœud $i \in N$, alors le chemin associé au label sélectionné est un plus court chemin de s à i. En effet les labels qui seront propagés à partir des labels non encore propagé, auront un coût nécessairement plus élevé que leur label géniteur (car on considère que le coût des arcs sont positifs, $c(a) \in \mathbb{R}_+, \forall a \in A$) et donc que les labels sélectionnés précédemment. Ceci implique également qu'au maximum un label par nœud est propagé durant l'exécution de l'algorithme. Les algorithmes de LS se distinguent les uns des autres par la structure de données utilisée pour stocker les labels de $\mathfrak{L}$. Les algorithmes qui ne peuvent pas être considérés comme des algorithmes de label-setting sont dit de "*label correcting*" (LC) [Ber93]. En effet les labels qui sont sélectionné dans les algorithmes LC ne sont pas assurés d'être des plus courts chemins, ils doivent donc parfois être supprimés (ou corrigés) plus tard dans l'exécution de l'algorithme. Les algorithmes de LC se différencient par la stratégie employée pour sélectionner un label dans $\mathfrak{L}$ à chaque itération. L'algorithme de Dijkstra est un algorithme de LS, l'algorithme de Bellman (une variante de la méthode originale [Bel58]) est un algorithme de LC. Pour une étude complète des différents algorithmes d'étiquetage mono-objectif on réfère à [GP86, Ber93]

Concernant l'algorithme A*, il utilise une fonction $q : N \to \mathbb{R}_+$ qui donne une borne inférieure pour chaque sommet $i \in N$ de coût d'un chemin $R_{i,t}$. L'*évaluation* d'un label est l'addition du coût du chemin associé au label avec la borne inférieure du sommet associé au label. À chaque itération, le label qui a l'évaluation minimale est sélectionné. Si la fonction q respecte certaines conditions alors au maximum un label par nœud est propagé durant l'exécution de l'algorithme, alors l'algorithme est dit de LS. L'algorithme A* est plus performant que l'algorithme de Dijkstra ou l'algorithme de Bellman néanmoins, cet algorithme ne permet de résoudre qu'un sous-problème $R_{s,t}, t \in N$ du problème plus général $R_{s,\bullet}$ résolu par les deux autres algorithmes.

2.1.1.3 Modélisation sous la forme de problèmes d'optimisation en variables entières

Les problèmes de plus court chemin peuvent être modélisés sous la forme de problèmes d'optimisation en variables entières avec des contraintes linéaires. Ainsi le problème $R_{s,t}$ peut se modéliser de la manière suivante [MMO91] :

$$
\begin{array}{lllll}
\text{opt.} & z(x) & & & \\
\text{s.c.} & \sum_{j\in\Gamma^+(s)} x_{s,j} - \sum_{j\in\Gamma^-(s)} x_{j,s} & = 1 & & (a) \\
 & \sum_{j\in\Gamma^+(i)} x_{i,j} - \sum_{j\in\Gamma^-(i)} x_{j,i} & = 0 & \forall i \in N \setminus \{s,t\} & (b) \\
 & \sum_{j\in\Gamma^+(t)} x_{t,j} - \sum_{j\in\Gamma^-(t)} x_{j,t} & = -1 & & (c) \\
 & x_{i,j} \in \{0,1\} & & \forall (i,j) \in A &
\end{array}
\tag{2.1}
$$

Où $\forall i, j \in N$, $x_{i,j} = 1$ si l'arc (i,j) fait partie de la solution (chemin) et $x_{i,j} = 0$ sinon. La contrainte (a) implique que les chemins contiennent un arc sortant du nœud source. Les contraintes (b) impliquent que les chemins contiennent, soit un arc entrant et un arc sortant, soit aucun arc sortant et aucun arc entrant, pour chaque nœud (ni source, ni puits) du graphe. La contrainte (c) implique que les chemins contiennent un arc entrant du nœud puits. Le problème de plus court chemin $R_{s,\bullet}$ peut aussi être modélisé

sous la forme d'un problème d'optimisation en variables entières [MMO91] :

$$\begin{array}{lllll} \text{opt.} & z(x) & & & \\ \text{s.c.} & \sum_{j\in\Gamma^+(s)} x_{s,j} - \sum_{j\in\Gamma^-(s)} x_{j,s} & = n-1 & & (a) \\ & \sum_{j\in\Gamma^+(i)} x_{i,j} - \sum_{j\in\Gamma^-(i)} x_{j,i} & = -1 & \forall i \in N \setminus \{s\} & (b) \\ & x_{i,j} \in \mathbb{N}_+ & & \forall (i,j) \in A & \end{array} \tag{2.2}$$

Ou $\forall i, j \in N$, $x_{i,j}$ indique le nombre de solutions (chemins) qui contiennent l'arc (i, j). Le problème 2.2 calcule un chemin optimal $(R_{s,i})$ pour chaque nœud $i \in \mathbb{N} \setminus \{s\}$. La contrainte (a) implique que $n-1$ chemins optimaux contiennent un arc adjacent à s, i.e. $n-1$ chemins "commencent" au nœud s. La contrainte (b) implique qu'un chemin se "termine" à chaque nœud $i \in N \setminus \{s\}$. La résolution des problèmes linéaires (2.2) et (2.1) avec un algorithme de simplex n'est pas efficace par rapport à la résolution de problèmes MOSP équivalents avec un algorithme d'étiquetage [RE09]. Néanmoins, des résultats très récents sur le sujet [GLP11] montrent que pour l'optimisation de fonctions d'utilité convexes il pourrait être intéressant d'utiliser des algorithme de résolution de programme linéaire pour résoudre un problème de plus court chemin.

2.1.1.4 Variantes du problème de plus court chemin

Il existe de nombreuses variantes du problème de plus court chemin mono-objectif, tel que le problème de plus court chemin multi-modal [Gra10], le problème de plus court chemin avec contraints [KKKM04] (cf. §4.6.3, page 121), le problème de plus court chemin stochastique [BT91], le problème de ranking [Epp94, MPS99].

2.1.2 Le problème de plus court chemin multi-objectif (MOSP)

On trouve dans la littérature de nombreuses applications au problème MOSP : dans le domaine du transport [CM86, CM93, MHS07, Gra10, RP11, SNB11], dans le domaine des télécommunications [Beu06, NM99, MNK01, CCP06] (cf. Chapitre 4.5, page 117), en planification de satellites [GV02, RV03], dans le domaine militaire [HHW01] ou pour modéliser d'autres problèmes MOCO [CCF+03].

2.1.2.1 Les notations multi-objectif

Le passage au multi-objectif implique que sur chaque objectif $k \in P$ est défini une fonction de coût $c_k : A \to \mathbb{R}_+$, un opérateur binaire $\otimes_k : \mathbb{R}_+^2 \to \mathbb{R}_+$ qui agrège les coûts, une fonction objectif $z_k : R_{\bullet,\bullet} \to \mathbb{R}_+$. Toutes les fonctions objectif n'étant pas à minimiser alors par mesure de simplicité on définit l'inégalité $\leq_k$ telle $\leq_k \Longleftrightarrow \leq$ si l'objectif k est à minimiser et $\leq_k \Longleftrightarrow \geq$ si l'objectif k est à maximiser. De même on définit les éléments neutres $\mathbf{0}_1, \ldots, \mathbf{0}_p$ associés aux opérateurs binaires $\otimes_1, \ldots, \otimes_p$: $\mathbf{0}_k \otimes_k y_k = y_k$, $\forall y_k \in \mathbb{R}_+$. Pour un objectif $k \in P$: $\mathbf{0}_k = 0$ si $\otimes_k$ est l'opérateur $+$; $\mathbf{0}_k = -\infty$ (ou 0 si les coûts sont définis sur $\mathbb{R}_+$) si $\otimes_k$ est l'opérateur $\max$; $\mathbf{0}_k = 1$ si $\otimes_k$ est l'opérateur $\times$; $\mathbf{0}_k = +\infty$ si $\otimes_k$ est l'opérateur $\min$. On note que aussi que la notion d'efficacité est définie telle qu'un chemin (ou solution) $r \in R_{i,j}$ du nœud $i \in N$ au $j \in N$ est dit efficace s.s.i. r est efficace par rapport à l'ensemble des chemins de i à j : $\nexists r' \in R_{i,j}$ t.q. $r' \neq r$ et $z(r') \prec_P z(r)$.

Un problème MOSP est défini selon l'ensemble des solutions du problème $R_{s,\bullet}$ ou $R_{s,t}$ et selon les objectifs du problème. Les objectifs sont notés tels que [EG00, Tar07] : l-$\sum$ désigne l fonctions objectif additives à minimiser ; l-$\min$ désigne l fonctions objectif minimales à maximiser ; l-$\max$ désigne

l fonctions objectif maximales à minimiser ; l-$\prod$ désigne l fonctions objectif multiplicatives à minimiser (chaque coût sur ces objectifs est strictement positif) ; l-$\otimes$ désigne l fonctions objectif définies par l'opérateur $\otimes$ dont le sens d'optimisation n'est pas défini ; Q-$\sum$ désigne un nombre indéfini de fonctions objectif additives à minimiser, même chose pour Q-$\min$, Q-$\max$, Q-$\prod$ et Q-$\otimes$. Par exemple le problème MOSP $R_{s,t}$ avec objectifs Q-$\min$ 1-$\max$ est un problème MOSP du nœud source s au nœud puits t avec un nombre indéfini de fonctions objectif minimales à maximiser et une fonction objectif maximale à minimiser.

2.1.2.2 Complexité du problème MOSP

Le nombre de solutions efficaces est un élément déterminant de la difficulté d'un problème MOSP [War87, MHW06, SNB11]. Concernant le problème $R_{s,t}$, le nombre de solutions efficaces a été étudié dans plusieurs articles (cf. §2.2.3, page 58). On note que le nombre de solutions efficaces du problème $R_{s,\bullet}$ peut être calculé à partir du nombre de solutions efficaces du problème $R_{s,t}$ si t a été choisit aléatoirement dans N. Hansen [Han80] montre que la taille de l'ensemble maximal complet de solutions efficaces d'un problème $R_{s,t}$ l-$\sum$ ($l \geq 2$) est au pire exponentielle par rapport au nombre de nœuds. Il en est de même pour n'importe quel ensemble complet de solutions efficaces [SA00]. Néanmoins en pratique le nombre de solutions efficaces pour le problème $R_{s,t}$ est en général assez faible par rapport à d'autre problème MOCO [PGE10b]. Pour des instances non artificielles [MHW06, SNB11, RE09] du problème 2-$\sum$ le nombre de solutions efficaces est en général limité à une fraction du nombre de nœuds du graphe $|E_M(R_{s,t})| \leq \frac{|N|}{20}$. Il en est de même sur les instances artificielles 2-$\sum$ générées aléatoirement [MMO91, MHW06, GBR06, RE09], à l'exception des instances de type grille [MMO91, RE09] qui possèdent un nombre de solutions efficaces beaucoup plus important mais en général inférieur au nombre de nœud. Plus les valeurs sur les objectifs sont corrélées plus le nombre de solutions efficaces est petit, voir [SNB11] et [RE09] qui utilisent des instances de transport urbain avec des valeurs sur les objectifs qui sont corrélées de manière très différentes. Les rares expérimentations sur des instances 3-$\sum$ [MHW06, GBR06] montrent que le nombre de solutions efficaces augmentent de façon très importante par rapport aux instances 2-$\sum$. Gandibleux et al [GBR06] montrent, sur des instances Q-$\sum$ Q-$\min$ avec 2 ou 3 objectifs, que l'utilisation d'un objectif bottleneck à la place d'un objectif additif augmente sensiblement le nombre de solutions efficaces Les résultats de Iori et al [IMP10] sur des instances l-$\sum$t-min (avec $l + t \geq 4$) confirment les résultats précédents : plus le nombre d'objectifs additifs et surtout plus le nombre d'objectifs bottleneck est important plus le nombre de solutions efficaces est important. Ainsi sur certaines instances, le nombre de solutions efficaces est $|E_M(R_{s,t})| = 25 \times |N|$. De même plus le graphe est dense plus le nombre de solutions efficaces est important. L'influence de la densité semble légère dans le cas bi-objectif [GBR06, RE09] mais très forte avec plus de deux objectifs [GBR06, IMP10]. Müller et Weihe [MHW06] proposent une caractérisation des graphes dont le nombre de solutions efficaces est polynomial par rapport au nombre de nœuds.

Le nombre de solutions efficaces pouvant augmenter de façon exponentielle par rapport au nombre de nœuds alors la construction d'un ensemble complet de solutions efficaces du problème MOSP $R_{s,t}$ (et par conséquent $R_{s,\bullet}$) est un problème NP-difficile.

2.1.2.3 État de l'art

Concernant la résolution du problème MOSP, différents états de l'art sont disponibles dans la littérature (cf. Table 2.1, page suivante). Current et Min [CM86] proposent une classification de l'ensemble des algorithmes de résolution du problème MOSP. Une actualisation de l'article précédent est proposée

Auteur	Type d'état de l'art	Type de méthodes	citation
Current et Min 1986	classification	—	[CM86]
Current et Marsh 1993	classification	—	[CM93]
Huarng et al 1996	comparaison temps de calcul	exacte, $E(X)$	[HPS96]
Skriver 2000	analyse	exacte, $E(X)$	[Skr00]
Ehrgott et Gandibleux 2002	classification	—	[EG02]
Tarapata 2007	classification	—	[Tar07]
Climaco et Pascoal 2010	classification	exacte	[CP10]

Table 2.1 – États de l'art sur le problème MOSP, "exact" signifie que l'état de l'art concerne uniquement les méthodes exactes, $E(X)$ signifie que uniquement les méthodes pour construire un ensemble complet de solutions efficaces sont concernées et "—" signifie que toutes les méthodes sont concernées.

par Current et Marsh [CM93]. Huarng et al [HPS96] proposent une comparaison des temps de calcul des différents algorithmes exacts construisant l'ensemble des solutions efficaces. Skriver [Skr00] présente en détail les principaux algorithmes exacts de la littérature construisant l'ensemble des solutions efficaces. Ehrgott et Gandibleux [EG02] proposent une classification de l'ensemble des algorithmes pour le problème MOSP. Tarapata [Tar07] reprend la classification précédente et ajoute certaines publications plus récentes, néanmoins de nombreux articles ne sont pas cités. Climaco et Pascoal [CP10] utilise une classification (identique à celle de Current et Min) des algorithmes exacts.

2.1.2.4 Présentation du chapitre

Ce chapitre est un état de l'art concernant les algorithmes de résolution du problème de plus court chemin multi-objectif. La résolution d'un problème MOSP peut être (a) un ensemble complet de solutions efficaces ($E(X)$) ou (b) une solution qui optimise une fonction d'utilité. Les algorithmes pour résoudre le premier cas (a) sont traités dans les sections 2.2 et 2.3. La section 2.2 présente les algorithmes d'étiquetage et la section 2.3 présente les autres algorithmes. De plus, contrairement aux autres états de l'art, nous proposons de différencier les algorithmes de résolution du problème MOSP $R_{s,t}$ (cf. §2.2.3, page 58) de ceux de résolution du problème MOSP $R_{s,\bullet}$ (cf. §2.2.2, page 55). Les algorithmes pour résoudre le deuxième cas (b) sont traités dans la section 2.4. Précisons que tous les algorithmes pour l'optimisation d'une fonction d'utilité sont des algorithmes d'étiquetage.

Il existe dans le cadre du problème de plus court chemin de nombreuses méthodes dites de "*preprocessing*" [GSSD08]. Ces méthodes sont utilisées quand différentes instances du problème $R_{s,t}$ sont résolues sur un même graphe, i.e. avec des nœuds source et des nœuds puits différents. Ces méthodes permettent en autre de réduire le nombre de nœuds ou d'arcs du graphe original, ce qui permet in fine de réduire les temps de calcul de l'algorithme de plus court chemin utilisé pour résoudre chacune de ces instances. Ces méthodes sont particulièrement utiles dans le cadre de problèmes de transports routiers, où la taille des graphes est particulièrement importante [Gra10, SNB11]. Certaines de ces méthodes ont été adaptées aux problèmes multi-objectifs [DW09, Gra10, SNB11]. Ces méthodes ne sont pas détaillées dans cette thèse.

2.2 Les algorithmes d'étiquetage pour le problème MOSP

Les algorithmes d'étiquetage permettent de résoudre deux types de problèmes MOSP, le problème MOSP $R_{s,t}$ et le problème MOSP $R_{s,\bullet}$. Le problème MOSP $R_{s,\bullet}$ étant plus général, tout algorithme qui résout le problème MOSP $R_{s,\bullet}$ résout le problème MOSP $R_{s,t}$ quel que soit le nœud puits.

La sous-section 2.2.1 présente le principe général des algorithmes d'étiquetage. Dans cette section nous proposons des théorèmes permettant de valider l'utilisation d'un algorithme d'étiquetage pour un problème MOSP avec des objectifs qui ne sont pas des sommes. Ensuite la sous-section 2.2.2 présente les algorithmes d'étiquetage existant pour le problème $R_{s,\bullet}$. Pour finir, la sous-section 2.2.3 présente les algorithmes d'étiquetage existant pour le problème $R_{s,t}$.

2.2.1 Principe des algorithmes d'étiquetage

Les algorithmes d'étiquetage utilisent des d'étiquettes (*labels*) comme structure de données efficace pour stocker des chemins avec leurs sous-chemins. Une étiquette ou label $\ell_j = [y_1, \ldots, y_p, i, \ell_i]$, $j \in N$, *correspond* à un chemin $r = \langle s, \ldots, i, j\rangle \in R_{s,j}$ tel que $z(r) = (y_1, \ldots, y_p)$ est le vecteur des coûts du chemin r et $i \in N$ est le nœud auquel est associé le plus long sous-chemin $r' = \langle s, \ldots, i\rangle \in R_{s,i}$ de r. Ainsi l'ensemble des labels forme l'arbre des plus courts chemins du nœud source à tous les autres nœuds et il est donc possible par récursion de reconstruire ces chemins à partir des labels. On note L_i, $i \in N$ l'ensemble des labels ℓ_i associé au nœud i.

La "*propagation*" d'un label ℓ_j, $j \in N$, consiste à la création de nouveau labels associés aux nœuds adjacents à $i : j \in \Gamma^+(i)$. Ces nouveaux labels correspondent aux extensions du chemin associé à ℓ_j par les arcs (i, j), $j \in \Gamma^+(i :)$. $\mathfrak{L}$ est l'ensemble des *labels ouverts*, i.e. l'ensemble des labels qui n'ont pas encore été propagés. Le label initial est $[\mathbf{0}_1, \ldots, \mathbf{0}_p, s, \star]$. Une *règle de dominance* $\prec$ est utilisée pour comparer les labels entre eux. Le schéma général des algorithmes d'étiquetage est décrit par l'algorithme 2.1.

Algorithme 2.1. : Algorithme d'étiquetage général

1. **Initialisation :** $L_i := \emptyset, \forall i \in N$, le label initial $\ell_s = [\mathbf{0}_1, \ldots, \mathbf{0}_p, \star, \star]$ est ajouté à l'ensemble des labels du nœud source $L_s := \{\ell_s\}$ et à l'ensemble des labels ouverts $\mathfrak{L} := \{\ell_s\}$.
2. **Sélection :** Le label courant ℓ_i, associé au nœud $i \in N$, est sélectionné parmi l'ensemble des labels ouverts $\mathfrak{L}$ suivant une **règle de sélection**. ℓ_i est retiré de cet ensemble : $\mathfrak{L} := \mathfrak{L} \setminus \{\ell\_i\}$.
3. **Propagation :** Le label ℓ_i est propagé aux nœuds adjacents à i : $\forall j \in \Gamma^+(i)$, $\ell_j := [z_1(\ell_i) \otimes_1 c_1(i,j), \ldots, z_p(\ell_i) \otimes_p c_p(i,j), i, \ell_i]$. Chaque label propagé, ℓ_j, $j \in \Gamma^+(i)$ est ajouté à l'ensemble des labels de j et à l'ensemble des labels ouverts : $L_j := L_j \bigcup \{\ell_j\}$ et $\mathfrak{L} := \mathfrak{L} \bigcup \{\ell_j\}$.
4. **Règle de dominance :** La dominance $\prec$ est vérifiée dans les ensembles de labels associés aux nœuds adjacents de i : les labels dominés sont supprimés, $\forall j \in \Gamma^+(i)$, $L_j := L_j \setminus \{\ell_j^a \in L_j : \exists \ell_j^b \in L_j, z(\ell_j^b) \prec z(\ell_j^a)\}$. Les labels dominés sont aussi supprimés de l'ensemble des labels ouverts : $\forall j \in N$, $\mathfrak{L} := \mathfrak{L} \setminus \{\ell_j^a \in \mathfrak{L} : \exists \ell_j^b \in L_j, z(\ell_j^b) \prec z(\ell_j^a)\}$.
5. **Test de terminaison :** Si $\mathfrak{L} = \emptyset$ alors l'algorithme se termine et $\bigcup_{i \in N} L_i$ correspond à l'ensemble des chemins $\prec$-efficaces construits, sinon aller en 2.

Précisons que si le graphe est acyclique alors l'algorithme d'étiquetage est équivalent à une équation de programmation dynamique : un label ℓ_i ($i \in N$) est sélectionné uniquement si il n'existe plus de label ouvert associé aux nœuds précédents du nœud i, $\Gamma^-(i)$ [Hen86, CMM90, GV02]. Il existe de multiples variantes de l'algorithme d'étiquetage qui sont présentés dans les sous-sections 2.2.2 et 2.2.3. On démontre ci-dessous que de manière générale les algorithmes d'étiquetage terminent (cf. Théorème 2.5, page 55) et qu'ils construisent l'ensemble maximal complet des solutions efficaces (cf. Définition 1.2, page 13).

2.2.1.1 Construction de l'ensemble des solutions efficaces

Le *principe d'optimalité* de Bellman [Bel57] qui indique que "*une solution optimale est composée uniquement de sous-solutions optimales*" est vérifié dans le cadre du problème de plus court chemin mono-objectif avec un objectif additif à minimiser. Ce principe permet d'utiliser des algorithmes de programmation dynamique pour résoudre de manière exacte ce problème, tous les algorithmes d'étiquetage étant des algorithmes de programmation dynamique. Le *principe fort d'efficacité* est une généralisation du principe d'optimalité de Bellman dans le cadre du problème MOSP.

Définition 2.4. Principe d'efficacité fort [MS00].
Un chemin efficace est composé uniquement de sous-chemins efficaces.

Théorème 2.1. Validité du principe d'efficacité fort.
Le principe d'efficacité fort est vérifié si $\forall y^a, y^b, y^c \in \mathbb{R}^p$, $\forall k \in P$:

$$y_k^a <_k y_k^b \Longrightarrow y_k^a \otimes_k y_k^c <_k y_k^b \otimes_k y_k^c$$

avec $\otimes_k$ *l'opérateur binaire et* $<_k$ *l'inéquation associé à l'objectif* k.

Preuve du Théorème 2.1.
On suppose que $\forall y^a, y^b, y^c \in \mathbb{R}^p$, *chaque objectif* $k \in P$ *est défini* $(\otimes_k, \leq_k, <_k)$ *tel que :*

$$y_k^a <_k y_k^b \Longrightarrow y_k^a \otimes_k y_k^c <_k y_k^b \otimes_k y_k^c \tag{2.3}$$

Si $y_k^a = y_k^b$ *alors par définition* $y_k^a \otimes_k y_k^c = y_k^b \otimes_k y_k^c$. *Par conséquent :*

$$y_k^a \leq_k y_k^b \Longrightarrow y_k^a \otimes_k y_k^c \leq_k y_k^b \otimes_k y_k^c \tag{2.4}$$

Selon (2.3) et (2.4), si la dominance de Pareto est vérifiée entre y^a *et* y^b *alors elle est aussi vérifiée entre* $y^a \otimes y^c$ *et* $y^b \otimes y^c$ *:*

$$y^a \prec_P y^b \Longrightarrow y^a \otimes y^c \prec_P y^b \otimes y^c \tag{2.5}$$

Soient $r = \langle s, v_1, \ldots, v_l \rangle \in R_{s,v_l}$ *un chemin efficace.* $r^a = \langle s, v_1, \ldots, v_h \rangle \in R_{s,v_h}$ *un sous-chemin quelconque de* r. *Si* r^a *n'est pas efficace alors il existe un chemin* $r^b \in R_{s,v_h}$ *tel que* $r^b \prec_P r^a$. *Alors le chemin* $r^b \Diamond \langle v_h, \ldots, v_l \rangle$ *domine le chemin* r *:* $z(r^b) \otimes z(\langle v_h, \ldots, v_l \rangle) \prec_P z(r)$, *avec* $z(r) = z(r^a) \otimes z(\langle v_h, \ldots, v_l \rangle)$. *Ainsi le chemin* r *n'est pas efficace.*
Réduction par l'absurde : le principe d'efficacité fort est vérifié.

□

On peut facilement montrer que les problèmes MOSP Q-$\sum$ et Q-$\sum$ Q-$\prod$ (si les coûts sont positifs $c(i,j) \in \mathbb{R}_+^p, \forall (i,j) \in A$) vérifient le théorème 2.1 et donc le principe d'efficacité fort.

Corollaire 2.1. Validité du principe d'efficacité fort [Mar84a].
Le principe d'efficacité fort est vérifié dans le cadre du problème Q-$\sum$.

Si le principe d'efficacité fort est respecté alors il suffit que la règle de dominance vérifie la dominance de Pareto ($\prec_P$) entre les labels pour que l'algorithme d'étiquetage construisent l'ensemble des solutions efficaces (cf. Théorème 2.2, de la présente page).

Théorème 2.2. Construction de l'ensemble maximal complet des solutions efficaces [Mar84a].
Dans le cadre du problème MOSP Q-$\sum$, *si uniquement des labels dominés sont supprimés durant l'exécution de l'algorithme d'étiquetage (cf. Algorithme 2.1, page précédente) alors l'ensemble maximal complet des solutions efficaces est construit à la fin de l'algorithme :* $\bigcup_{i \in N} L_i = E_M(R_{s,\bullet})$.

Preuve du Théorème 2.2.
Soit $r \in E(R_{s,\bullet})$ *un chemin efficace de* s *à un nœud quelconque de* N *:*

- *Si* r *est composé d'aucun arc alors* $r = \langle s \rangle$. *Ce chemin est construit durant l'initialisation de l'algorithme d'étiquetage.*
- *Si* r *est composé d'un seul arc alors* $r = \langle s, v_1 \rangle$. *Le chemin* $\langle s \rangle$ *étant efficace alors ce dernier est construit lors de l'initialisation de l'algorithme puis étendu vers le nœud* v_1 *car* $v_1 \in \Gamma^+(s)$.
- *Si* r *est composé de* l *arcs alors* $r = \langle s, v_1, \ldots, v_{l-1}, v_l \rangle$. *Le chemin* $\langle s, v_1, \ldots, v_{l-1} \rangle$ *étant efficace (cf. Définition 2.4, page précédente) alors il est construit puis étendu vers* v_l *car* $v_l \in \Gamma^+(v_{l-1})$.

Par récurrence, tout chemin efficace de s *à un autre nœud est construit par l'algorithme d'étiquetage (cf. Algorithme 2.1, page 51) si uniquement les labels correspondant aux chemins dominés sont supprimés par l'algorithme.*

□

2.2.1.2 Construction de l'ensemble des solutions efficaces avec des objectifs bottleneck

Dans le cadre du problème plus général $Q\text{-}\sum\ Q\text{-}\max\ Q\text{-}\min$ (avec au moins un objectif bottleneck), le principe d'efficacité fort n'est pas vérifié [GBR06]. Martins et Santos [MS00] proposent alors un principe d'efficacité plus général.

Définition 2.5. Principe d'efficacité faible [MS00]
Pour chaque point non-dominé il existe une solution efficace qui est composée uniquement de sous-chemins efficaces.

Théorème 2.3. Validité du principe d'efficacité faible.
Le principe d'efficacité faible est vérifié si $\forall y^a, y^b, y^c \in \mathbb{R}^p$, $\forall k \in P$:

$$y^a_k <_k y^b_k \Longrightarrow y^a_k \otimes_k y^c_k \leq_k y^b_k \otimes_k y^c_k$$

Preuve du Théorème 2.3.
On suppose que $\forall y^a, y^b, y^c \in \mathbb{R}^p$, *chaque objectif* $k \in P$ *est défini* $(\otimes_k, \leq_k, <_k)$ *tel que :*

$$y^a_k <_k y^b_k \Longrightarrow y^a_k \otimes_k y^c_k \leq_k y^b_k \otimes_k y^c_k \tag{2.6}$$

Si $y^a_k = y^b_k$ *alors par définition* $y^a_k \otimes_k y^c_k = y^b_k \otimes_k y^c_k$. *Par conséquent :*

$$y^a_k \leq_k y^b_k \Longrightarrow y^a_k \otimes_k y^c_k \leq_k y^b_k \otimes_k y^c_k \tag{2.7}$$

Selon 2.6 et 2.7, si la dominance de Pareto est vérifiée entre y^a *et* y^b *alors* $y^a \otimes y^c$ *est meilleur ou égale à* $y^b \otimes y^c$ *sur chaque objectif :*

$$y^a \prec_P y^b \Longrightarrow y^a \otimes y^c \leq_k y^b \otimes y^c,\ \forall k \in P \tag{2.8}$$

Soient $r = \langle s, v_1, \ldots, v_l \rangle \in R_{s,v_l}$ *un chemin efficace,* $r^a = \langle s, v_1, \ldots, v_h \rangle \in R_{s,v_h}$ *un sous-chemin quelconque de* r. *Si* r^a *n'est pas efficace alors il existe un chemin* $r^b \in R_{s,v_h}$ *tel que* $r^b \prec_P r^a$. *Alors le chemin* $r' = r^b \Diamond \langle v_h, \ldots, v_l \rangle$ *domine ou a un vecteur de coûts égal au chemin* r *:*

$$z(r') = z(r^b) \otimes_k z(\langle v_h, \ldots, v_l \rangle) \leq_k z(r) = z(r^a) \otimes_k z(\langle v_h, \ldots, v_l \rangle),\ \forall k \in P \tag{2.9}$$

Comme r *est efficace alors* $z(r') = z(r)$ *et donc* r' *et* r *sont deux chemins équivalents, i.e. leur point dans l'espace des objectif est identique. Alors il existe un troisième chemin* $r'' \in R_{s,l}$ *tel que tous les sous-chemins de ce chemins sont efficaces et que* r'', r', r *sont trois chemins équivalents.*
Le principe d'efficacité faible est vérifié.

□

On peut facilement montrer que le problème $Q\text{-}\sum\ Q\text{-}\prod\ Q\text{-}\max\ Q\text{-}\min$ (avec des coûts positifs $c(i,j) \in \mathbb{R}^p_+, \forall (i,j) \in A$) vérifie le théorème 2.3 et donc le principe d'efficacité faible. Si le principe d'efficacité faible est vérifié alors l'utilisation de la dominance de Pareto pour supprimer des labels implique qu'un ensemble complet de solutions efficaces est construit ($\bigcup_{i \in N} L_i = E(X)$). Certaines solutions efficaces ne sont pas construites : en effet, le label associé au sous-chemin non efficace d'une solution efficace peut avoir été supprimé avant d'avoir été propagé. En pratique il est parfois nécessaire de connaitre l'ensemble maximal complet de solutions efficaces [RP11], par exemple si il n'est pas possible d'emprunter un chemin efficace alors il est intéressant de connaitre un chemin efficace équivalent. Dans le but de construire l'ensemble maximal complet de solutions efficaces dans le cadre du problème $Q\text{-}\sum\ Q\text{-}\prod\ Q\text{-}\min\ Q\text{-}\max$, une dominance plus faible que la dominance de Pareto peut être utilisée : la dominance $\prec_{GBR}$ [GBR06].

Définition 2.6. Dominance GBR [GBR06].
Soit un problème MOSP *avec l fonctions objectif additives ou multiplicatives $(z_1, \ldots, z_l)$ et $p-l$ fonctions objectif bottleneck $(z_{l+1}, \ldots, z_{l+2})$. Pour deux vecteurs de coûts $y^a, y^b \in \mathbb{R}^p$, la dominance* GBR *entre y^a et y^b est définie telle que :*

$$y^a \prec_{GBR} y^b \iff \forall k \in \{1, .., l\}\ y^a_k \leq_k y^b_k \text{ et } \exists k \in \{1, \ldots, l\} \text{ t.q. } y^a_k <_k y^b_k$$

L'utilisation de la règle de dominance GBR dans un algorithme d'étiquetage pour le problème MOSP (avec des objectifs bottleneck) implique que l'algorithme d'étiquetage construit l'ensemble maximal complet des solutions efficaces $E_M(R_{s,\bullet})$.

Théorème 2.4. Construction de l'ensemble maximal complet des solutions efficaces avec des objectifs bottleneck [GBR06].
Dans le cadre du problème MOSP $Q\text{-}\sum\ Q\text{-}\prod\ Q\text{-}\min\ Q\text{-}\max$, *si uniquement des labels dominés selon la dominance $\prec_{GBR}$ sont supprimés durant l'exécution de l'algorithme d'étiquetage (cf. Algorithme 2.1, page 51), alors l'ensemble maximal complet des solutions efficaces est construit : $\bigcup_{i \in N} L_i = E_M(R_{s,\bullet})$.*

Preuve du Théorème 2.4.
Identique à la preuve 2.2.

□

2.2.1.3 Terminaison des algorithmes d'étiquetage

La terminaison des algorithmes d'étiquetage est associé avec l'existence d'un cycle avec des coûts négatifs ou nuls [MS00]. Dans le cadre général du problème MOSP avec plusieurs objectifs quelconque Q-$\otimes$ la notion de coût positif (ni négatif ni nul) est redéfini.

Définition 2.7. Coût positif.
Soit un objectif $k \in P$ associé à un opérateur binaire $\otimes_k$, avec un sens d'optimisatiton défini par l'inéquation $<_k$ et un élément neutre $\mathbf{0}_k$. Un coût $y_k \in \mathbb{R}$ est positif *si $\mathbf{0}_k <_k y_k$ (afin de respecter la notion d'élément neutre) et $0 < y_k$ (afin de respecter la notion usuel de coût positif).*

Ainsi pour un objectif multiplicatif, un coût est positif si il est supérieur à 1, neutre si il est égale à 1 et négatif si il est inférieur à 1. Le théorème 2.5 (ci-dessous) montre que sous certaines conditions l'algorithme d'étiquetage 2.1 termine.

Théorème 2.5. Terminaison de l'algorithme d'étiquetage [MS00].
L'algorithme d'étiquetage 2.1 termine si pour tout cycle $r \in R_{\bullet,\bullet}$:

(a) r a des coûts positifs ou nuls, i.e. le vecteur de coûts du cycle domine le vecteur d'éléments neutres [MS00] : $(\mathbf{0}_1, \ldots, \mathbf{0}_p) \prec_P z(r)$.

(b) il existe un objectif $k \in P$ tel que le coût de r sur cet objectif soit positif ($\mathbf{0}_k <_k z_k(r)$) et l'opérateur binaire de l'objectif est strictement croissant, $\forall y^a_k, y^b_k \in \mathbb{R}$: $\mathbf{0}_k <_k y^b \Rightarrow y^a_k <_k y^a_k \otimes_k y^b_k$.

La non existence de cycle nul ou négatif est une condition courante dans les problèmes de graphes, ainsi la plupart des graphes modélisant un réseau routier ou de télécommunication respectent les conditions (a) et (b). Les opérateurs binaires $+$ et $\times$ respectent la condition (b), les opérateurs binaires bottleneck ($\max$,$\min$) ne respectent pas la condition (b).

2.2.1.4 Types d'algorithmes d'étiquetage

Chaque algorithme d'étiquetage pour le problème MOSP peut être classé dans l'une des trois catégories suivantes en fonction de la règle de sélection :

- algorithme de mise en étiquette "*label setting*" (LS), le label ouvert minimisant une fonction d'agrégation des coûts est sélectionné, ce label doit correspondre à un chemin efficace.
- algorithme de correction d'étiquette "*label correcting*" avec stratégie de sélection de nœud (LC/N), un nœud est sélectionné puis les labels ouverts associés à ce nœud sont sélectionnés puis propagés les uns après les autres.
- algorithme de correction d'étiquette "*label correcting*" avec stratégie de sélection de label (LC/L), une règle simple sur l'ensemble des labels ouverts est utilisée afin de sélectionner rapidement un label, le label sélectionné est quelconque.

Les trois catégories sont définies plus précisément dans la sous-section 2.2.2.

2.2.2 Algorithmes d'étiquetage pour le problème $R_{s,\bullet}$

Le tableau 2.2 catégorise chaque algorithme d'étiquetage pour le problème MOSP $R_{s,\bullet}$ proposé dans la littérature. La plupart de ces algorithmes emploient la dominance de Pareto ($\prec_P$) dans la règle de dominance, excepté les algorithmes proposés dans [GBR06, IMP10] qui emploient la dominance $\prec_{GBR}$ (cf. Définition 2.6, page ci-contre). Chacun de ces algorithmes est décrit en détails dans cette sous-section. Les algorithmes LS sont présentés dans un premier temps, puis les algorithmes LC/N, pour finir les algorithmes LC/L sont présentés.

Auteurs	Objectifs	Type	Ensemble	Citation
Hansen 1980	2-$\sum$	LS	$E_M(R_{s,\bullet})$	[Han80]
Daellenbach and De Kluyver 1980	2-$\sum$	LC/N	$E_M(R_{s,\bullet})$	[DD80]
Martins 1984	Q-$\sum$	LS	$E_M(R_{s,\bullet})$	[Mar84a]
Corley and Moon 1985	Q-$\sum$	LC/N	$E_M(R_{s,\bullet})$	[CM85]
Henig 1986	2-$\sum$	DP, LC/N	$E_M(R_{s,\bullet})$	[Hen86]
Brumbaugh-Smith et Shier 1989	2-$\sum$	LC/N	$E_M(R_{s,\bullet})$	[BSS89]
Martins et Santos 2000	Q-$\sum$	LC/L,LS	$E_M(R_{s,\bullet})$	[MS00]
Guerriero et Musmanno 2001	Q-$\sum$	LC/L	$E_M(R_{s,\bullet})$	[GM01]
Gandibleux et al 2006	Q-$\sum$ 1-min	LS	$E_M(R_{s,\bullet})$	[GBR06]
Paixao et Santos 2007	Q-$\sum$	LC/L,LS	$E_M(R_{s,\bullet})$	[PRS07]
Iori et al 2010	Q-$\sum$ Q-min	LC	$E_M(R_{s,\bullet})$	[IMP10]

Table 2.2 – Littérature sur les algorithmes d'étiquetage pour le problème $R_{s,\bullet}$. La colonne "Ensemble" indique l'ensemble de solutions calculé par l'algorithme.

2.2.2.1 Algorithmes de mise en étiquette (label setting, LS)

Si la règle de sélection d'un algorithme d'étiquetage respecte le principe LS alors cet algorithme est de LS.

Définition 2.8. Principe LS pour une règle de sélection.
Le label sélectionné à chaque itération de l'algorithme d'étiquetage correspond à un chemin efficace.

Si ce principe est respecté alors un nombre minimal de propagation de label (itération) est garantie, puisque exactement un label sera propagé pour chaque chemin efficace de $R_{s,\bullet}$. Si le principe LS n'est pas respecté alors l'algorithme d'étiquetage est un algorithme LC. Considérons que les coûts sur les arcs soient positifs (cf. Définition 2.7, page 54) ou nuls alors il existe une condition suffisante pour que le principe LS soit vérifié.

Théorème 2.6. Condition pour qu'une règle de sélection vérifie le principe LS [PRS07]
Soit ℓ_i un label associé à un nœud $i \in N$, sélectionné durant une itération de l'algorithme d'étiquetage. Si le point du label courant $z(\ell_i)$ est non dominé par rapport à l'ensemble des labels ouverts $\mathfrak{L}$ ($\nexists \ell'_i \in \mathfrak{L}$ t.q. $z(\ell'_i) \prec_P z(\ell_i)$), alors le principe LS *(cf. Définition 2.8, page précédente) est vérifié et ℓ_i correspond à un chemin efficace de s à i.*

Le théorème 2.6 implique que le label sélectionné soit efficace à chaque itération. Tous les algorithmes LS de la littérature utilisent une fonction d'agrégation des objectifs $h : \mathbb{R}^p_+ \to \mathbb{R}_+$ telle que le label ouvert minimisant h est sélectionné à chaque itération. La fonction h doit être strictement croissante sur chaque objectif (cf. Définition 1.9, page 18). Cette fonction peut être une fonction lexicographique, une somme pondérée, ... En général une structure de données triée est utilisée pour $\mathfrak{L}$ afin de conserver les labels ordonnés selon la fonction h. Le choix de la structure de données est très important [PRS07, Rai10] afin d'éviter d'appliquer la fonction h à tous les labels ouverts afin de trouver le plus petit. Le principe LS implique que le nombre d'itérations des algorithmes LS est minimal. En effet, dans les algorithmes LS il y a exactement une itération par solution efficace (du problème $R_{s,\bullet}$). Or, chaque solution efficace devant être propagée, il n'est pas possible de faire moins d'une itération par solution efficace. Néanmoins, à chaque itération un temps de calcul non négligeable est requis pour sélectionner un label non dominé.

Hansen [Han80] propose un algorithme LS pour le problème 2-$\sum$. La fonction de sélection h est une fonction lexicographique (cf. Définition 1.4, page 15). Martins [Mar84a] généralise l'algorithme de Hansen au problème multi-objectif Q-$\sum$ et démontre que le principe LS est vérifié.

Martins et Santos [MS00] proposent d'utiliser une structure de liste triée pour stocker chaque ensemble de labels $L_1, \ldots, L_n$ afin que les labels soient ordonnés selon l'ordre lexicographique dans chacun de ces ensembles. Cette méthode permet réduire le temps de calcul lors de la vérification de la règle de dominance. De plus, garder une telle liste ordonnée ne nécessite aucun temps de calcul supplémentaire [MS00] car l'insertion des nouveaux labels dans une telle liste est faite durant la vérification de la règle de dominance. Cette méthode peut aussi être utilisée dans les algorithmes de LC/N.

Gandibleux et al [GBR06] proposent une adaptation de l'algorithme de Martins au problème 2-$\sum$ 1-min. h est une fonction lexicographique et les auteurs proposent la règle de dominance $\prec_{GBR}$ (cf. Définition 2.6, page 54).

Paixao et Santos [PRS07] étudient l'impact de différentes structures de données pour stocker l'ensemble des labels ouverts $\mathfrak{L}$: une liste, un tas binaire, ... Ils étudient aussi différentes fonctions h : une somme, une fonction lexicographique, ... Les différentes versions de l'algorithme LS ainsi obtenues sont comparées à l'algorithme de Guerriero et Musmanno [GM01] (LC/L avec stratégie FIFO). La sélection des labels selon une somme pondérée apparait comme une des meilleures stratégies et celle selon un ordre lexicographique comme l'une des moins performantes. L'utilisation d'une structure de tas binaire apparait comme une des structure de donnée les plus performantes pour stocker les labels ouverts, Raith [Rai10] en arrive à la même conclusion. Les temps de calcul entre la meilleure version de l'algorithme LS proposé et ceux de l'algorithme LC/L (FIFO) proposé sont similaires.

Iori et al [IMP10] proposent une généralisation de l'algorithme de Gandibleux et al [GBR06] pour le problème t-$\sum$ Q-min (avec t un nombre quelconque d'objectifs additifs). Les auteurs proposent de

conserver mais de ne pas étendre les labels qui sont $\prec_P$-dominés mais non $\prec_{GBR}$-dominés, ces labels sont alors dit "cachés". Les chemins efficaces engendrés depuis ces labels cachés peuvent être retrouvés à partir des chemins efficaces générés depuis les labels qui $\prec_P$-dominent les labels cachés. Dans un deuxième temps ils proposent d'utiliser une somme pondérée pour sélectionner le label courant : $h(\ell) = \alpha.\sum_{k=1}^{t} z_k(\ell) + \beta.\sum_{k=t+1}^{p} z_k(\ell)$, avec $\alpha, \beta \in \mathbb{R}_+ \setminus \{0\}$ sont deux valeurs strictement positives. L'influence des valeurs de α et β sur les temps de calcul est ensuite étudiée.

2.2.2.2 Algorithmes de correction d'étiquette avec une stratégie de sélection de nœud (LC/N)

Les algorithmes de correction d'étiquette (*label correcting*) avec stratégie de sélection de nœud (LC/N) sont des variantes de l'algorithme général d'étiquetage. À chaque itération de l'algorithme, plutôt que de propager un un seul label ouvert (le label courant), les algorithmes LC/N propagent l'ensemble des labels ouverts associés $l_i \in L_i \bigcap \mathfrak{L}$ à un nœud $i \in N$ (le nœud courant). Ainsi le nœud courant i est sélectionné pendant la phase de sélection puis la phase de propagation est répétée pour chaque label ouvert associé à i. Lors de l'application de la règle de dominance, plusieurs labels ont été ajoutés dans les ensembles de labels ($L_j, j \in \Gamma^+(i)$) associés aux nœuds adjacents au nœud courant. La vérification de la règle de dominance étant une étape critique pour le temps de calcul des algorithmes d'étiquetage. Alors plusieurs algorithmes spécifiques pour vérifier la règle de dominance , appelés "MERGE", sont proposés dans la littérature. On note aussi que comme plusieurs labels sont propagés à chaque itération, il est en général impossible de vérifier le principe LS.

Corley et Moon [CM85] proposent une adaptation de l'algorithme de Bellman [Bel58] pour le problème de plus court chemin Q-$\sum$. L'algorithme sélectionne les nœuds les uns après les autres selon leur numéro dans l'ensemble des nœuds $N = \{1, 2, \ldots, n\}$. Chaque nœud est propagé exactement n fois durant l'algorithme, et si il reste encore des labels ouverts ($\mathfrak{L} \neq \emptyset$) à la fin de la procédure alors il existe un cycle r négatif (i.e. $\exists k \in P z_k(r) < 0$). À une itération de l'algorithme, soit t le nombre de fois que chaque nœud a été sélectionné. Durant cette itération, seuls les labels correspondant à des chemins composés de exactement t arcs sont sélectionnés et propagés. Un algorithme MERGE basé sur le tri des ensembles de labels $L_1, \ldots, L_p$ est proposé.

Henig [Hen86] propose une équation de programmation dynamique qui peut être assimilée à un algorithme LC/N. Néanmoins cette équation n'est valide que dans le cadre des graphes acycliques. De plus, aucun détail n'est fourni sur la règle de sélection du nœud courant.

Brumbaugh-Smith et Shier [BSS89] proposent un cadre général pour l'algorithme de correction d'étiquette avec stratégie de sélection de nœud. Soit $\mathfrak{N}$ l'ensemble des nœuds auxquels est associé au moins un label dans $\mathfrak{L}$. Plusieurs stratégies sont proposées pour sélectionner le nœud courant dans $\mathfrak{N}$. Deux des stratégies proposées sont : "*first in first out*" (FIFO) où le premier nœud entré dans $\mathfrak{N}$ est sélectionné et "*last in first out*" (LIFO) où le dernier nœud entré dans $\mathfrak{N}$ est sélectionné. Les autres stratégies proposées sont des variantes de ces deux stratégies. Les résultats numériques montrent que la stratégie FIFO est la plus efficace de toutes. De plus, un algorithme MERGE spécifique est proposé : cet algorithme repose sur le tri, selon l'ordre lexicographique, des ensemble, de labels $L_1, \ldots, L_n$, l'idée est identique à celle de Martins et Santos [MS00] (cf. §2.2.2.1, page 55). On note que l'algorithme proposé, dans le cadre de la programmation dynamique et uniquement pour des graphes acycliques, par Daellenbach et De Kluyver [DD80] est équivalent à l'algorithme de Brumbaugh-Smith et Shier.

Guerriero et Musmanno [GM01] proposent deux nouvelles stratégies pour sélectionner le nœud courant. Ces deux nouvelles stratégies donnent des temps de calcul similaires à la stratégie FIFO.

2.2.2.3 Algorithmes de correction d'étiquette avec stratégie de sélection de label

Les algorithmes de correction d'étiquette (*label correcting*) avec stratégie de sélection de label (LC/N) sont la variante la plus simple de l'algorithme général d'étiquetage. Le principe LS n'étant pas (nécessairement) respecté, la règle de sélection peut être quelconque. Ainsi l'ensemble des labels ouverts ($\mathfrak{L}$) n'est pas ordonné et des règles de sélection simples sont utilisées : par exemple la règle "*First In First Out*" (FIFO) où le premier label entré dans $\mathfrak{L}$ est sélectionné.

Guerriero et Musmanno [GM01] propose un algorithme LC/L pour le problème Q-$\sum$. Plusieurs règles de sélection sont proposées, notamment les stratégies FIFO et LIFO. L'algorithme est comparé avec l'algorithme LC/N de [BSS89] (stratégie FIFO) et avec l'algorithme LS de Martins [Mar84a]. Les différentes règles de sélection proposées pour l'algorithme LC/L donnent des temps de calcul similaires. Les temps de calcul de l'algorithme LC/L sont meilleurs que ceux de l'algorithme LC/N et similaires avec ceux de l'algorithme de LS.

Martins et Santos [PRS07] proposent une nouvelle règle de sélection, basée sur une fonction de sélection h (aussi utilisée dans les algorithmes LS) et une liste doublement chainée. Cette règle de sélection est comparée à la règle FIFO proposée par Guerriero et Musmanno [GM01] : ces deux règles de sélection donnent à l'algorithme LC/L des temps de calcul similaires.

2.2.3 Algorithmes d'étiquetage pour le problème $R_{s,t}$

Les algorithmes d'étiquetage pour résoudre le problème $R_{s,t}$ sont basés sur les algorithmes d'étiquetage pour le problème $R_{s,\bullet}$ (cf. §2.2.2, page 55). Il en est de même dans le cas mono-objectif : l'algorithme d'étiquetage A* qui permet de résoudre le problème $R_{s,t}$, est basé sur l'algorithme de Dijkstra qui permet de résoudre le problème $R_{s,\bullet}$ (cf. §2.1.1, page 45). Tous les algorithmes présentés ici s'intéressent aux problèmes Q-$\sum$ ou 2-$\sum$ et la dominance utilisée est toujours la dominance de Pareto.

2.2.3.1 Définitions

Les algorithmes d'étiquetage pour le problème $R_{s,t}$ utilisent une étape supplémentaire par rapport aux algorithmes $R_{s,\bullet}$ afin de propager moins de label et donc diminuer le nombre d'itérations.

Définition 2.9. Règle de coupe.
Dans l'algorithme d'étiquetage (cf. Algorithme 2.1, page 51), une règle de coupe peut être ajoutée entre l'étape de propagation et l'étape de règle de dominance :

3bis. **Règle de coupe :** *Pour chaque label propagé $\ell_j, j \in \Gamma^+(i)$, si la règle de coupe est vérifiée alors le label propagé ℓ_j est supprimé de l'ensemble des labels ouverts et de l'ensemble des labels associés au nœud j, $\mathfrak{L} = \mathfrak{L} \setminus \{\ell_j\}$ et $L_j = L_j \setminus \{\ell_j\}$.*

Cette règle de coupe est aussi parfois appelée "*goal dominance*" [RP11] puisque les labels sont (en général) comparés aux labels associés au nœud puits t (le *goal*).

De nombreux algorithmes pour le problème $R_{s,t}$ utilisent, dans la règle de coupe, une fonction vectorielle $q : N \to \mathbb{R}^p_+$ appelée fonction d'évaluation (ou heuristique) (cf. Figure 2.1, de la présente page) (cf. Définition 2.10, de la présente page). Cette fonction donne une évaluation du coût chemin restant à parcourir depuis un nœud $i \in N$ jusqu'au nœud puits t. Une fonction scalaire équivalente est utilisée dans l'algorithme A*.

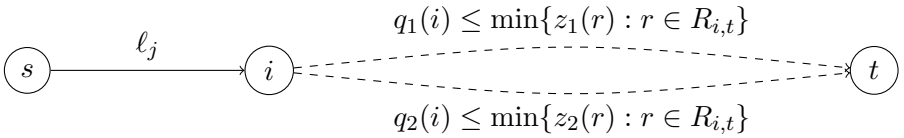

Figure 2.1 – La fonction d'évaluation $q : N \to \mathbb{R}^2_+$ pour deux objectifs.

Définition 2.10. Fonction d'évaluation.
Une fonction d'évaluation $q : N \to \mathbb{R}^p_+$ donne pour tout nœud $i \in N$ des bornes inférieures aux coûts des chemins de i à t :

$$q_k(i) \leq \min\{z_k(r) : r \in R_{i,t}\},\ \forall k \in P,\ \forall i \in N$$

La fonction d'évaluation q est optimiste *[HHW01] s.s.i. :*

$$q_k(i) = \min\{z_k(r) : r \in R_{i,t}\},\ \forall k \in P,\ \forall i \in N$$

La fonction d'évaluation q est admissible (ou monotone[*]*) [MdlC05a] s.s.i. :*

$$q_k(i) + c_k(i,j) \geq q_k(j),\ \forall k \in P,\ \forall (i,j) \in A$$

Notons que si une fonction d'évaluation q est optimiste alors elle est admissible. Pour calculer une fonction d'évaluation q optimiste, il suffit de résoudre le problème de plus court chemin $(R_{\bullet,t})$ de n'importe quel nœud au nœud puits t pour chaque objectif $k \in P$ sur le graphe $G' = (N, A, c_k)$. Dans ce cas, q représente le point idéal pour les chemins $R_{\bullet,t}$. Il est intéressant de calculer une fonction d'évaluation q optimiste dans le cas multi-objectif [SN10] mais pas dans le cas mono-objectif.

Dans le cas mono-objectif un résultat connu est : si la fonction d'évaluation q est admissible alors l'algorithme A* génère au pire un seul et unique label pour chaque nœud du graphe durant toute l'exécution de l'algorithme [MdlC05a]. D'autres propriétés d'une fonction d'évaluation sont définies dans les articles suivants [SI91, DCD95, MdlC05a].

2.2.3.2 Présentation des algorithmes

Tung et Chew [TC88] proposent un algorithme LS pour le problème $2 - \sum$. La fonction de sélection pour un label l_i associé au nœud $i \in N$ est $h(\ell) = \sum_{k \in P} z_k(\ell_i) + q_k(i)$, avec q une fonction d'évaluation optimiste. La règle de coupe est vérifiée pour le label ℓ_i s.s.i. : $\exists \ell_t \in L_t$, t.q. $z(\ell_t) \leq z_k(\ell_i) + q_k(i)$, $\forall k \in P$. Comme la comparaison entre le label ℓ_j et le label ℓ_t utilise uniquement l'inégalité $\leq$ sur chaque objectif et non la dominance de Pareto qui est plus stricte, alors l'ensemble minimal complet des chemins de s à t est calculé car les labels équivalents sont supprimés. Aucune règle de dominance n'est pas utilisée dans l'algorithme et q_k est calculé de manière eoptimiste. La sélection des labels vérifie le principe LS (cf. Définition 2.8, page 55). Tung et Chew [TC92] propose une généralisation de leur précédent algorithme [TC88] au problème MOSP Q-$\sum$ avec un nombre indéfini d'objectifs additifs.

Hallam et al [HHW01] propose une variante de l'algorithme de Tung et Chew [TC92]. La règle de dominance (avec la dominance de Pareto) est utilisée contrairement à l'algorithme de Tung et Chew. Cet algorithme est utilisé dans une application militaire consistant à calculer un chemin minimisant le

∗. On n'utilise ce terme uniquement pour qualifier une fonction d'utilité pas pour qualifier une fonction d'évaluation, voir page 67

temps de parcours ainsi que la probabilité de détection pour un sous-marin militaire. Des contraintes sont ajoutées sur les objectifs afin de satisfaire certaines restrictions sur les coûts. Ces contraintes sont introduites dans l'algorithme par le biais de la règle de coupe, soit $\theta_k \in \mathbb{R}_+$ la borne maximale pour l'objectif k, alors la règle de coupe est aussi vérifiée pour le label $\ell_j \in L_j$ s.s.i. : $\exists k \in P$ t.q. $z_k(\ell_j) + q_k(j) \geq \theta_k$.

Steward et White III [SI91] proposent une généralisation au multi-objectif de l'algorithme d'étiquetage A*. L'algorithme est appelé *multi-objectif A** (MOA*). Il s'agit d'un algorithme de LC/N. La stratégie de sélection n'est pas précisée. La règle de dominance n'est pas utilisée pour chaque label ouvert indépendamment, mais elle est utilisée sur l'ensemble des labels ouverts d'un nœud $\mathfrak{L}_i = \mathfrak{L} \bigcap L_i$. Ainsi un label ouvert $\ell_i \in L_i \bigcap \mathfrak{L}$ est supprimé uniquement si l'ensemble des labels ouverts $L_i \cap \mathfrak{L}$ du nœud i sont dominés par un label $\ell'_i \in L_i \setminus \mathfrak{L}$ déjà propagé du nœud. De même la règle de coupe est vérifié pour un label ouvert $\ell_j \in L_j \bigcap \mathfrak{L}$ s.s.i. : $\exists \ell_t \in L_t,$ t.q. $\forall \ell'_j \in L_j \cap \mathfrak{L}$, $z(\ell_t) \prec_P z(\ell'_j) + q(j)$. Une méthode pour améliorer le calcul de la fonction d'évaluation q est proposée par Dagupta et al [DCD95]. Cette amélioration n'est pas utile si q est optimiste.

Skriver et Andersen [SA00] propose d'ajouter une règle de coupe à l'algorithme LC/N de Brumbaugh-Shier et Smith [BSS89] (stratégie FIFO). Cette règle de coupe est vérifiée pour un label $\ell_j \in L_j$ s.s.i. : $\exists k \in P$ t.q. $y^N_k \leq z_k(\ell_j) + q_k(j) = \max\{z_k(r) : r \in E(R_{s,t})\}$, avec y^N le point nadir du problème $R_{s,t}$. Cette règle de coupe est nécessairement vérifiée pour un nombre moindre de labels que la règle de coupe de Tung et Chew [TC92] car $\forall \ell_t \in L_t, \forall k \in P, y^N_k \geq z_k(\ell_t)$. Néanmoins la vérification de cette règle de coupe demande moins de temps de calcul que pour la règle de coupe de Tung et Chew [TC92]. En effet, chaque label propagé est comparé avec un unique vecteur de coûts, le point nadir. La règle de dominance est employée et un algorithme de MERGE est proposé, les temps de calcul de cet algorithme sont légèrement moindre que ceux de l'algorithme MERGE proposé par Brumbaugh-Smith et Shier [BSS89].

Mandow et Pérez-de-la-Cruz [MdlC05b] proposent un algorithme LS pour le problème MOSP Q-$\sum$ appelé NAMOA*. La fonction de sélection h est une fonction lexicographique sur les coûts des labels augmentés des coûts donnés par la fonction d'évaluation. Ainsi le principe LS (cf. Définition 2.8, page 55) est vérifié si la fonction d'évaluation q est admissible [MdlC05a]. L'algorithme NAMOA* propage alors nécessairement moins de labels que l'algorithme MOA*. Ainsi les auteurs ont montré que NAMOA* utilise nettement moins d'espace mémoire que MOA*. La règle de coupe est vérifiée pour un label $\ell_j \in L_j$ s.s.i. : $\exists \ell_t \in L_t,$ t.q. $z(\ell_t) \prec_P z(\ell_j) + q(j)$. Ainsi les labels équivalents ne sont pas développés, néanmoins ils sont conservés de telle sorte que l'ensemble maximal des chemins efficaces est obtenu à la fin de l'algorithme. La comparaison des temps de calcul entre NAMOA* et MOA* montre que ces deux algorithmes ont des temps de calcul similaires, comme c'est le cas pour les algorithmes LS et LC pour le problème $R_{s,\bullet}$ [GM01, PRS07]. Une comparaison plus approfondie [MMdlCRS10] de ces deux algorithmes montre que l'algorithme NAMOA* est plus rapide que l'algorithme MOA* sur les instances de grande taille.

Raith [Rai10] propose d'utiliser un algorithme d'étiquetage LC/N ou LS dans lequel est ajouté une règle de coupe. Cette règle de coupe est vérifiée pour un label ℓ_j s.s.i. : $\exists \ell_t \in L_t,$ t.q. $z_k(\ell_t) \prec_P z_k(\ell_j)$. L'utilisation de cette règle de coupe permet de réduire de façon importante les temps de calcul des algorithmes LC/N et LS, puisque seuls les chemins efficaces $R_{s,t}$ sont calculés.

Sauvanet et al [SNB11] proposent un algorithme LS pour résoudre le problème 2-$\sum$. La fonction de sélection utilisée h est une fonction lexicographique. La règle de coupe est vérifiée pour un label $\ell_j \in L_j$ s.s.i. : $\exists \ell_t \in L_t,$ t.q. $z(\ell_t) \prec_P z(\ell_j) + q(j)$. Une deuxième règle de coupe est proposée pour un label $\ell_j \in L_j$ s.s.i. : $\exists k \in P$ t.q. $y^N_k < z_k(\ell_j) + q_k(j) = \max\{z_k(r) : r \in E(R_{s,t})\}$, avec y^N le point nadir du problème $R_{s,t}$. Pendant l'initialisation deux chemins $r^a_i, r^b_i \in R_{i,t}$ sont calculés pour chaque nœud $i \in N$, le chemin r^a_i étant construit en optimisant uniquement le premier objectif et r^b_i en optimisant

Auteurs	Objectifs	Type	Ensemble	q optimiste	Citation
Tung et Chew 1988	2-$\sum$	LS	$E(R_{s,t})$	oui	[TC88]
Tung et Chew 1992	Q-$\sum$	LS	$E(R_{s,t})$	oui	[TC92]
Steward et White III 1991	Q-$\sum$	LC/N	$E_M(R_{s,t})$	non	[SI91]
Skriver et Andersen 2000	2-$\sum$	LC/N	$E(R_{s,t})$	oui	[SA00]
Hallam et al 2001	2-$\sum$	LS	$E(R_{s,t})$	oui	[HHW01]
Mandow et Pérez-de-la Cruz 2005	Q-$\sum$	LS	$E_M(R_{s,t})$	non	[MdlC05b]
Raith 2010	2-$\sum$	LC/N,LS	$E_M(R_{s,t})$	–	[Rai10]
Sauvanet et al 2011	2-$\sum$	LS	$E_M(R_{s,t})$	oui	[SNB11]
Machuca et al 2010	—— comparaison MOA* et NAMOA*——-				[MMdlCRS10]

Table 2.3 – Littérature sur les algorithmes d'étiquetage pour le problème MOSP $R_{s,t}$. La colonne "Ensemble" indique l'ensemble de solutions calculé par l'algorithme. La colonne "q optimiste" indique si la fonction d'évaluation q est supposée être optimiste.

le deuxième objectif. Soit $\ell_i \in L_i$ un label propagé durant l'itération de l'algorithme représentant un chemin $r \in R_{i,t}$, deux labels ℓ_t^a, ℓ_t^b représentant les chemins $r \Diamond r_i^a$ et $r \Diamond r_i^b$ respectivement sont ajoutés à L_t.

On cite aussi Harikumar et Kumar [HK96], Mandow et Pérez-de-la Cruz [MdlC03], qui proposent des formalismes généraux pour la recherche heuristique dans les problèmes multi-objectifs.

2.2.4 Conclusion sur les algorithmes d'étiquetage

Pour le problème $R_{s,\bullet}$, on note que les différentes comparaisons de temps de calcul proposées dans la littérature [HPS96, GM01, PRS07, MPRS07, PS08] montrent qu'aucune stratégie de sélection du label courant ne réduit significativement le temps de calcul par rapport à l'algorithme LS proposé par Martins [Mar84a]. On peut voir dans la littérature que les algorithmes de LS et ceux de LC sont équivalents, dans le sens où pour certaines instances les algorithmes LS sont légèrement plus performants et pour d'autres instances les algorithmes LC sont légèrement plus performants [GM01, RE09]. Dans le cadre des algorithmes LS, la structure de données utilisée pour stocker l'ensemble des labels ouverts $\mathfrak{L}$ semble influer de façon importante sur les temps de calcul [PRS07, Rai10]. Pour cette raison les stratégies de sélection employées dans les algorithmes LC/N et LC/L sont plus faciles à mettre en œuvre que les stratégies de sélection des algorithmes LS. La stratégie de sélection du label courant FIFO est parmi les plus efficaces pour les algorithmes LC/N [BSS89, GM01], de même pour les algorithmes LC/L [GM01, PRS07, PS08]. Pour les algorithmes LS, la fonction de sélection somme $h(\ell) = \sum_{k=1}^{p} z(\ell)$ est parmi les plus efficaces [PRS07]. Les algorithmes LC/L semble avoir des temps de calcul légèrement inférieurs à ceux des algorithmes LS ou LC/N [GM01, PRS07, MPRS07, PS08].

Pour le problème MOSP $R_{s,t}$ la seule comparaison existante porte sur les algorithmes NAMOA* et MOA*. Il est donc difficile de distinguer une meilleure stratégie de sélection ou une meilleure règle de coupe. Néanmoins il semble encore possible d'améliorer significativement les temps de calcul pour le problème $R_{s,t}$ contrairement au problème $R_{s,\bullet}$. En effet de nouvelles règles de coupe plus efficaces sont envisageables alors qu'il semble impossible d'améliorer la règle de dominance (avec $\prec_P$) pour le problème Q-$\sum$. Par exemple, l'algorithme NAMOA* [MdlC05b] propose d'utiliser plusieurs fonctions d'évaluation $q^1, \ldots, q^l$ simultanément, et la règle de coupe est testée en utilisant chacune de ces fonctions. Si la règle de coupe est vérifiée avec toutes les fonctions d'évaluation alors le label peut être

supprimé. Une idée similaire a été proposée par Ehrgott et Gandibleux [EG01, EG07] : un ensemble de points bornant l'ensemble des solutions efficaces dans l'espace des objectifs, les points dans notre cas représentant l'addition des vecteurs de coûts des labels et des vecteurs de coûts donnés par les fonctions d'évaluation. Les auteurs de NAMOA*, comme pour la plupart des algorithmes présentés dans le cadre de la recherche heuristique (A*), ne précisent pas comment calculer les fonctions d'évaluations utilisées dans l'algorithme d'étiquetage. De telles fonctions peuvent être calculées a priori en utilisant des informations extérieures au graphe, par exemple la distance à vol d'oiseau entre les nœuds si le graphe représente un réseau routier peut être utilisée pour les calculer. Dans le cadre de notre travail nous ne considérons pas que de telles informations extérieures au graphe puissent être utilisées. On peut alors calculer ces fonctions d'évaluation en optimisant plusieurs sommes pondérées [Hen86, SN10] telles que, pour un sommet $i \in N$, $\{q^1(i), \ldots, q^l(i)\}$ représente un ensemble de points qui domine l'ensemble des chemins de i à t : pour tout chemin $r_{i,t} \in R_{i,t}$ de i à t il existe $d \in \{1, \ldots, l\}$ t.q. $q^t(i) \prec_P z(r_{i,t})$. Des méthodes similaires existent par exemple pour le problème sac à dos [DS10, Jor10].

2.3 Méthode spécifique au multi-objectif pour le problème MOSP

La sous-section 1.1.4 du chapitre 1 présente les méthodes spécifiques au multi-objectif pour résoudre les problèmes MOCO, telles que ϵ-constraint, 2-phase, ranking, dichotomie,... Ces différentes méthodes étant génériques aux problèmes MOCO elles peuvent aussi être appliquées pour le problème MOSP. Cette section présente la littérature concernant l'application de telles méthodes pour résoudre le problème MOSP.

2.3.1 Algorithmes ϵ-contraint pour le problème MOSP avec un objectif additif

Comme nous l'avons vu précédemment, une méthode ϵ-contrainte nécessite de pouvoir intégrer facilement des contraintes sur un objectif (dans le cas bi-objectif). Hors dans le cadre du problème de plus court chemin il est relativement facile d'ajouter une contrainte (borne minimale) sur un objectif $\max$ à minimiser (ou $\min$ à maximisé) puisqu'il suffit de retirer les arcs du graphe qui ne satisfont pas la contrainte. Ainsi l'ajout d'une contrainte à un problème de plus court chemin mono-objectif peut être fait durant l'initialisation et l'algorithme de Dijkstra (par exemple) peut être ensuite appliqué sur le nouveau graphe afin de résoudre ce problème sur-contraint. La méthode ϵ-contraint peut facilement être appliqué au problème MOSP avec un seul objectif additif et un ou plusieurs objectifs bottleneck.

Hansen [Han80] propose le premier algorithme de ce type pour le problème MOSP 1-$\sum$ 1-$\min$. Dans un premier temps, le problème de plus court chemin mono-objectif sur l'objectif additif est résolu, ce qui permet d'initialiser la borne inférieure sur l'objectif bottleneck. Ensuite (1) la borne inférieure est augmentée, (2) les arcs avec un coût supérieur à la borne inférieure sont supprimés puis (3) le problème de plus court chemin sur l'objectif additif est résolu. Les étapes (1), (2) et (3) sont répétées jusqu'à ce que le dernier plus court chemin obtenu ne soit pas efficace.

Martins [Mar84b] propose de généraliser l'algorithme de Hansen pour le problème MOSP bi-objectif avec un objectif quelconque et un objectif bottleneck ($\min$) : 1-$\otimes$ 1-$\min$. Martins suppose qu'il existe un algorithme de plus court chemin pour résoudre le problème mono-objectif sur l'objectif quelconque ($1-\otimes$). Par exemple, si l'objectif quelconque est un objectif additif (1-$\sum$), alors l'algorithme de Dijkstra [Dij59] peut être utilisé. Un deuxième algorithme est présenté par Martins [Mar84a], cet algorithme utilise un algorithme de ranking pour résoudre le problème mono-objectif afin de calculer l'ensemble maximal complet des solutions efficaces.

Berman et al [BEH90] proposent l'introduction d'une fonction de mesure bottleneck qui permet de diminuer la complexité de l'algorithme par rapport à l'algorithme de Hansen dans les graphes denses.

Pelegrin et Fernandez [PF98] propose une variante de l'algorithme de Hansen : un algorithme dit de "*quickest path*" est utilisé pour résoudre le problème de plus court chemin mono-objectif. L'utilisation de cet algorithme permet de générer l'ensemble maximal complet des solutions efficaces. Cette variante proposée est nettement moins performante que l'algorithme de Hansen.

Pinto et al [PBM09] proposent une généralisation de l'algorithme de Hansen pour le problème $1\text{-}\sum\ 1\text{-}\min\ 1\text{-}\max$, où une paire de contraintes est ajoutée sur les deux objectifs bottleneck. Pinto et Pascoal [PP10] proposent d'améliorer l'algorithme précédent [PBM09] en changeant l'ordre selon lequel sont fixés les paires de contraintes sur les deux objectifs bottleneck . Cette amélioration de l'algorithme permet d'éviter de résoudre le problème mono-objectif sur certaines paires de contraintes et ainsi de diminué les temps de calcul sur l'ensemble des instances considérées.

Auteurs	Objectifs	Ensemble	Citation
Hansen 1980	$1\text{-}\sum\ 1\text{-}\min$	$E(R_{s,t})$	[Han80]
Martins 1984	$1\text{-}\otimes\ 1\text{-}\min$	$E_M(R_{s,t})$	[Mar84b]
Berman et al 1990	$1\text{-}\sum\ 1\text{-}\min$	$E(R_{s,t})$	[BEH90]
Pelegrin et Fernandez 1998	$1\text{-}\sum\ 1\text{-}\max$	$E_M(R_{s,t})$	[PF98]
Pinto et al 2009	$1\text{-}\sum\ 1\text{-}\max\ 1\text{-}\min$	$E(R_{s,t})$	[PBM09]
Pinto et Pascoal 2010	$1\text{-}\sum\ 2\text{-}\max$	$E(R_{s,t})$	[PP10]

Table 2.4 – Littérature sur les méthodes ϵ-contrainte pour le problème MOSP. Les objectifs sont définis dans la sous-section 2.1.2. La colonne "Ensemble" indique l'ensemble de solutions calculé par l'algorithme. Les méthodes proposées se limitent aux problèmes MOSP avec un seul objectif additif et plusieurs objectifs bottleneck.

2.3.2 Méthodes de Ranking pour le problème MOSP

Les méthodes de ranking utilisent un algorithme de ranking mono-objectif afin de générer l'ensemble des solutions efficaces. Les algorithmes de ranking construisent les solutions une à une selon leur coût sur l'objectif considéré, de la meilleure des solutions à la moins bonne des solutions.

Climaco et Martins [CM81] proposent, dans le cadre du problème MOSP $2\text{-}\sum$, d'utiliser un algorithme de ranking sur le premier objectif afin générer l'ensemble des chemins $\{r \in R_{s,t}, z_1(r) \leq y_1^N\}$, seuls les chemins efficaces sont conservés. L'algorithme de ranking est arrêté quand la dernière solution $r \in R_{s,t}$ construite est supérieure au point nadir y^N (cf. Définition 1.3, page 13) sur le premier objectif : $z_1(r) > y_1^N$. La borne $z_1(r) \leq y_1^N$ évite de générer l'ensemble des solutions du problème. Une stratégie spécifique est utilisée afin d'éviter au maximum les comparaisons entre les chemins qui servent à vérifier lesquels sont efficaces. Ces mêmes auteurs proposent une seconde méthode [CM82] similaire à cette première méthode [CM81]. Les auteurs démontrent que cette méthode construit l'ensemble maximal des solutions efficaces. Azedo et Martins [AM91] proposent d'adapter un algorithme de Ranking mono-objectif spécifique aux graphes acycliques qui leur permet de générer toutes les solutions selon un ordre lexicographique. La borne de Climaco et Martins est utilisée.

Martins et al [MPRS07] proposent d'utiliser un algorithme de ranking dit de "déviation". Dans la méthode proposée l'algorithme de ranking est modifié afin d'éviter de générer les solutions non efficaces, en ajoutant une règle de dominance basé sur la dominance de Pareto. Les solutions sont générées

selon un ordre lexicographique ou selon une somme pondérée. Ainsi l'algorithme mono-objectif ainsi modifié est à mi-chemin entre un algorithme de ranking mono-objectif et un algorithme LC multi-objectif. L'ensemble maximal des chemins efficaces du nœud source à tous les nœuds est construit. Les temps de calcul de cet algorithme sont semblables à ceux des algorithmes d'étiquetage. Notons que l'algorithme est capable contrairement aux algorithmes d'étiquetage de construire très rapidement un petit ensemble de solutions efficaces.

Paixao et Santos [PS08] propose une méthode de ranking qui utilise le même algorithme de ranking que Martins et al. Contrairement à la méthode de ranking de Martins et al, la méthode proposée calcul uniquement les chemins efficaces d'un nœud source à un nœud puits. Durant l'initialisation de l'algorithme une solution efficace $r^\star \in R_{s,t}$ est calculée en scalarisant le problème MOSP. Pour chaque objectif $k \in P$, la méthode utilise l'algorithme de ranking afin de générer l'ensemble des solutions qui ont un coût sur l'objectif k moindre que la solution initiale $r^\star$: $\{r \in R_{s,t} : z_k(r) \leq z_k(r^\star), r \in E(R_{s,t})\}$. L'union de tous ces ensembles est l'ensemble maximal complet des solutions efficaces : $E_M(R_{s,t}) = \bigcup_{k\in P}\{r \in R_{s,t} : z_k(r) \leq z_k(r^\star), r \in E(R_{s,t})\}$.

Raith et Ehrgott [RE09] proposent d'utiliser un algorithme de ranking dit "*near shortest path*" sur une version scalarisé du problème MOSP. Cet algorithme de ranking est utilisé de telle sorte que seuls les chemins $r \in \mathbb{R}_{s,t}$ tels que $\sum_{k=1}^{2} \lambda_k.z(r) \leq \sum_{k=1}^{2} \lambda_k.y_k^N$ sont générés, avec λ_1, λ_2 des coefficients construit en fonction du point nadir y^N et du point idéal y^I (cf. Définition 1.3, page 13).

Un reproche peut être fait à certaines de ces méthodes [CM81, CM82, AM91, PS08, RE09] est qu'elles permettent uniquement de construire les solutions efficaces du nœud source au nœud puits ($R_{s,t}$) et elles sont comparées à des algorithmes d'étiquetage pour le problème $R_{s,\bullet}$.

Auteurs	Objectifs	Ensemble	Citation
Climaco et Martins 1981	2-$\sum$	$E_M(R_{s,t})$	[CM81]
Climaco et Martins 1982	2-$\sum$	$E_M(R_{s,t})$	[CM82]
Azevedo et Martins 1991	Q-$\sum$	$E_M(R_{s,t})$	[AM91]
Martins et al 2007	Q-$\sum$	$E_M(R_{s,\bullet})$	[MPRS07]
Paixao et Santos 2008	Q-$\sum$	$E_M(R_{s,t})$	[PS08]
Raith et Ehrgott 2009	2-$\sum$	$E_M(R_{s,t})$	[RE09]

Table 2.5 – Littérature sur les méthodes de ranking pour le problème MOSP. La colonne "Ensemble" indique l'ensemble de solutions calculé par l'algorithme.

2.3.3 Méthodes en 2-phases pour le problème de plus court chemin bi-objectif

On note, l'algorithme de White [Whi82] qui propose de calculer un ensemble complet de solutions supportées extrêmes du problème $R_{\bullet,s}$. Cette méthode n'est pas une méthode en 2-phases mais elle correspond à la première phase des méthodes en 2-phases. Il s'agit donc d'une méthode de dichotomie (cf. §1.1.4, page 20). Le problème MOSP est scalarisé à l'aide d'une somme pondérée (cf. Définition 1.5, page 15), puis résolu à l'aide de l'algorithme du simplex. En effet l'ensemble des contraintes du problème $R_{s,t}$ modélisé en variable binaire (cf. Équation 2.2, page 48) est unimodulaire. Alors les solutions efficaces de la relaxation continue ($x_{i,j} \in \mathbb{R}_+$) de ce problème sont des solutions composées uniquement de variables entières [MMO91]. L'opération est répétée pour obtenir un ensemble complet de solutions supportées extrêmes.

Deux méthodes en 2-phases pour le problème de plus court chemin sont proposée dans la littérature : [MMO91] et [RE09]. Les deux méthodes proposent d'utiliser une méthode de dichotomie pour calculer un ensemble complet des solutions supportées extrêmes. Plusieurs articles [HPS96, Skr00, RE09] montrent que l'utilisation d'un simplex pour résoudre la relaxation continue du problème de plus court chemin $R_{s,\bullet}$ n'est pas une méthode efficace pour construire un ensemble complet de solutions supportées extrêmes. Pour cette raison, Raith et al proposent aussi d'utiliser un algorithme d'étiquetage mono-objectif (cf. §2.1.1, page 45) dans la méthode de dichotomie pour résoudre les différentes scalarisation mono-objectif du problème MOSP. Notons aussi un résultat intéressant de Mote et al [MMO91] qui montrent que les chemins supportés sont composés de sous-chemins supportés.

Durant la deuxième phase, des algorithmes sont utilisés pour construire l'ensemble des solutions efficaces qui n'ont pas été construites durant la première phase. Mote et al [MMO91] proposent d'utiliser un algorithme LC/L multi-objectif (cf. §2.2, page 50) où la règle de sélection n'est pas définie et la règle de dominance utilise la dominance de Pareto. Cet algorithme est initialisé avec les solutions supportées extrêmes calculées durant la première phase. Raith et Ehrgott [RE09] proposent d'utiliser trois algorithmes différents pour la deuxième phase : deux algorithmes d'étiquetage (LS et LC/N) multi-objectif et un algorithme de ranking mono-objectif. Les algorithmes d'étiquetage sont initialisés avec l'ensemble des solutions supportées extrêmes construites durant la première phase comme dans la méthode de Mote et al. Tandis que l'algorithme de ranking mono-objectif est utilisé pour résoudre une scalarisation du problème MOSP dans le but de construire l'ensemble des solutions efficaces situées dans chaque aire définie par deux points extrêmes consécutifs (calculés durant la phase 1). On réfère à [PGE08, RE09] pour plus de détail sur l'utilisation de l'algorithme de ranking mono-objectif. Raith et Ehrgott montrent que les temps de calcul de la méthode en deux phases utilisant des algorithmes d'étiquetage multi-objectif est plus efficace que l'utilisation d'un algorithme de ranking pour la deuxième phase de la méthode. Dans le cadre de l'utilisation d'algorithmes d'étiquetage pour la deuxième phase, on peut considérer que l'initialisation d'un algorithme LS, LC/N multi-objectif avec un ensemble de solutions supportées extrêmes est équivalent à un prétraitement dans l'algorithme 2.1 dans le but de supprimer plus de labels durant la vérification de la règle de dominance.

Les méthodes en 2-phases par rapport aux algorithmes d'étiquetage (cf. §2.2, page 50) sont plus rapides pour certaines classes d'instances et plus lentes pour d'autres classes d'instances [MMO91, RE09]. De plus l'utilisation d'une méthode en 2-phases pour un problème avec plus de deux objectifs est difficile [PGE10a].

Auteurs	Objectifs	Ensemble	Citation
Mote et Murthy et Olson 1991	2-$\sum$	$E_M(R_{s,\bullet})$	[MMO91]
Raith et Ehrgott 2009	2-$\sum$	$E_M(R_{s,\bullet})$	[RE09]

Table 2.6 – Littérature sur les méthodes en 2-phases pour le problème MOSP. La colonne "Ensemble" indique l'ensemble de solutions calculé par l'algorithme. Les méthodes proposées concernent uniquement le problème de plus court chemin bi-objectif (deux objectifs additifs).

2.3.4 Conclusion sur les méthodes spécifiques au multi-objectif

Les différentes comparaisons des temps de calcul [HPS96, PS08, RE09], montrent que les méthodes de ranking pour le problème $R_{s,t}$ offrent des temps de calcul aux mieux équivalent à ceux des algorithmes d'étiquetage pour le problème $R_{s,\bullet}$. La méthode de ranking [MPRS07] pour résoudre $R_{s,\bullet}$

n'offre pas des temps de calcul satisfaisant par rapport à l'algorithme LC/L avec stratégie FIFO [GM01]. L'approche ϵ-contrainte se limite aux problèmes 1-$\sum Q$- max. Néanmoins, l'utilisation d'un algorithme MOSP bi-objectif (2-$\sum$) dans une méthode ϵ-contrainte afin de résoudre un problème 2-$\sum Q$- max, semble possible mais n'a pas été testée. Concernant les méthodes en 2-phases, l'utilisation d'un algorithme de ranking pour résoudre la deuxième phase ne semble pas être une piste intéressante [RE09]. Lorsqu'un algorithme d'étiquetage est utilisée pour la deuxième phase alors cette méthode en 2-phase est équivalente à un algorithme d'étiquetage (cf. Algorithme 2.1, page 51) initialisé avec des (quelques) solutions supportées. Cette initialisation réduit les temps de calcul de l'algorithme d'étiquetage [RE09]. Néanmoins il est difficile d'appliquer un méthode en 2-phase avec plus de deux objectifs.

Compte tenu de la littérature, les algorithmes d'étiquetage, contrairement aux méthodes spécifiques aux multi-objectif, constituent les méthodes les plus génériques (s'adaptent facilement aux différents problèmes) et les plus efficaces pour résoudre les différents problèmes MOSP.

2.4 Intégration a priori des préférences dans un algorithme exact

Dans cette section nous abordons la recherche d'une solution préférée d'un problème MOSP à l'aide d'un algorithme exact. La sous-section 2.4.1 présente la problématique général de l'intégration des préférences ainsi que les méthodes proposées dans la littérature pour répondre à la problématique général. La sous-section 2.4.2 présente le cadre de l'intégration des préférences modélisées sous la forme d'une fonction d'utilité. La sous-section 2.4.3 présente la littérature concernant l'optimisation d'une fonction d'utilité dans un problème MOSP.

2.4.1 Problématique général de l'intégration a priori des préférences

Afin d'intégrer a priori les préférences d'un décideur dans un algorithme pour le problème MOSP, ces préférences doivent être être définie à l'aide d'un modèle de préférence. Ce modèle de préférence peut être une fonction d'utilité, une modèle de surclassement, des contraintes sur les objectifs, ... (cf. §1.2, page 22). Si un modèle de préférences est connu a priori, alors il n'est pas nécessaire de calculer l'ensemble des solutions efficaces du problème (cf. §1.3.3, page 38). Nous avons vu précédemment que le nombre de solutions efficaces d'un problème MOSP peut être important (sous-section 2.1.2). Alors l'intégration a priori des préférences dans un problème MOSP doit permettre d'éviter de construire l'ensemble des solutions efficaces et ainsi réduire le temps de calcul de façon importante.

Perny et Spanjaard [PS05] proposent un cadre général pour intégrer des préférences dans les algorithmes d'étiquetage. Les auteurs utilisent une relation de préférence ($\precsim_{DM}$,$\prec_{DM}$,$\sim_{DM}$) qui est définie par un modèle de préférence non déterminé. Ils proposent un algorithme LC/N (voir algorithme 2.1) où les sommets sont sélectionnés les uns à la suite des autres (comme dans l'algorithme de Bellman [Bel58] et celui de Corley et Moon [CM85]). La règle de dominance utilise la dominance $\prec_{DM}$: un label l_i^a associé à un nœud $i \in N$ est supprimé si il existe un autre label l_i^b associé au nœud i tel que $l_i^b \prec_{DM} l_i^a$. L'algorithme permet d'obtenir une solution non $\prec_{DM}$-dominé s.s.i. la relation de préférence $\prec_{DM}$ vérifie la règle dite "d'indépendance" : $\forall i, j, d \in N$ trois nœuds quelconque, $\forall r^a, r^b \in R_{i,j}$, $\forall r^c \in R_{j,d}$, $r^a \prec_{DM} r^b \Longrightarrow r^a \Diamond r^c \prec_{DM} r^b \Diamond r^c$. Quand la relation de préférence ($\prec_{DM}$) ne respecte cette règle, les auteurs proposent d'utiliser une relation de dominance $\prec'_{DM}$, qui approxime $\prec_{DM}$ (chaque solution non $\prec_{DM}$-dominée est aussi non $\prec'_{DM}$-dominée) et qui respecte la règle d'indépendance.

2.4.2 Intégration d'une fonction d'utilité dans un algorithme d'étiquetage

La grande majorité de la littérature sur l'intégration des préférences dans un algorithme MOSP considère que les préférences du décideur sont modélisées sous la forme d'une fonction d'utilité U (cf. §1.2.3, page 26). Dans ce cadre, on utilise la notion d'optimalité plutôt que la notion d'efficacité ou de non dominance, les chemins étant évalués sur une échelle unique. Ainsi, au lieu du principe d'efficacité (cf. Définition 2.4, page 52), on peut utiliser le principe d'optimalité de Bellman, également appelé *principe d'optimalité fort* (cf. Définition 2.11, de la présente page).

Définition 2.11. Principe d'optimalité fort [Bel57].
Un chemin optimal est composé uniquement de sous-chemins optimaux.

Un résultat connu [CMM90, Hen86] est que, si la fonction d'utilité U et le vecteur d'opérateurs binaires $\otimes$ vérifie la propriété de monotonie alors le problème multi-objectif peut être réduit à un problème mono-objectif.

Définition 2.12. Monotonie de $(U, \otimes)$ [CMM90].
Une fonction d'utilité $U : \mathbb{R}_+^p \to \mathbb{R}_+$ devant être minimisée et un vecteur d'opérateurs binaires $\otimes = (\otimes_1, \ldots, \otimes_p)$ vérifient la propriété de monotonie s.s.i. :

$$U(y^a) \leq U(y^b) \Longrightarrow U(y^a \otimes y^c) \leq U(y^b \otimes y^c),\ \forall y^a, y^b, y^c \in \mathbb{R}_+^p$$

La notion de monotonie d'une fonction d'utilité est similaire à la notion d'indépendance préférentielle (cf. Définition 1.13, page 30) d'une relation de préférences, voir aussi Perny et Spanjaard [PS05]. Si $(U, \otimes)$ est monotone alors l'algorithme de LS peut être défini de la manière suivante :

- La fonction de sélection du label courant $h(\ell) = U(\ell)$.
- La notion de dominance $\prec$ est définie telle que pour deux labels $\ell_i, \ell'_i \in L_i$ associés avec le nœud $i \in N$, ℓ'_i domine ℓ_i ssi $U(\ell'_i) \leq U(\ell_i)$

L'algorithme est alors équivalent à l'algorithme de Dijkstra et un seul et unique label est propagé pour chaque nœud. Ainsi le problème d'optimiser cette fonction d'utilité correspond à un problème mono-objectif de complexité polynomial ou l'unique objectif est $z'(r) = U(z(r))$.

Le cas le plus courant de monotonie est celui où chaque opérateur binaire est une somme $\otimes = (+, \ldots, +)$ et la fonction d'utilité est une somme pondérée $U(y) = \varphi_\lambda(y)$, avec $\lambda \in \mathbb{R}^p$. Carraway et al [CMM90] donnent un autre cas de monotonie quand les coûts sont positifs : l'opérateur binaire est une multiplication $\otimes = (\times, \ldots, \times)$ et la fonction d'utilité est le produit des coûts $U(y) = \prod_{k \in P} y_k^{\alpha_k}$, avec $\alpha \in \mathbb{N}_+^p$.

Chaque algorithme présenté dans la sous-section (2.4.3) suivante, optimise une fonction d'utilité U (avec $\otimes = (+, \ldots, +)$) ne vérifiant pas la monotonie. De plus, toutes les fonctions d'utilité U utilisée dans la littérature sont croissantes sur chacun des objectifs : $y^a \prec_P y^b \Longrightarrow U(y^a) \leq U(y^b), \forall y^a, y^b \in \mathbb{R}_+^p$ (cf. §1.2, page 22). Ces algorithmes sont tous proposés pour le problème MOSP $(R_{s,t})$ d'un nœud source $s \in N$ à un nœud puits $t \in N$. Pour cette raison certains de ces algorithmes utilise une fonction d'évaluation $q : N \to \mathbb{R}_+^p$ (cf. Définition 2.10, page 59) qui donne pour chaque sommet $i \in N$ des bornes inférieures aux coûts des chemins de i à t : $\forall k \in P$, $q_k(i) \leq \min\{z_k(r) : r \in R_{i,t}\}$. De plus, on définit une fonction vectorielle $q^\lambda : N \to \mathbb{R}_+$ qui correspond au vecteur de coûts des chemins $r_{i,t}^\lambda$, $i \in N$ optimisant la somme pondérée $\varphi_\lambda : \mathbb{R}_+^p \to \mathbb{R}_+ : \forall i \in N$, $q^\lambda(i) = z(r_{i,t}^\lambda)$, avec $r_{i,t}^\lambda = \arg\min_{r \in R_{i,t}} \varphi_\lambda(z(r))$. La somme pondérée φ_λ étant une fonction d'utilité, alors la propriété de monotonie est vérifiée pour l'optimisation de φ_λ dans le problème MOSP Q-$\sum$ $(\otimes = (+, \ldots, +))$. Dans ce cas, pour calculer q^λ il suffit de résoudre le problème de plus court chemin mono-objectif $R_{\bullet,t}$ sur le graphe $G' = (N, A, c^\lambda)$ où $c^\lambda(a) = \varphi_\lambda(c(a)) = \sum_{k=1}^p \lambda_k . c_k(a), \forall a \in A$.

2.4.3 Algorithmes pour optimiser une fonction d'utilité dans un problème MOSP

À l'exception de l'algorithme de Reinhardt et Pisinger [RP11], tous les algorithmes présentés dans cette section concernent le problème MOSP Q-$\sum$

Carraway et al [CMM90] proposent une équation de programmation dynamique (DP) pour l'optimisation d'une fonction quelconque U dans un graphe acyclique. Cette équation définit un algorithme LC/N. Le graphe étant acyclique, chaque nœud est sélectionné une seule et unique fois. Les auteurs proposent la notion de *préférence locale* $\prec_i, i \in N$, qui est utilisée dans des règles de dominance appliquées localement : un label l_i^a associé au nœud i est supprimé si il existe un autre label l_i^b associé au nœud i tel que $l_i^b \prec_i l_i^a$. La construction de la préférence local $\prec_i, i \in N$ n'est précisée.

White et al [WSC92] proposent un algorithme de LC/N (U*) basé sur la connaissance a priori d'un ensemble de fonctions d'évaluation $q^1, \ldots, q^l : N \to \mathbb{R}_+$ pour le problème $R_{s,t}$ avec U une fonction d'utilité quelconque. Parmi ces fonctions il en existe au moins une, q^t $(1 \leq t \leq l)$ qui permette d'obtenir un chemin optimal $r^\star$ sur chacun des nœuds du chemin : $\forall i \in r^\star$, $q^t(i) = z(r_i^\star)$, $r_i^\star$ désignant le sous-chemin de i à t du chemin optimal $r^\star \in R_{s,t}$. Nembhard et White [NW99] proposent deux algorithmes BU* et DU* qui sont des améliorations de l'algorithme de White et al [WSC92].

Henig [Hen86] propose une méthode pour optimiser une fonction d'utilité U convexe pour le problème MOSP 2-$\sum$. Cette méthode détermine les deux solutions supportées extrêmes adjacentes $r^a, r^b \in R_{s,t}$ qui encadre le point $y^\star \in \mathbb{R}_+^p$ des solutions optimales (selon U) dans l'espace des objectifs : $z_1(r^a) \leq y_1^\star$, $y_2^\star \leq z_2(r^a)$ et $z_2(r^b) \leq y_2^\star$, $y_1^\star \leq z_1(r^b)$. Une méthode de dichotomie est utilisée (cf. §1.1.4.2, page 21) pour déterminer ces solutions supportées extrêmes. La fonction d'utilité U étant strictement convexe alors les solutions optimales sont nécessairement des solutions supportées extrêmes. Alors r^a ou r^b est une solution optimale.

Paixao et al [PMRS03] proposent un algorithme LC/L pour optimiser la distance euclidienne $||y||_2 = \sqrt{(y_1)^2 + \cdots + (y_p)^2}$. Une somme pondérée $\varphi_\lambda(y)$ avec $\lambda_k = 1/\sqrt{p}, \forall k \in P$ est utilisée pour calculer la fonction vectorielle q^λ. Cette somme pondérée borne inférieurement la distance euclidienne $\varphi_\lambda(y) \leq ||y||_2, \forall y \in \mathbb{R}_+^p$. Deux règles de coupe sont utilisées et la fonction d'évaluation q est optimiste. Un label ℓ_j vérifie la première règle de coupe s.s.i. $||q^\lambda(s)||_2 \leq ||\ell_j + q_j||_2$, $q^\lambda(s)$ représentant le vecteur de coûts d'un chemin de s à t optimal selon φ_λ. La deuxième règle utilise la somme pondérée φ_λ, elle permet de supprimer plus de labels que la première règle de coupe mais son utilisation nécessite plus de temps de calcul. La sélection du label courant suit la stratégie FIFO. Les temps de calcul de l'algorithme sont comparés à un algorithme LC pour le problème $R_{s,\bullet}$. Un algorithme de ranking est aussi proposé, les chemins sont générés selon l'objectif donné par la somme pondérée des coûts $\varphi_\lambda(c(i,j)), \forall (i,j) \in A$. L'algorithme s'arrête quand le dernier chemin généré $r_{s,t} \in R_{s,t}$ vérifie la règle d'arrêt suivante : $||r_{s,t}^\star||_2 \leq \varphi_\lambda(z(r))$, avec $r_{s,t}^\star \in R_{s,t}$ le meilleur chemin de s à t connu.

Galand et Perny [GP06b] proposent un algorithme LS appelé BCA* (voir aussi [FP00]) pour l'optimisation d'une norme de Tchebychev Υ_α (cf. Définition 1.8, page 17) par rapport à un point de référence $y^\star$. Les poids $\alpha_1, \ldots, \alpha_p$ sont calculés en fonction du point nadir y^N et du point idéal y^I. La fonction de sélection pour un label $\ell_i \in \mathfrak{L}_i$ associé au nœud $i \in N$ est $h(\ell_i) = \Upsilon_\alpha(z(\ell_i) + q(i))$. Un label $\ell_i \in L_i$, associé au nœud $i \in N$, vérifie la règle de coupe s.s.i. son coût augmenté avec la fonction d'évaluation q et agrégé par la fonction Υ_α est meilleur que celui du meilleur chemin connu $r_{s,t}^\star \in R_{s,t}$: $\Upsilon_\alpha(z(r_{s,t}^\star)) \leq \Upsilon_\alpha(z(\ell_i) + q(i))$.

Climaco et al [CCP06] proposent un algorithme LS pour optimiser une norme de Tchebychev Υ_α par rapport à un point de référence $y^\star$ et avec un vecteur de poids $\alpha \in \mathbb{R}_+^p$ définis a priori par le décideur. La fonction de sélection h est une fonction lexicographique. La particularité de cet algorithme est qu'il calcule les K-meilleurs chemins selon Υ_α. Un label $\ell_j \in L_j$ vérifie la règle de coupe s.s.i.

$\Upsilon_\alpha(\ell_t^K) \leq \Upsilon_\alpha(\ell_j)$ avec ℓ_t^K le K-ème meilleur label associé au nœud t connu. Afin de construire les K-meilleurs chemins, la règle de dominance (voir algorithme 2.1) n'est pas utilisée, car des chemins non efficaces peuvent faire partis des K-meilleurs chemins.

Galand et Spanjaard [GS07] proposent un algorithme LS basé sur NAMOA* pour optimiser une fonction d'agrégation OWA Ψ_α^{owa} (cf. Définition 1.14, page 32) convexe. La fonction d'évaluation q (cf. Définition 2.10, page 59) et la fonction vectorielle q^λ (cf. §2.4.2, page 67) sont utilisées, avec un vecteur de poids $\lambda = (\frac{1}{p}, \ldots, \frac{1}{p})$. La somme pondérée φ_λ donne une borne inférieure de la fonction OWA Ψ_α^{owa} : $\forall y \in \mathbb{R}_+^p$, $\varphi_\lambda(y) \leq \Psi_\alpha^{owa}(y)$. Pour un label quelconque ℓ_i associé à un nœud $i \in N$, la fonction $e(\ell_i) \in \mathbb{R}_+^p$ est définie par rapport à q et q^λ :

$$\begin{array}{rlll} e(\ell_i) = & \min & \Psi_\alpha^{owa}(y) & \\ & \text{s.c.} & \sum_{k=1}^p y_k \geq \sum_{k=1}^p (z_k(\ell_i) + q_k^\lambda(i)) & \\ & & y_k \geq z_k(\ell_i) + q_k(i) & \forall k \in P \\ & & y \in \mathbb{R}_+^p & \end{array} \tag{2.10}$$

Le problème 2.10 peut être résolu en résolvant 2^p problèmes linéaires. Les auteurs proposent deux définitions pour la règle de coupe : (a) le label ℓ_i est supprimé si $\min_{\ell_t \in L_t} U(\ell_t) \leq e(\ell_i)$, ou (b) le label ℓ_i est supprimé si $\min_{\ell_t \in L_t} U(\ell_t) \leq U(z(\ell_i) + q(i))$, L_t est l'ensemble des labels associés au nœud t. Les temps de calcul montrent que l'utilisation de la règle de coupe (a) est nettement plus efficace que la règle de coupe (b).

Galand et Perny [GP07] proposent un algorithme LS pour optimiser une fonction d'utilité U composée d'une unique fonction d'utilité partielle u ($u = u_1 = \cdots = u_k$) convexe et d'une intégrale de Choquet convexe (cf. §1.2.5, page 32) : $U(y) = C_\mu(u(y_1), \ldots, u(y_p))$. Le modèle de préférence est proposé dans un cadre d'optimisation robustesse et non dans un cadre d'aide à la décision multi-critère. La fonction vectorielle q^λ est définie telle que φ_λ borne inférieurement U : $\varphi_\lambda(y) = \sum_{k \in P} \lambda_k . y_k \leq U(y), \forall y \in \mathbb{R}_+^p$. Pour calculer un tel vecteur de poids λ, les auteurs proposent d'utiliser l'indice de Shapley ϕ (cf. Définition 1.20, page 35) de l'intégrale de Choquet C_μ : $\lambda_k = \phi_k, \forall k \in P$. L'évaluation d'un label $\ell_j \in L_j$ est définie, par rapport à q et q^λ, telle que $e(\ell_i) = \max\{U(z(\ell_i) + q(i)), \sum_{k=1}^p \lambda_k . (z_k(\ell_i) + q_k^\lambda(i))\}$. La fonction de sélection h des labels est la fonction e : $h = e$. La règle de coupe est définie telle que : un label ℓ_i, $i \in N$ est supprimé s.s.i. $\min_{\ell_t \in L_t} U(\ell_t) \leq e(\ell_i)$. Galand et al [GPS10] propose un algorithme identique à l'algorithme de Galand et Perny, avec une fonction d'utilité identique. Les auteurs proposent aussi d'utiliser un algorithme de ranking mono-objectif, les chemins sont générés selon la somme pondérée φ_λ (la monotonie est vérifiée). L'algorithme s'arrête lorsque la dernière solution générée r vérifie une règle d'arrêt : $U(r^\star) \leq u(\varphi_\lambda(z(r)))$, avec $r^\star$ la meilleure solution connue selon U.

Sauvanet et Néron [SN10] propose un algorithme LS basé sur BCA* pour optimiser une norme de Tchebychev Υ_α par rapport à un point de référence $y^\star$ défini a priori : $\Upsilon_\alpha(y) = \max_{k=1}^2 \alpha_k . |y_k^\star - y_k|$ avec $\alpha_1, \alpha_2 \in \mathbb{R}_+$. La fonction d'évaluation q est optimiste (cf. Définition 2.10, page 59). La fonction vectorielle q^λ est calculée avec $\lambda = (1/2, 1/2)$. L'évaluation d'un label $\ell_j \in L_j$ est $e(\ell_j) = \min\{\Upsilon_\alpha(z_1(\ell_j) + q_1, z_2(\ell_j) + q_2^\lambda), \Upsilon_\alpha(z_1(\ell_j) + q_1^\lambda, z_2(\ell_j) + q_2)\}$. Un label ℓ_j vérifie la règle de coupe s.s.i. $\min_{\ell_t \in L_t} \Upsilon_\alpha(\ell_t) \leq e(\ell_j)$. La fonction de sélection h est définie par rapport à q et q^λ.

Reinhardt et Pisinger [RP11] proposent trois règles de dominance pour un algorithme de LC pour le problème MOSP Q-$\sum$. La fonction d'utilité est définie comme la somme des fonctions d'utilité partielles : $U(y) = \sum_{k=1}^p u_k(y_k)$, $y \in \mathbb{R}^p$. Les fonctions d'utilité partielles sont strictement croissantes. Deux dominances $\prec_{RP1}, \prec_{RP2}$ sont proposées. Un label $\ell_i \in L_i$ est supprimé si il existe un autre label $\ell_i' \in L_i$ associé au même nœud $i \in N$ t.q. $\ell_i \prec_{RP1} \ell_i' \Leftrightarrow u(z(\ell_i)) \prec_P u(z(\ell_i))$ ou $\ell_i \prec_{RP2} \ell_i' \Leftrightarrow \sum_{k=1}^p \delta_k . z_k(\ell_i) < \sum_{k=1}^p \delta_k . z_k(\ell_i')$. Avec $\delta_k = \min\{\dot{u}_k(y) : y \in \mathbb{R}\}$ la plus petite dérivée de u_k si $z_k(\ell_i) < z_k(\ell_i')$, $\delta_k = 1$ si $z_k(\ell_i) = z_k(\ell_i')$ et $\delta_k = \max\{\dot{u}_k(y) : y \in \mathbb{R}\}$ la plus grande dérivée

Auteurs	Objectifs	Fct. utilité	Ensemble	Type	Citation
Henig 1986	2-$\sum$	U, convexe	$R_{s,t}$	DP	[Hen86]
Carraway et al 1990	Q-$\otimes$	U	$R_{s,t}$	DP	[CMM90]
White et al 1992	Q-$\otimes$	U	$R_{s,t}$	LC/N	[WSC92]
Nembhard et White 1999	—	U	$R_{s,t}$	LC/N	[NW99]
Paixao et al 2003	Q-$\sum$	$\|\|.\|\|_2$	$R_{s,t}$	LC,ranking	[PMRS03]
Climaco et al 2006	Q-$\sum$	Υ_α	$R_{s,t}$	LS	[CCP06]
Galand et Perny 2006	Q-$\sum$	Υ_α	$R_{s,t}$	LS, ranking	[GP06a]
Galand et Spanjaard 2007	Q-$\sum$	Ψ_α^{owa}	$R_{s,t}$	LS	[GS07]
Galand et Perny 2007	Q-$\sum$	C_μ,u	$R_{s,t}$	LS, ranking	[GP07]
Sauvanet et Néron 2010	2-$\sum$	Υ_α	$R_{s,t}$	LS	[SN10]
Galand et al 2010	Q-$\sum$	C_μ	$R_{s,t}$	LS, ranking	[GPS10]
Reinhardt and Pisinger 2011	Q-$\sum$	$\sum_{k=1}^{p} u_k$	$R_{s,t}$	LC/N	[RP11]
Reinhardt and Pisinger 2011	1-$\sum$ Q-max	$\sum_{k=1}^{p} u_k$	$R_{s,t}$	LC/N	[RP11]

Table 2.7 – Littérature sur l'optimisation d'une fonction d'utilité dans un problème MOSP. La colonne "Ensemble" indique l'ensemble de solutions calculé par l'algorithme. La colonne "Préférences" indique la fonction d'utilité optimisée, U désigne une fonction d'utilité quelconque, C_μ désigne l'agrégation des objectifs par l'intégrale de Choquet, Υ_α désigne l'agrégation des objectifs par la norme de Tchebychev, $||.||_2$ désigne l'agrégation des objectifs par une norme euclidenne, Ψ_α^{owa} désigne l'agrégation des objectifs par la fonction OWA.

de u_k si $z_k(\ell'_i) < z_k(\ell_i)$. Si les fonctions d'utilité partielles $u_1, \ldots, u_{p-1}$ sont des fonctions d'identité ($u_k(y_k) = y_k, \forall k \in \{1, \ldots, p-1\}, \forall y_k \in \mathbb{R}$) et u_p est une fonction strictement croissante, alors une troisième dominance est proposée $\prec_{RP3}$ est proposée. Soit $\beta = \max\{u_p(y^a + y^c) - u(y^a) - [u(y^b + y^c) - u(y^b)] : y^a, y^b, y^b \in \mathbb{R}\}$ la différence de variation maximum entre deux valeurs y^a, y^b pour une même augmentation y^c. Un label $\ell_i \in L_i$ est supprimé si il existe un autre label $\ell'_i \in L_i$ associé au même nœud $i \in N$ t.q. $\ell_i \prec_{RP3} \ell'_i \Leftrightarrow U(z(\ell_i)) + \beta < U(z(\ell_i))$.

Reinhardt et Pisinger [RP11] proposent aussi une dominance $\prec_{RP4}$ pour le problème 1-$\sum$ Q-max. Les fonctions d'utilité partielles sont des fonctions d'identité : $u_k(y_k) = y_k, \forall k \in P, \forall y_k \in \mathbb{R}$, Un label $\ell_i \in L_i$ est supprimé si il existe un autre label $\ell'_i \in L_i$ associé au même nœud $i \in N$ t.q. $\ell_i \prec_{RP4} \ell'_i \Leftrightarrow z_1(\ell_i) - z_1(\ell'_i) + \sum_{k=2}^{p} \min\{0, z_k(\ell_i) - z_k(\ell_i)\} < 0$. Cette dominance indique que les coûts des labels sur les objectifs bottleneck n'est pas pris en compte si ce coût est au désavantage du label que l'on souhaite supprimer. On voit que, comme pour la dominance $\prec_{GBR}$ (cf. Définition 2.6, page 54), les coûts de labels sur des objectifs bottleneck ne peuvent pas être pris en compte pour la suppression d'un label.

2.5 Conclusion

Concernant la littérature sur les méthodes pour calculer l'ensemble des solutions efficaces d'un problème MOSP (cf. Sections 2.2 et 2.3) quelques remarques générales peuvent être formulées. Dans cet état de l'art nous avons proposé de différencier les méthodes pour résoudre le problème $R_{s,t}$ et les méthodes pour résoudre le problème $R_{s,\bullet}$, cette différenciation n'est en général pas faite dans la littérature. La résolution d'un problème de plus courts chemins d'un nœud à tous les nœuds ($R_{s,\bullet}$) permet de résoudre

n problèmes de plus court chemin d'un nœud à un nœud ($R_{s,t}$). Ainsi il faut prendre en compte cette différence entre ces deux problèmes ($R_{s,t}$ et $R_{s,\bullet}$) quand on compare une méthode pour le problème $R_{s,t}$ et une méthode pour le problème $R_{s,\bullet}$. On remarque aussi que dans la littérature, les travaux provenant de l'intelligence artificielle (tels que l'algorithme NAMOA*) et ceux venus de la recherche opérationnelle (tel que l'algorithme de Tung et Chew) en général s'ignorent mutuellement.

Concernant la littérature sur les méthodes pour optimiser une fonction d'utilité (cf. §2.4, page 66), aucun algorithme proposé ne permet de résoudre le problème $R_{s,\bullet}$ (cf. Table 2.7, page suivante). Cette problématique est abordée dans le chapitre 4. On note aussi que la plupart des algorithmes optimisent des fonctions concaves ou convexes : norme euclidienne, norme de Tchebychev, OWA, ..., qui sont pour la plupart des cas particuliers de l'intégrale de Choquet. Dans le chapitre 3 nous proposons un algorithme qui optimise une fonction d'utilité MAUT basée sur l'intégrale de Choquet quelconque pour le problème $R_{s,t}$. Pour finir on note qu'aucun algorithme n'est proposé pour optimiser une fonction d'utilité dans un problème (Q-$\max$ Q-$\sum$) avec plusieurs objectifs bottleneck. Cette problématique est aussi traitée dans le chapitre 4.

CHAPITRE 3

Optimisation d'une fonction d'utilité basée sur l'intégrale de Choquet pour le problème MOSP nœud à nœud

On s'intéresse dans ce chapitre à la minimisation d'une fonction d'utilité multi-critère U dans un problème de plus court chemin ($R_{s,t}$) entre un nœud source $s \in N$ et un nœud puits $t \in N$. On considère l'utilisation d'un algorithme d'étiquetage pour résoudre ce problème, ce type d'algorithme étant considéré comme efficace pour ce problème de plus court chemin (cf. §2.5, page 70). Dans le cadre du problème $R_{s,t}$ les méthodes proposées dans la littérature pour intégrer une fonction d'utilité U consistent à comparer les labels à une solution connue du problème. En section 3.1 on propose une règle de coupe générale qui généralise ces différentes approches. Cette règle de coupe nécessite le calcul d'une borne inférieure pour chaque label. Afin de supprimer un maximum de labels, cette borne inférieure doit prendre en compte le chemin qui reste à parcourir pour atteindre le nœud puits. La borne inférieure doit être calculée en résolvant (durant l'initialisation de l'algorithme d'étiquetage) un ou plusieurs problèmes de plus court chemin mono-objectif.

On s'intéresse tout d'abord, en section 3.2, à l'optimisation d'une fonction d'utilité basée sur l'intégrale de Choquet dans le problème MOSP avec uniquement des objectifs additifs (Q-$\sum$). Pour calculer les bornes inférieures des labels on propose différentes méthodes pour construire une somme pondérée qui borne inférieurement l'intégrale de Choquet. On propose notamment (cf. §3.2.2, page 85), l'utilisation de bornes inférieures et supérieures sur les objectifs afin d'améliorer la borne calculée par la somme pondérée. Des méthodes sont aussi proposées (cf. §3.2.4, page 95) pour calculer une somme pondérée qui borne inférieurement une fonction d'utilité constituée de l'intégrale de Choquet et de fonctions d'utilité partielles linéaires par morceaux. Certaines de ces méthodes ont fait l'objet d'une publication [FGL11a].

3.1 Règles de coupe

Notre ambition est d'optimiser une fonction d'utilité U dans un problème MOSP $R_{s,t}$. On suppose que le graphe est connexe. On utilise l'algorithme d'étiquetage général (cf. Définition 2.1, page 51) avec une règle de coupe (cf. Définition 2.9, page 59). On suppose que l'instance du problème MOSP, composée du graphe $G = (N, A)$ et de p fonctions objectif $(z_1, \ldots, z_p)$, respecte les conditions de terminaison des algorithmes d'étiquetage (cf. Théorème 2.5, page 55) : les coûts sur les arcs sont positifs, $\forall a \in A$, $(\mathbf{0}_1, \ldots, \mathbf{0}_p) \prec_P c(a)$. Les p fonctions objectif (P l'ensemble des objectifs) sont définies par p opérateurs binaires $(\otimes_1, \ldots, \otimes_p)$. Un opérateur binaire $\otimes_k$, $k \in P$, peut par exemple être une addition $\otimes_k = +$, une multiplication $\otimes_k = \times$, une fonction maximale $\otimes_k = \max$, une fonction minimale $\otimes_k = \min$ (cf. §2.1.2, page 48). Soient $\mathbf{\Omega} = \Omega_1 \times \cdots \times \Omega_p \subset \mathbb{R}^p$ l'espace des objectifs, $y \in \mathbf{\Omega}$ un vecteur de coûts et $\xi \subset \mathbb{R}$ l'échelle de satisfaction commune des objectifs, $U : \mathbf{\Omega} \to \xi$.

Exemple 3.1.
Considérons un graphe $G = (N, A, c)$ défini sur la figure 3.1 et deux fonctions objectif additives $P = \{1, 2\}$. On s'intéresse au problème MOSP *(2-$\sum$) d'un nœud source $s = 1$ à un nœud puits $t = 3$ ($R_{1,3}$). Les coûts sur les arcs sont positifs ou nuls $\mathbf{\Omega} = \Omega_1 \times \Omega_2 = \mathbb{R}^2_+$. Les solutions efficaces et non efficaces du*

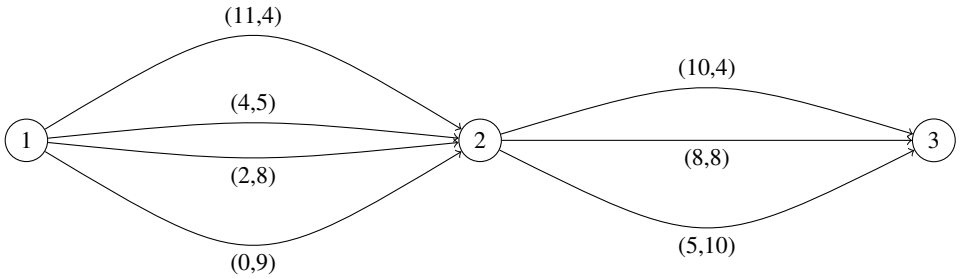

Figure 3.1 – Un graphe $G = (N, A, c)$, les nœuds $N = \{1, 2, 3\}$. Deux coûts sont définis sur les arcs du graphe, $c(a) \in \mathbf{\Omega}$. Le graphe contient des arcs multiples afin de simplifier l'exemple.

problème sont définies sur la figure 3.2. On considère la minimisation d'une fonction d'utilité $U = C_\mu$ qui est définie par une intégrale de Choquet. On ne considère pas de fonction d'utilité partielle afin de simplifier l'exemple. La fonction de capacité (cf. Définition 1.16, page 32) est définie telle que : $\mu(\emptyset) = 0$, $\mu(\{1\}) = 0.7$, $\mu(\{2\}) = 0.7$ $\mu(\{1, 2\}) = 1$. Cette fonction de capacité est submodulaire *(cf. Définition 1.18, page 34). La solution optimale de ce problème selon C_μ est définie sur la figure 3.3.*

Afin de prendre en compte la fonction d'utilité U dans l'algorithme d'étiquetage, la définition 3.1 ci-dessous propose une règle de coupe (cf. Définition 2.9, page 59) générale. Soit $r^*_{s,t}$ le meilleur (selon U) chemin connu entre s et t au cours de l'algorithme. Ce chemin n'est pas nécessairement le chemin optimal (selon U). Ce chemin peut être construit durant l'exécution de l'algorithme d'étiquetage (alors $U(z(r^*_{s,t})) = \min\{U(z(l_t)) : l_t \in L_t\}$) ou a priori de l'algorithme d'étiquetage en résolvant un problème de plus court chemin mono-objectif. Dans la suite de cette section on présente différentes définitions permettant de calculer, pour chaque nœud $i \in N$ et chaque label $\ell_i \in L_i$, une borne inférieure notée BORNE_INF(ℓ_i) du plus court chemin pouvant être obtenue en développant ℓ_i. La définition 3.1 présente l'utilisation d'une borne inférieure dans une règle de coupe.

Définition 3.1. Règle de coupe générale.
Soit $U : \mathbf{\Omega} \to \xi$ une fonction d'utilité croissante (cf. Définition 1.9, page 18). Soit ℓ_i un label associé au

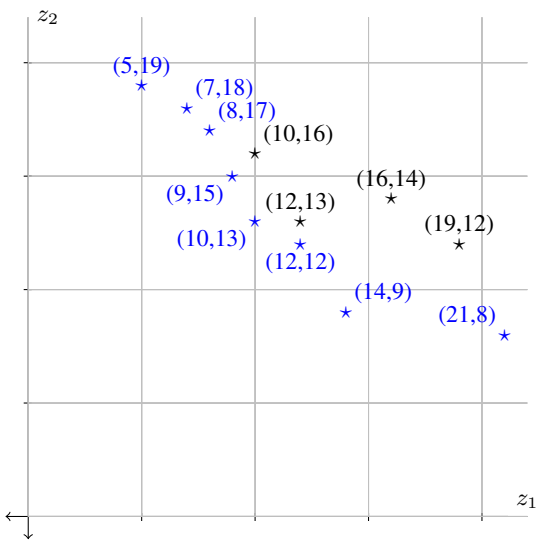

Figure 3.2 – Espace des objectifs, les points des solutions efficaces sont en bleus et les points des solutions non efficaces sont en noires.

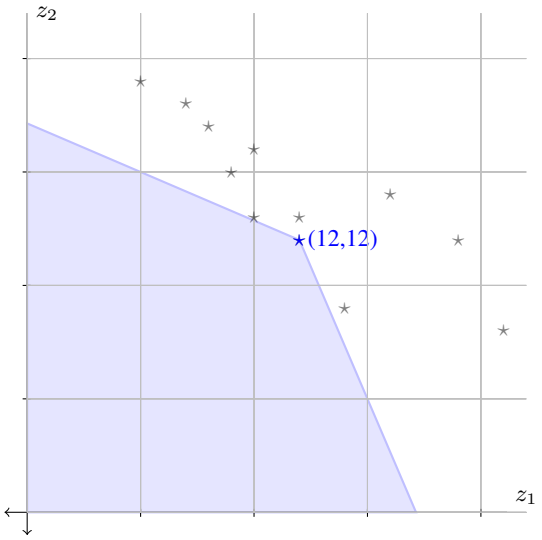

Figure 3.3 – La zone bleu représente les points dans l'espace des objectifs qui sont meilleurs que le point $(12, 12)$ qui représente la solution optimale pour l'intégrale de Choquet C_μ.

nœud i propagé dans l'algorithme d'étiquetage. Le label ℓ_i peut être supprimé si :

$$U(z(r^*_{s,t})) \leq \text{BORNE\_INF}(\ell_i),$$

avec BORNE_INF(ℓ_i) *une borne du label ℓ_i, inférieure à la valeur de la fonction d'utilité U pour n'importe quelle propagation du label ℓ_i : $\forall r_{i,t} \in R_{i,t}$,* BORNE_INF$(\ell_i) \leq U(z(\ell_i) \otimes z(r_{i,t}))$.

Théorème 3.1. Validité de la règle de coupe 3.1.
Soit BORNE_INF *une borne inférieure pour chaque label telle que $\forall r_{i,t} \in R_{i,t}$,* BORNE_INF$(\ell_i) \leq U(z(\ell_i) \otimes z(r_{i,t}))$. *Alors l'algorithme d'étiquetage (cf. Algorithme 2.1, page 51) utilisant la règle de coupe 3.1 permet de construire au moins une solution optimale selon U.*

Preuve du Théorème 3.1.
Soit $\mathbf{r}_{s,t} = \langle v_1, \ldots, v_l \rangle \in R_{s,t}$ un chemin optimal selon la fonction d'utilité U, avec $v_1, \ldots, v_l \in N$ des nœuds et $v_1 = s$ le nœud source, $v_l = t$ le nœud puits. Tout sous-chemin $r_{s,v_i} \in R_{s,v_i}$ du chemin $\mathbf{r}_{s,t}$ est associé avec un label ℓ_{v_i}, avec $1 \leq i \leq l$. Si aucun label $\ell_{v_1}, \ldots, \ell_{v_l}$ n'est supprimé par la règle de coupe alors un chemin optimal $\mathbf{r}_{s,t}$ est construit par l'algorithme d'étiquetage. Si un label ℓ_{v_i} $(1 \leq i \leq l)$ est supprimé par la règle de coupe alors, par définition, l'utilité du meilleur chemin connu $(U(r^*_{s,t}))$ est inférieure à l'utilité du chemin optimal $(U(\mathbf{r}_{s,t}))$ et donc le chemin $r^*_{s,t}$ est optimal selon U. Au moins un chemin optimal selon U entre s et t est construit.

□

3.1.1 Monotonie de la fonction d'utilité

Nous avons vu que dans le cadre du problème MOSP, si la fonction d'utilité U vérifie la monotonie (cf. Définition 2.12, page 67) alors l'optimisation de la fonction d'utilité U correspond à un problème de

plus court chemin mono-objectif [CMM90]. Même si la fonction d'utilité U ne vérifie pas la monotonie alors il est quand même possible d'exploiter la propriété de monotonie pour calculer une borne inférieure pour la règle de coupe 3.1.

Dans un premier temps nous définissons la propriété de linéarité pour une fonction U. Cette propriété, comme nous le voyons ensuite, peut être associée avec la propriété de monotonie. On considère par la suite que les opérateurs binaires sont identiques $\otimes_1 = \cdots = \otimes_p$, on utilise $\otimes = \otimes_1 = \cdots = \otimes_p$ et $\bigotimes = (\otimes_1, \ldots, \otimes_p)$. Ainsi si tous les opérateurs binaires sont des additions alors le problème MOSP correspondant est Q-$\sum$.

Définition 3.2. Fonction linéaire.
Une fonction d'agrégation (ou d'utilité) des coûts $U : \mathbf{\Omega} \to \xi$ est linéaire s.s.i. :

$$U(y^a) \otimes U(y^b) = U(y^a \bigotimes y^b), \quad \forall y^a, y^b \in \mathbf{\Omega}$$

Théorème 3.2. Condition nécessaire et suffisante pour la monotonie.
Une fonction d'agrégation ou d'utilité U vérifie la monotonie si il existe une fonction $g : \xi \to \xi$ croissante et une fonction d'agrégation linéaire $U' : \mathbf{\Omega} \to \xi$ tels que :

$$\forall y \in \mathbf{\Omega}, \; U(y) = g(U'(y))$$

Preuve du Théorème 3.2.
Soient une fonction d'utilité U composée d'une fonction g croissante et une fonction d'agrégation linéaire U'. Soit deux vecteurs de coûts $\forall y^a, y^b \in \mathbf{\Omega}$ tels que $U(y^a) \leq U(y^b)$ (ou $g(U'(y^a)) \leq g(U'(y^b))$) alors comme g est croissante : $U'(y^a) \leq U'(y^b)$. La fonction d'agrégation U' est linéaire alors $\forall y^c \in \mathbf{\Omega}$, $U'(y^a \bigotimes y^c) \leq U'(y^b \bigotimes y^c)$. Et donc, comme la fonction g est croissante alors la fonction U est monotone :

$$\forall y^c \in \mathbf{\Omega}, \; U(y^a \bigotimes y^c) \leq U(y^b \bigotimes y^c)$$

□

Le théorème précédent nous indique que la si la fonction d'utilité U est linéaire alors cette fonction vérifie la monotonie. Notons que Perny et Spanjaard [PS05] proposent un axiome d'indépendance qui est une définition plus générale de la monotonie telle que considérée dans cette thèse. Le théorème suivant nous permet définir différents cas de linéarité (et donc de monotonie) pour différents problèmes MOSP, Q-$\sum$, Q-$\prod$, Q-$\min$, Q-$\max$.

Théorème 3.3. Conditions suffisantes pour la linéarité.
Soit U une fonction d'utilité composée d'une fonction d'agrégation V et de fonctions d'utilité partielles $u_1, \ldots, u_p$, $\forall y \in \mathbf{\Omega}$, $U(y) = V(u_1(y_1), \ldots, u_p(y_p))$. U est linéaire si :

(a) La fonction d'agrégation V est définie par l'opérateur binaire : $\forall y \in \xi^p$, $V(y) = y_1 \otimes \cdots \otimes y_p$.

(b) Chaque fonction d'utilité partielle u_k, $k \in P$ est linéaire par rapport à l'opérateur binaire $\otimes_k$: $\forall y^a, y^b \in \mathbf{\Omega}$, $u_k(y^a_k) \otimes_k u_k(y^b_k) = u_k(y^a_k \otimes_k y^b_k)$.

Preuve du Théorème 3.3.
Soit U une fonction d'utilité vérifiant les conditions (a) et (b) du théorème 3.3. Soient $y^a, y^b \in \mathbf{\Omega}$ deux vecteurs de coûts. Les fonctions d'utilité partielles $u_1, \ldots, u_p$ sont linéaires :

$$V\left(u(y^a \bigotimes y^b)\right) = V\left(u_1(y^a_1) \otimes u_1(y^b_1), \ldots, u_p(y^a_p) \otimes u_p(y^b_p)\right)$$

La fonction d'agrégation V est définie par l'opérateur binaire $\otimes$, alors on peut décomposer la fonction d'agrégation :

$$V\left(u(y^a \otimes y^b)\right) = u_1(y_1^a) \otimes u_1(y_1^b) \otimes u_2(y_2^a) \otimes \dots u_{p-1}(y_{p-1}^b) \otimes u_p(y_p^a) \otimes u_p(y_p^b)$$

La fonction d'agrégation peut donc être recomposée en séparant les vecteurs de coûts y^a et y^b :

$$V\left(u(y^a \otimes y^b)\right) = V\left(u(y^a)\right) \otimes V\left(u(y^b)\right)$$

□

Grâce aux théorèmes précédents on peut définir clairement la forme d'une fonction d'agrégation vérifiant la monotonie pour différents problème MOSP. Et ainsi il nous est possible de déterminer avec quelle fonction d'agrégation l'optimisation de cette fonction dans le problème MOSP associé est équivalent à un problème de plus court chemin mono-objectif. Considérons une fonction d'utilité U composée d'une fonction d'agrégation V et de fonctions d'utilité partielles $u_1, \dots, u_k$. Le théorème 3.3 permet de construire plusieurs cas de monotonie :

- Les fonctions objectif sont additives ($\otimes = \otimes_1 = \dots = \otimes_p = +$)(problème Q-$\sum$). La fonction d'agrégation V est la somme des coûts : $V(y) = \sum_{k=1}^{p} u_k(y_k)$. Les fonctions d'utilité partielles sont linéaires $u_k(y_k) = \alpha_k.y_k$, $\alpha \in \mathbb{R}_+^p$. La fonction d'utilité U est alors une somme pondérée :

  $$\forall y \in \mathbf{\Omega},\ U(y) = \sum_{k=1}^{p} \alpha_k \times y_k$$

  Nous avons vu précédemment (cf. §1.1.2.2, page 15) que pour optimiser une somme pondérée dans un problème MOCO il suffit de scalariser les objectifs à l'aide d'une somme pondérée.
- Les fonctions objectif sont multiplicatives ($\otimes_1 = \dots = \otimes_p = \times$)(problème Q-$\prod$). La fonction d'agrégation V est le produit des coûts $V(y) = \prod_{k=1}^{p} u_k(y_k)$. Les fonctions d'utilité partielles sont des fonctions puissances $u_k(y_k) = (y_k)^{\alpha_k}$, $\alpha \in \mathbb{N}_+^p$ (cf. §2.4.2, page 67).

3.1.2 Règle de coupe utilisant un point idéal

Une borne simple peut être utilisée dans la règle de coupe 3.1 à condition que la fonction d'utilité U soit croissante (cf. Définition 1.9, page 18). Cette condition est nécessaire dans le cadre de l'optimisation multi-objectif (cf. §1.1.2.4, page 18). Cette borne utilise une fonction d'évaluation $q : N \to \mathbf{\Omega}$ telle que décrite dans la définition 2.10. En supposant que cette fonction d'évaluation soit optimiste (cf. Définition 2.10, page 59), alors $q(i)$ correspond alors pour chaque nœud $i \in N$ au point idéal (cf. Définition 1.3, page 13) des solutions efficaces de i à t : $\forall k \in P$, $q_k(i) = \min_{r_{i,t} \in R_{i,t}} z_k(r_{i,t})$.

Définition 3.3. Borne inférieure définie par une fonction d'évaluation.
Soit $q : N \leftarrow \mathbf{\Omega}$ une fonction d'évaluation (cf. Définition 2.10, page 59) permettant de borner inférieurement les coûts des chemins de n'importe que nœud $i \in N$ au nœud puits t : $q(i) \prec_P z(r_{i,t)}$, $\forall r_{i,t} \in R_{i,t}$. $U : \mathbf{\Omega} \to \xi$ une fonction d'utilité croissante (cf. Définition 1.9, page 18). Soit ℓ_i un label associé au nœud i propagé dans l'algorithme d'étiquetage, alors $q(i)$ correspond à la propagation idéale de ℓ_i pour atteindre le nœud puits t. La borne inférieure du label ℓ_i est :

$$\text{BORNE\_INF}^a(\ell_i) = U\left(z(\ell_i) \otimes q(i)\right)$$

Cette borne inférieure est proposée par Futtersack et Perny [FP00] (voir aussi [GP06a]) dans le cadre de l'optimisation d'une norme de Tchebychev et par Paixao et al [PMRS03] dans le cadre de l'optimisation de la norme euclidienne. Cette borne inférieure 3.3, peut être utilisée dans la règle de coupe 3.1. En effet selon la définition de la fonction d'évaluation q (cf. Définition 2.10, page 59) $\forall r_{i,t} \in R_{i,t}$, $q(i) \prec_P z(r_{i,t})$ et comme la fonction d'utilité U est croissante cette borne inférieure est vérifiée : $\forall r_{i,t} \in R_{i,t}$, $\text{BORNE\_INF}^a(\ell_i) \leq U(z(\ell_i) \otimes z(r_{i,t}))$.

Exemple 3.2. Utilisation de la règle de coupe 3.1 avec la borne inférieure 3.3, suite de l'exemple 3.1.
Cet exemple illustre l'application de la règle de coupe 3.1 avec la borne inférieure 3.3, dans un algorithme d'étiquetage (cf. Algorithme 2.1, page 51) pour le problème de l'exemple 3.1. On suppose que la fonction d'évaluation q (cf. Définition 2.10, page 59) est optimiste : $q(1) = (5, 8)$, $q(2) = (5, 4)$ et $q(3) = (0, 0)$. Soit $r^*_{s,t} \in R_{s,t}$ un chemin connu de s à t, tel que $z(r^*_{s,t}) = (10, 13)$. On considère deux labels ℓ_2, ℓ'_2 associés au nœud 2 : $z(\ell_2) = (0, 9)$ et $z(\ell'_2) = (11, 4)$. L'utilisation de la borne 3.3 avec la règle de coupe 3.1 permet alors de supprimer le label ℓ'_2 (cf. Figure 3.4, de la présente page) :

$$C_\mu(10, 13) = 10 \times 0.3 + 13 \times 0.7 = 12.1 \; > \; \text{BORNE\_INF}^a(\ell'_2) = C_\mu(5, 13) = 5 \times 0.3 + 13 \times 0.7 = 10.6$$

Cette borne ne permet pas de supprimer le label ℓ_2 (cf. Figure 3.5, de la présente page) : $C_\mu(10, 13) = 12.1 < 13.6 = \text{BORNE\_INF}^a(\ell'_2) = C_\mu(16, 8)$.

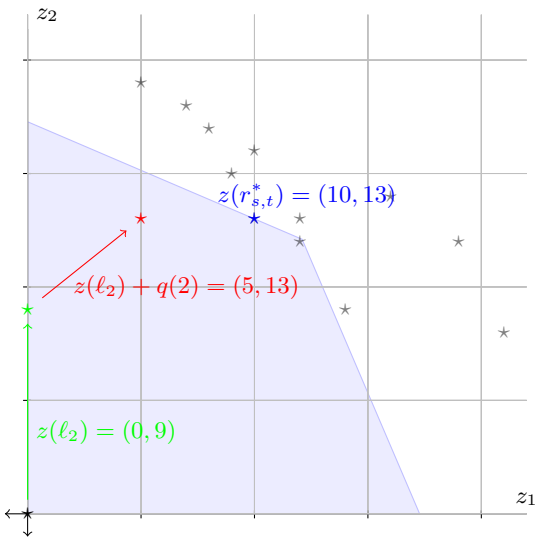

Figure 3.4 – Le vecteur vert $(0, 9)$ correspond au label ℓ_2, le vecteur rouge $(5, 4)$ correspond à $q(2)$. Le point bleu correspond à la meilleure solution connue.

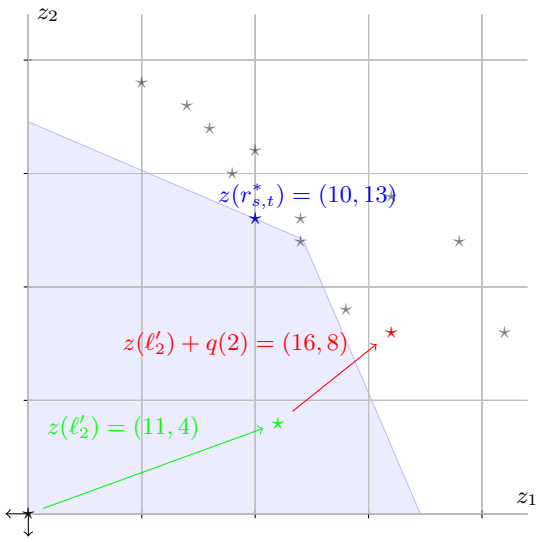

Figure 3.5 – Le vecteur vert $(11, 4)$ correspond au label ℓ'_2, le vecteur rouge $(5, 4)$ correspond à $q(2)$. Le point bleu correspond à la meilleure solution connue.

Il est possible de généraliser la borne 3.3 en calculant un ensemble de l fonctions d'évaluation $\mathfrak{Q} = \{q^1, \ldots, q^l\}$ plutôt qu'une unique fonction d'évaluation q.

Définition 3.4. Ensemble bornant.
Soient $q^a : N \to \mathbf{\Omega}$, $a \in \{1, \ldots, l\}$ des fonctions de coûts sur les nœuds. Soit $\mathfrak{Q} = \{q^1, \ldots, q^l\}$ un ensemble bornant inférieurement les chemins, entre les nœuds du graphe et le nœud puits, tel que :

$$\forall i \in N, \; \forall r_{i,t} \in R_{i,t}, \exists q \in \mathfrak{Q} \text{ t.q. } q(i) \prec_P z(r_{i,t})$$

Un ensemble bornant est une généralisation de la notion de fonction d'évaluation bornant inférieurement les chemins telle que définie en 2.10. L'ensemble bornant $\mathfrak{Q}$ peut être construit en scalarisant (cf. §1.1.2.2, page 15) le problème MOSP à plusieurs reprises avec plusieurs sommes pondérées afin de calculer un certain nombre de solutions supportées. La méthode de dichotomie (cf. §1.1.4.2, page 21) peut par exemple être utilisée pour calculer ces solutions supportées. Ce type de méthode est donc difficile à utiliser avec plus de deux objectifs. Dans le cas bi-objectif, si on souhaite calculer un ensemble $\mathfrak{Q}$ de taille l, il faut calculer un ensemble $l+1$ de solutions supportées. Ensuite, pour un nœud $i \in N$, chaque paire de solutions supportées adjacentes $r^a_{s,i}$ et $r^b_{s,i}$ ($z_1(r^a_{s,t}) < z_1(r^b_{s,t})$) permet de définir les valeurs de la fonction d'évaluation q sur le nœud i : $q(i) = (z_1(r^a_{s,t}), z_2(r^b_{s,t}))$.

Définition 3.5. Borne inférieure définie par un ensemble bornant.
Soit $\mathfrak{Q}$ un ensemble bornant. $U : \mathbf{\Omega} \to \xi$ une fonction d'utilité croissante (cf. Définition 1.9, page 18). Soit ℓ_i un label associé au nœud $i \in N$, la borne inférieure du label ℓ_i est :

$$\text{BORNE\_INF}^b(\ell_i) = \min_{q \in \mathfrak{Q}} U\left(z(\ell_i) \otimes q(i)\right)$$

Cette borne inférieure est une généralisation des travaux de Sauvanet et Néron [SN10] pour l'optimisation de la norme de Tchebychev dans le cadre du problème MOSP 2-$\sum$. Elle est aussi utilisée dans le cadre du calcul de l'ensemble des solutions efficaces [EG01, MdlC05b]. La borne inférieure 3.3, peut être utilisée dans la règle de coupe 3.1. En effet selon la définition de l'ensemble bornant $\mathfrak{Q}$, $\forall r_{i,t} \in R_{i,t}$, $\exists q' \in \mathfrak{Q}$, tel que $q'(i) \prec_P z(r_{i,t})$ et comme la fonction d'utilité U est croissante, cette borne inférieure est valide : $\forall r_{i,t} \in R_{i,t}$, $\text{BORNE\_INF}^b(\ell_i) \leq U(z(\ell_i) \otimes q'(i)) \leq U(z(\ell_i) \otimes z(r_{i,t}))$.

Exemple 3.3. Utilisation de la règle de coupe 3.1 avec la borne inférieure 3.3, suite de l'exemple 3.1.
Cet exemple illustre l'application de la règle de coupe 3.1 avec la borne inférieure 3.3, dans un algorithme d'étiquetage pour le problème de l'exemple 3.1. Les fonctions d'évaluation $q^1, q^2 \in \mathfrak{Q}$ sont calculées telles que : $q^1(1) = (5,13)$, $q^2(1) = (10,8)$, $q^1(2) = (5,8)$, $q^2(2) = (8,4)$, $q^1(3) = (0,0)$ et $q^2(3) = (0,0)$. Les deux fonction d'évaluation q^1 et q^2 sont calculées en résolvant 3 scalarisation du problème MOSP à l'aide des sommes pondérées $\varphi_{(1,0)}$, $\varphi_{(0,1)}$, $\varphi_{(0.5,0.5)}$ (cf. Figure 3.6).

Soit $r^*_{s,t} \in R_{s,t}$ un chemin connu de s à t, tel que $z(r^*_{s,t}) = (10,13)$. On considère deux labels ℓ_2, ℓ'_2 associés au nœud 2 : $z(\ell_2) = (0,9)$ et $z(\ell'_2) = (11,4)$. L'utilisation de la borne inférieure avec la règle de coupe 3.1 permet alors de supprimer le label ℓ'_2 (cf. Figure 3.7, page 81) :

$$C_\mu(10,13) = 12.1 \;\leq\; \text{BORNE\_INF}^b(\ell'_2) = \min \left\{ \begin{array}{c} C_\mu(\ell'_2 + q^1(2)) = C_\mu(16,12) = 14.8 \\ C_\mu(\ell'_2 + q^2(2)) = C_\mu(19,8) = 15.7 \end{array} \right\} = 14.8$$

Cette règle ne permet pas de supprimer le label ℓ_2 (cf. Figure 3.8, page 81) : $C_\mu(10,13) = 12.1 >$ $\text{BORNE\_INF}^b(\ell'_2) = \min\{C_\mu(4,17) = 13.5, C_\mu(8,13) = 11.5\}$.

3.1.3 Règle de coupe utilisant une fonction linéaire pour borner inférieurement une fonction d'utilité

Si il est possible de calculer une fonction linéaire bornant inférieurement une fonction d'utilité U, alors un autre type de borne inférieure pour un label ℓ_i peut être calculé.

Définition 3.6. Fonction d'agrégation bornant une fonction d'utilité.
Soit U une fonction d'utilité. La fonction d'agrégation $\underline{U}$ donne une borne inférieure de la fonction U sur $\mathbf{\Omega}$ s.s.i. :

$$\forall y \in \mathbf{\Omega},\ \underline{U}(y) \leq U(y)$$

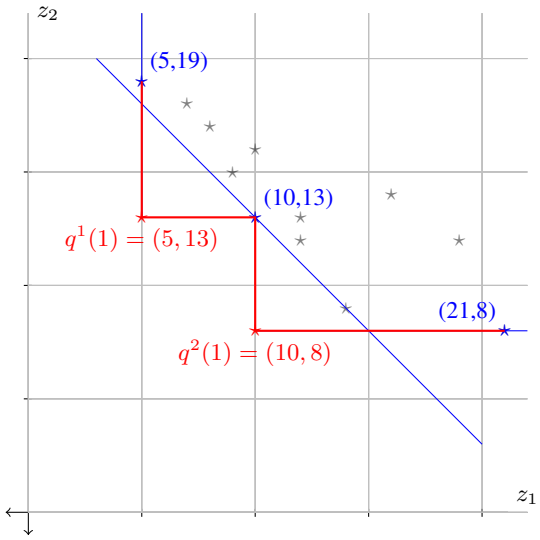

Figure 3.6 – Les trois points en bleus représentent les trois solutions supportées calculées, les deux points rouges représentent les valeurs des fonctions d'évaluation q^1 et q^2 pour le nœud 1.

Perny et Spanjaard [PS05] proposent une définition semblable dans le cadre de l'optimisation d'une relation de préférence $\precsim$ (cf. §1.2.2.2, page 25). En bornant les fonctions d'utilité partielles et la fonction d'agrégation alors il est possible de calculer une fonction d'agrégation bornant la fonction d'utilité. Si pour chaque objectif $k \in P$, la fonction d'utilité partielle $\underline{u_k}$ borne inférieurement u_k (sur Ω_k) et la fonction d'agrégation $\underline{V}$ borne inférieurement V (sur ξ) alors la fonction d'agrégation $\underline{U}$ borne inférieurement U (sur $\mathbf{\Omega}$).

Définition 3.7. Fonction d'évaluation d'une fonction d'agrégation monotone.
Soit $\underline{U} : \mathbf{\Omega} \to \xi$ une fonction d'agrégation monotone par rapport , la fonction d'évaluation $q^{\underline{U}} : N \to \mathbf{\Omega}$ (cf. Définition 2.10, page 59) est définie par rapport $\underline{U}$ telle que :

$$\forall i \in N,\ \forall r_{i,t} \in R_{i,t},\ \underline{U}(q^{\underline{U}}(i)) \leq \underline{U}(r_{i,t})$$

Dans le cadre du problème Q-$\sum$ la somme pondérée est linéaire et donc monotone (cf. Définition 3.2, page 76), d'autres cas de monotonie sont présentés dans la sous-section 3.1.1. Puisque $\underline{U}$ est monotone alors il est facile de calculer a priori la fonction d'évaluation $q^{\underline{U}}$. L'optimisation de $\underline{U}$ dans le problème de plus court chemin $R_{\bullet,t}$ (qui permet de définir $q^{\underline{U}}$ pour chaque nœud $i \in N$) est équivalent à un problème mono-objectif puisque $\underline{U}$ est monotone.

Définition 3.8. Borne inférieure définie par une fonction d'agrégation monotone.
Soient $U : \mathbf{\Omega} \to \xi$ une fonction d'utilité et $\underline{U}$ une fonction d'agrégation linéaire (cf. Définition 3.2, page 76) qui borne inférieurement U (cf. Définition 3.6, page précédente). $q^{\underline{U}}$ est la fonction d'évaluation définie selon $\underline{U}$ (cf. Définition 3.7, de la présente page). Soit ℓ_i un label associé au nœud $i \in N$ propagé dans l'algorithme d'étiquetage. La borne inférieure du label ℓ_i est :

$$\text{BORNE\_INF}^c(\ell_i) = \underline{U}\left(z(\ell_i) \otimes q^{\underline{U}}(i)\right)$$

Cette borne inférieure est une généralisation des travaux de Paixao et al [PMRS03] dans le cadre de l'optimisation de la norme euclidienne ($U = ||.||_2$) et des travaux de Galand et Spanjaard [GS07]

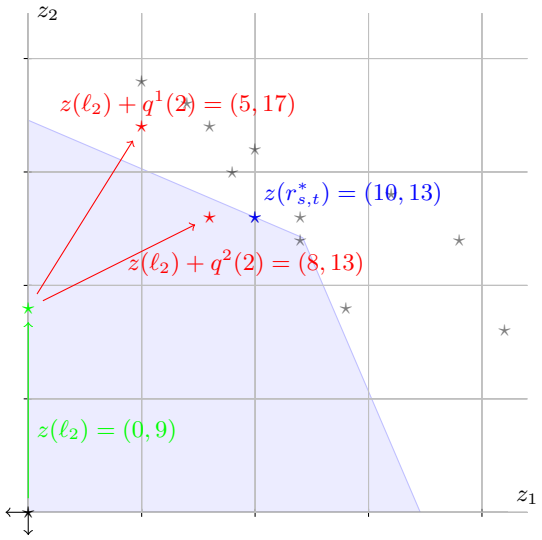

Figure 3.7 – Le vecteur vert $(0, 9)$ correspond au label ℓ_2, les vecteurs rouges $(5, 8)$ et $(8, 4)$ correspondent à $q^1(2)$ et $q^2(2)$ respectivement. Le point bleu correspond à la meilleure solution connue.

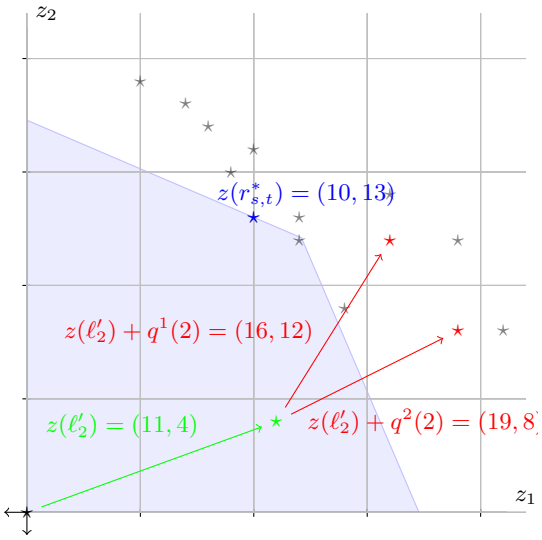

Figure 3.8 – Le vecteur vert $(11, 4)$ correspond au label ℓ'_2, les vecteurs rouges $(5, 8)$ et $(8, 4)$ correspondent à $q^1(2)$ et $q^2(2)$ respectivement. Le point bleu correspond à la meilleure solution connue.

dans le cadre de l'optimisation d'une fonction d'agrégation OWA ($U = \Psi^{owa}_{\alpha}$). Elle est aussi utilisée pour le problème de plus court chemin dans les travaux suivants [GP07, FGL11a, GPS10]. Cette borne inférieure 3.8, peut être utilisée dans la règle de coupe 3.1. En effet selon la définition de la fonction d'évaluation $q^{\underline{U}} : \forall r_{i,t} \in R_{i,t}, \underline{U}(z(\ell_i) \otimes q^{\underline{U}}(j)) \leq \underline{U}(z(\ell_i) \otimes z(r_{i,t}))$. Comme la fonction d'agrégation $\underline{U}$ borne inférieurement la fonction d'utilité $U : \forall r_{i,t} \in R_{i,t}$, BORNE_INF$^c(\ell_i) \leq U(z(\ell_i) \otimes z(r_{i,t}))$.

Exemple 3.4. Utilisation de la règle de coupe 3.1 avec la borne inférieure 3.8, suite de l'exemple 3.1. *Soit une somme pondérée $\varphi_{(0.3,0.7)}$ qui permet de borner inférieurement l'intégrale de Choquet C_μ (cf. Exemple 3.1, page 74). La construction de la somme pondérée est décrite en section 3.2.1. La somme pondérée $\varphi_{(0.3,0.7)}$ est linéaire (cf. Définition 3.2, page 76) comme il s'agit d'un problème 2-$\sum$. La fonction d'évaluation $q^{\varphi_{(0.3,0.7)}}$ est construite telle que : $q^{\varphi_{(0.3,0.7)}}(1) = (14, 9)$, $q^{\varphi_{(0.3,0.7)}}(2) = (10, 4)$ et $q^{\varphi_{(0.3,0.7)}}(3) = (0, 0)$. Soit $r^*_{s,t} \in R_{s,t}$ un chemin connu de s à t, tel que $z(r^*_{s,t}) = (12, 12)$. On considère deux labels ℓ_2, ℓ'_2 associés au nœud 2 : $z(\ell_2) = (0, 9)$ et $z(\ell'_2) = (11, 4)$. L'utilisation de la borne 3.8 avec la règle de coupe 3.1 permet de supprimer le label ℓ_2 (cf. Figure 3.9) :*

$$C_\mu(12, 12) = 12 \leq \text{BORNE\_INF}^c(\ell_2) = \varphi_{(0.3,0.7)}(\ell_2 + q^{\varphi_{(0.3,0.7)}}(2)) = 10 \times 0.3 + 13 \times 0.7 = 12.1$$

Cette borne ne permet pas de supprimer le label ℓ'_2 (cf. Figure 3.10) : $C_\mu(12, 12) = 12 >$ BORNE_INF$^c(\ell'_2) = \varphi_{(0.3,0.7)}(\ell'_2 + q^{\varphi_{(0.3,0.7)}}(2)) = 21 \times 0.3 + 8 \times 0.7 = 11.9$

3.1.3.1 Généralisation

Une approche similaire à celle des ensembles bornant peut être utilisée en définissant plusieurs fonctions d'agrégation monotones bornant la fonction d'utilité U. Ces fonctions peuvent ensuite être utilisées simultanément.

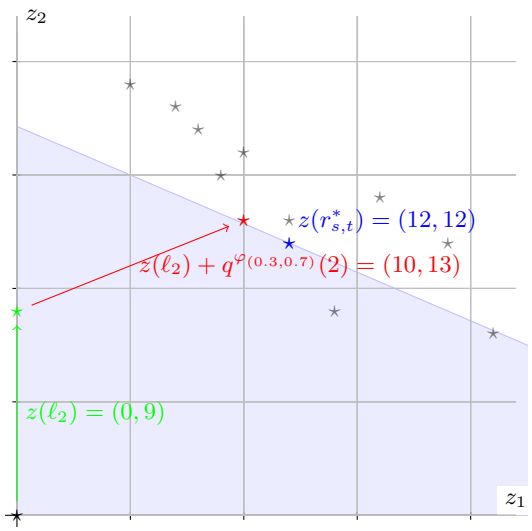

Figure 3.9 – Le vecteur vert $(0, 9)$ correspond au label ℓ_2, le vecteur rouge $(10, 4)$ correspond à $q^{\varphi_{(0.3,0.7)}}(2)$. Le point bleu correspond à la meilleure solution connue.

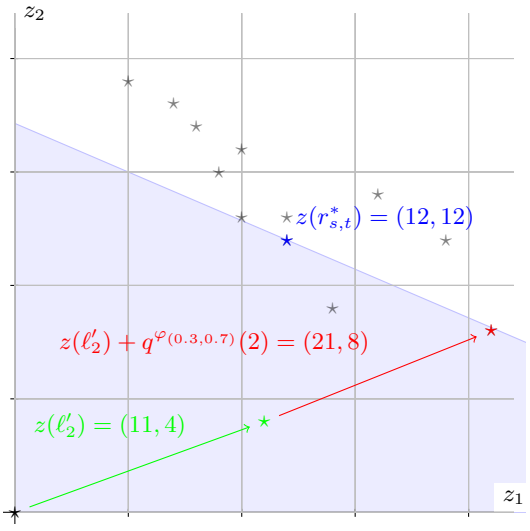

Figure 3.10 – Le vecteur vert $(11, 4)$ correspond au label ℓ'_2, le vecteur rouge $(10, 4)$ correspond à $q^{\varphi_{(0.3,0.7)}}(2)$. Le point bleu correspond à la meilleure solution connue.

Définition 3.9. Borne inférieure selon un ensemble de fonctions d'agrégation monotones.
Soit $U : \mathbf{\Omega} \to \xi$ une fonction d'utilité $\mathfrak{U} = \{\underline{U}^1, \ldots, \underline{U}^l\}$ un ensemble de l fonctions d'agrégation monotones qui bornent inférieurement U. Soient $q^{\underline{U}^1}, \ldots, q^{\underline{U}^l}$, les fonctions d'évaluation des fonctions d'agrégation de $\mathfrak{U}$. Soit ℓ_i un label associé au nœud $i \in N$ propagé dans l'algorithme d'étiquetage. La borne inférieure du label ℓ_i est :

$$\text{BORNE\_INF}^d(\ell_i) = \max_{\underline{u} \in \mathfrak{U}} \left\{ \underline{U} \left(z(\ell_i) \otimes q^{\underline{U}}(i) \right) \right\}$$

Cette borne inférieure 3.9, peut être utilisée dans la règle de coupe 3.1. En effet, selon la définition des fonctions linéaires $\mathfrak{U}$ et des fonctions d'évaluation $q^{\underline{U}^1}, \ldots, q^{\underline{U}^l} : \forall r_{i,t} \in R_{i,t}, \forall \underline{U} \in \mathfrak{U}, \underline{U}(z(\ell_i) \otimes q^{\underline{U}}(j)) \leq \underline{U}(z(\ell_i) \otimes z(r_{i,t}))$. Comme la fonction d'agrégation $\underline{U}$ borne inférieurement la fonction d'utilité U : $\forall r_{i,t} \in R_{i,t}$, $\text{BORNE\_INF}^d(\ell_i) \leq U(z(\ell_i) \otimes z(r_{i,t}))$.

Exemple 3.5. Utilisation de la règle de coupe 3.1 avec la borne inférieure 3.9, suite de l'exemple 3.1.
Soient deux sommes pondérées $\varphi_{(0.3,0.7)}$ et $\varphi_{(0.7,0.3)}$ qui chacune indépendamment borne inférieurement l'intégrale de Choquet C_μ (cf. Exemple 3.1, page 74). Comme tous les objectifs sont additifs alors les sommes pondérées $\varphi_{(0.3,0.7)}$ et $\varphi_{(0.6,0.4)}$ sont monotones. Les fonctions d'évaluation $q^{\varphi_{(0.3,0.7)}}$ et $q^{\varphi_{(0.6,0.4)}}$ des sommes pondérées $\varphi_{(0.3,0.7)}$ et $\varphi_{(0.6,0.4)}$ sont construites telles que : $q^{\varphi_{(0.3,0.7)}}(1) = (14, 9)$, $q^{\varphi_{(0.3,0.7)}}(2) = (10, 4)$, $q^{\varphi_{(0.3,0.7)}}(3) = (0, 0)$, $q^{\varphi_{(0.6,0.4)}}(1) = (10, 13)$, $q^{\varphi_{(0.6,0.4)}}(2) = (5, 10)$ et $q^{\varphi_{(0.6,0.4)}}(3) = (0, 0)$. Soit $r^*_{s,t} \in R_{s,t}$ un chemin connu de s à t, tel que $z(r^*_{s,t}) = (12, 12)$. On considère deux labels ℓ_2, ℓ'_2 associés au nœud 2 : $z(\ell_2) = (0, 9)$ et $z(\ell'_2) = (11, 4)$. L'utilisation de la borne 3.9 avec la règle de coupe 3.1 permet alors de supprimer le label ℓ_2 (cf. Figure 3.11, page ci-contre) :

$$C_\mu(12, 12) = 12 \;\geq\; \text{BORNE\_INF}^d(\ell_i) = \max \left\{ \begin{array}{l} \varphi_{(0.6,0.4)}(5, 19) = 10.6 \\ \varphi_{(0.3,0.7)}(10, 13) = 12.1 \end{array} \right\} = 12.1;$$

Cette borne permet aussi de supprimer le label ℓ'_2 (cf. Figure 3.11, de la présente page) : $C_\mu(12,12) = 12 \geq \text{BORNE\_INF}^d(\ell_i) = \max\{\varphi_{(0.6,0.4)}(16,14) = 15.2, \varphi_{(0.3,0.7)}(21,8) = 11.9\} = 15.4$

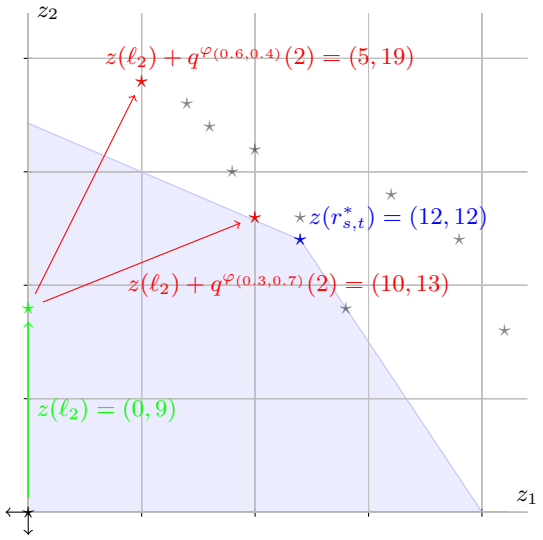

Figure 3.11 – Le vecteur vert $(0,9)$ correspond au label ℓ_2, les vecteurs rouges $(10,4)$ et $(5,10)$ correspondent à $q^{\varphi(0.3,0.7)}(2)$ et $q^{\varphi(0.6,0.4)}(2)$ respectivement. Le point bleu correspond à la meilleure solution connue.

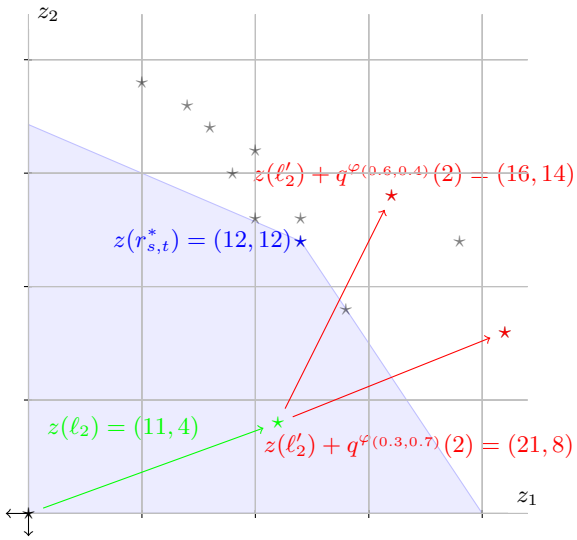

Figure 3.12 – Le vecteur vert $(11,4)$ correspond au label ℓ'_2, les vecteurs rouges $(10,4)$ et $(5,10)$ correspondent à $q^{\varphi(0.3,0.7)}(2)$ et $q^{\varphi(0.6,0.4)}(2)$ respectivement. Le point bleu correspond à la meilleure solution connue.

3.1.4 Conclusion : bornes inférieures des labels

Nous avons proposé dans cette section une généralisation de différentes travaux existants sur les bornes inférieures qui peuvent être utilisées dans la règle de coupe (cf. Définition 3.1, page 74). La dernière borne inférieure (BORNE_INF^d) peut être considérée comme une nouvelle proposition. Ces 4 bornes inférieures sont toutes basées sur la résolution a priori d'un ou de plusieurs problèmes de plus court chemin mono-objectif. Les deux premières bornes inférieures sur le calcul d'une fonction d'évaluation en résolvant pour chaque objectif indépendamment le problème de plus court chemin mono-objectif. Les deux autres bornes inférieures sont basées sur le calcul d'une fonction d'agrégation monotone qui borne la fonction d'utilité puis sur l'optimisation de cette fonction d'agrégation en résolvant un problème de plus court chemin mono-objectif.

3.2 Calcul d'une fonction d'agrégation monotone bornant une fonction d'utilité basée sur l'intégrale de Choquet

Considérons le problème MOSP $(R_{s,t})$ avec un nombre indéfini de fonctions objectif (Q-$\sum$). Les coûts des arcs sont positifs, $\mathbf{\Omega} = \Omega_1 \times \cdots \times \Omega_p = \mathbb{R}^p_+$ et vérifient les conditions de terminaison de l'algorithme d'étiquetage (cf. Théorème 2.5, page 55). On cherche à minimiser une fonction d'utilité $U = \Psi^u_\mu$ composée d'une intégrale de Choquet $C_\mu : \xi^p \to \xi$ (cf. Définition 1.17, page 33) et de fonctions

d'utilité partielles $u_1, \ldots, u_p$ linéaires par morceaux et croissantes (cf. Définition 1.12, page 28). Pour un vecteur de coûts $y \in \mathbf{\Omega}$ la fonction Ψ_μ^u avec $u_k : \Omega_k \in \mathbb{R} \to \xi \in \mathbb{R}$, les fonctions d'utilité partielles est définie telle que :

$$\Psi_\mu^u(y) = C_\mu(u_1(y_1), \ldots, u_p(y_p))$$

L'intégrale de Choquet est par définition croissante (cf. Définition 1.9, page 18). Les fonctions d'utilité partielles sont aussi croissantes, alors la fonction d'utilité Ψ_μ^u est croissante. Les bornes 3.3 et 3.3 peuvent donc être utilisées avec cette fonction d'utilité.

Une somme pondérée est linéaire dans le cadre du problème Q-$\sum$ (cf. §3.1.1, page 75). Ainsi, une somme pondérée $\underline{U} = \varphi_\lambda$ ($\lambda \in \mathbb{R}_+^p$) bornant la fonction d'utilité $U = \Psi_\mu^u$ doit être construite. Les sous-sections suivantes proposent différentes méthodes pour construire une somme pondérée bornant Ψ_μ^u. Dans un premier temps des méthodes pour calculer une somme pondérée bornant l'intégrale de Choquet sont proposées : $\forall y \in \mathbf{\Omega}$, $\underline{V}(y) = \varphi_\lambda(y) = \sum_{k=1}^p \lambda_k \times y_k$. Dans un deuxième temps des méthodes pour construire des fonctions linéaires bornant les fonctions d'utilité partielles sont proposées : $\forall y \in \mathbf{\Omega}$, $\forall k \in P$, $\underline{u}_k(y_k) = \alpha_k \times y_k$. On en déduit une somme pondérée $\underline{U}(y) = \sum_{k=1}^p \lambda_k \times \alpha_k \times y_k$ borne la fonction d'utilité Ψ_μ^u pour tout $y \in \mathbf{\Omega}$.

3.2.1 Construire une somme pondérée bornant l'intégrale de Choquet

Une somme pondérée φ_λ borne inférieurement l'intégrale de Choquet (cf. Définition 3.6, page 79) s.s.i. :

$$\forall y \in \mathbb{R}_+^p,\ \varphi_\lambda(y) \leq C_\mu(y) \tag{3.1}$$

Résoudre l'équation (3.1) ne peut être fait en posant programme linéaire puisque l'intégrale de Choquet n'est pas une fonction linéaire. Néanmoins on montre * dans le théorème suivant qu'une somme pondérée bornant l'intégrale de Choquet peut être calculée en résolvant un problème linéaire.

Théorème 3.4.
Soit μ une fonction de capacité, C_μ l'intégrale de Choquet correspondante et $\lambda \in \mathbb{R}_+^p$ un vecteur de poids. L'équivalence suivante est alors vérifiée :

$$\forall y \in \mathbb{R}_+^p,\ \varphi_\lambda(y) \leq C_\mu(y) \Leftrightarrow \forall A \subseteq P,\ \sum_{k \in A} \lambda_k \leq \mu(A) \tag{3.2}$$

Preuve du Théorème 3.4.
Les deux sens de l'équivalence 3.4 sont démontrés ci-dessous :

- $\Rightarrow$ *Soit μ une fonction de capacité et $\lambda \in \mathbb{R}_+^p$ un vecteur de poids tel que : $\forall A \subseteq E$, $\sum_{i \in A} \lambda \leq \mu(A)$. $y \in \mathbb{R}_+^p$ un vecteur de coûts quelconque et $(.)$ une permutation telle que $0 \leq y_{(1)} \leq \cdots \leq y_{(p)}$. Selon la définition d'une somme pondérée : $\varphi_\lambda(y) = \sum_{k=1}^p \lambda_k \times y_k = \sum_{k=1}^p \lambda_{(k)} \times y_{(k)}$. L'équation suivante est vérifiée :*

$$\varphi_\lambda(y) = \sum_{k=1}^p \left[\sum_{j=k}^p \lambda_{(j)} - \sum_{j=k+1}^p \lambda_{(j)} \right] \times y_{(k)}$$

Soit $y_{(0)} = 0$, alors φ_λ peut être définie de la manière suivante :

$$\varphi_\lambda(y) = \sum_{k=1}^p \left[y_{(k)} - y_{(k-1)} \right] \times \sum_{j=k}^p \lambda_{(j)}$$

∗. Ce résultat a été présenté à la conférence MCDM 2009 et est publié dans les actes de la conférence [FGL11a].

Selon la définition de l'intégrale de Choquet (cf. Définition 1.17, page 33) : $C_\mu(y) = \sum_{k=1}^{p} \left[y_{(k)} - y_{(k-1)}\right] \times \mu(Y_{(k)})$, *avec* $Y_{(k)} = \{(1), \ldots, (k)\}$. *Ainsi* $C_\mu(y) - \varphi_\lambda(y) = \sum_{k=1}^{p} \left[y_{(k)} - y_{(k-1)}\right] \times \left(\mu(Y_{(k)}) - \sum_{j=k}^{p} \lambda_{(j)}\right)$. *Selon la définition de la permutation* $(.)$ *:* $y_{(k)} - y_{(k-1)} \geq 0$. *Selon la définition du vecteur de poids* $\lambda \in \mathbb{R}_+^p$ *:* $\mu(Y_{(k)}) - \sum_{j=k}^{p} \lambda_{(j)} \geq 0$. *Alors* $C_\mu(y) - \varphi_\lambda(y) \geq 0$.

- $\Leftarrow$ *Soient* μ *une fonction de capacité et* $\lambda \in \mathbb{R}_+^p$ *un vecteur de poids tel que :* $\forall y \in \mathbb{R}_+^p$, $\varphi_\lambda(y) \leq C_\mu(y)$. *Pour un sous-ensemble d'objectifs* $A \subset P$ *on définit* $y^A \in \mathbb{R}_+^p$ *un vecteur de coûts tel que :* $\forall k \in A$, $y_k^A = 1$ *et* $\forall k \in P \setminus A$, $y_k^A = 0$. *Pour l'intégrale de Choquet* C_μ *l'équation suivante est vérifiée :*

$$C_\mu(y^A) = \sum_{k=1}^{|P\setminus A|} [0-0] \times \mu(Y_{(i)}) + 1 \times \mu(A) + \sum_{k=|P\setminus A|+1}^{p} [1-1] \times \mu(Y_{(i)}) = \mu(A)$$

Pour la somme pondérée φ_λ *l'équation suivante est vérifiée :*

$$\varphi_\lambda(y^A) = \sum_{k\in A} \lambda_k \times 1 + \sum_{k\in P\subset A} \lambda_k \times 0 = \sum_{k\in A} \lambda_k$$

On obtient donc l'inéquation suivante $\mu(A) \geq \sum_{k\in A} \lambda_k$.

□

Parmi l'ensemble des vecteurs de poids satisfaisant les contraintes de l'équation 3.2, tous ne permettent pas d'obtenir des bornes de la même qualité. Supposons qu'une fonction objectif f permette de sélectionner le meilleur vecteur de poids, on peut construire une somme pondérée φ_λ bornant C_μ en résolvant le problème linéaire (3.3) suivant.

$$\begin{array}{lll} \max & f(\lambda) & \\ \text{s.c.} & \sum_{k\in A} \lambda_k \leq \mu(A) & \forall A \subseteq P \\ & \lambda_k \in]0,1] & \forall k \in P \end{array} \tag{3.3}$$

Afin que la somme pondérée φ_λ soit strictement croissante, on considère que chaque poids est strictement positif : $\forall k \in P$, $\lambda_k \in]0,1]$. Le choix d'une fonction objectif est un problème complexe étudié dans la sous-section 3.2.3.

L'intégrale de Choquet étant idempotente ($C_\mu(y_1, \ldots, y_p) = y_1 = \cdots = y_p$) alors la somme des poids est nécessairement inférieure ou égale à 1 : $\sum_{k=1}^{p} \lambda_k \leq 1$. Tout les objectifs étant à minimiser alors chaque poids λ_k, $k \in P$, est positif ou nul : $0 \leq \lambda_k$. Si la fonction de capacité μ est submodulaire avec une inégalité stricte telle que : $\exists A, B \subset P$ $A \cap B = \emptyset$, $\mu(A) + \mu(B) = \mu(A \cup B)$. Alors la somme des poids λ construit avec le problème linéaire (3.3) est strictement inférieure à 1 : $\sum_{k=1}^{p} \lambda_k < 1$. Et donc la borne inférieure 3.8 sera moins haute et la règle de coupe 3.1 permettra de supprimer moins de labels.

3.2.2 Utilisation de bornes sur les objectifs pour construire une somme pondérée

L'équation (3.1) indique que la somme pondérée doit borner l'intégrale de Choquet pour chaque point dans $\mathbf{\Omega}$ et on considère donc que chaque point représente une solution. Néanmoins il n'est pas nécessaire de prendre en compte l'ensemble des points potentiels $\mathbf{\Omega}$ mais puisque l'on souhaite uniquement construire une solution optimale, alors l'espace de recherche peut être restreint.

Définition 3.10.
La somme pondérée φ_λ borne (inférieurement) l'intégrale de Choquet C_μ s.s.i. il existe une solution optimale $\mathbf{r}_{s,t} \in R_{s,t}$ selon C_μ ($C_\mu(z(\mathbf{r}_{s,t})) \leq C_\mu(z(r'_{s,t}))\ \forall r'_{s,t} \in R_{s,t}$) telle que :

$$\varphi_\lambda(z(\mathbf{r}_{s,t})) \leq C_\mu(z(\mathbf{r}_{s,t})) \tag{3.4}$$

Théorème 3.5.
Si la somme pondérée φ_λ vérifie l'équation (3.4), alors cette somme pondérée peut être utilisée pour calculer la borne 3.8 dans la règle de coupe 3.1. Au moins une solution optimale sera construite.

Preuve du Théorème 3.5.
Soit ℓ_j avec $j \in N$, un label correspondant à un chemin $r_{s,i}$ de s à i. Ce chemin $r_{s,i}$ est un sous-chemin du chemin $\mathbf{r}_{s,t}$ de s à t optimal selon U, tel que $\varphi_\lambda(z(\mathbf{r}_{s,t})) \leq C_\mu(z(\mathbf{r}_{s,t}))$. Alors le label ℓ_i n'est pas supprimé par la règle de coupe 3.1 et la borne BORNE_INF^c *(cf. Définition 3.8, page 80) : $\forall r'_{s,t} \in R_{s,t}$, $\forall r_{i,t} \in R_{i,t}$,*

$$\text{BORNE\_INF}^c(\ell_i) = \varphi_\lambda(z(\ell_i) + q^{\varphi_\lambda}(i)) \leq \varphi_\lambda(z(\ell_i) + z(r_{i,t})) \leq C_\mu(z(\mathbf{r}_{s,t})) \leq C_\mu(z(r'_{s,t}))$$

Ainsi, le coût agrégé, selon la somme pondérée φ_λ, du label ℓ_i associé au nœud i augmenté de la valeur de la fonction d'évaluation q^{φ_λ} pour le nœud i est nécessairement moins important que le coût agrégé de n'importe quel chemin de s à t selon l'intégrale de Choquet.

□

Notons que l'équation (3.4) de la définition précédente est une redéfinition de l'équation (3.1). La définition 3.4 et le théorème 3.5 peuvent être généralisés dans le cadre de l'utilisation d'une fonction d'agrégation linéaire $\underline{U}$ bornant inférieurement la fonction d'utilité U. En pratique l'équation (3.4) ne peut être utilisée pour calculer une somme pondérée φ_λ puis aucun chemin optimal selon C_μ n'est connu a priori. On peut néanmoins exploiter certaines propriétés de cet ensemble de chemins.

Définissons $\underline{\xi}$ la borne inférieure et $\overline{\xi}$ la borne supérieure encadrant les coûts de $\mathbf{r_{s,t}}$: $\forall k \in P$, $z_k(\mathbf{r_{s,t}}) \in [\underline{\xi}, \overline{\xi}] \subseteq \xi \subseteq \mathbb{R}_+$. Les bornes inférieure $\underline{\xi}$ et supérieure $\overline{\xi}$ peuvent être déterminées a priori,ou calculées durant l'initialisation de l'algorithme d'étiquetage. Comme l'intégrale de Choquet est une fonction croissante, au moins une solution optimale selon C_μ est efficace (cf. §1.1.2.4, page 18). Alors plutôt que de considérer des bornes pour une unique solution optimale, on peut considérer des bornes sur l'ensemble des solutions efficaces : $\forall r_{s,t} \in E(R_{s,t})$, $z(r_{s,t}) \in [\underline{\xi}, \overline{\xi}]^p$. Ces bornes peuvent être définies par le point idéal (y^I) et le point nadir (y^N) : $\underline{\xi} = \min_{k \in P} u_k(y^I_k)$ et $\overline{\xi} = \max_{k \in P} u_k(y^N_k)$. Ainsi définissons l'équation (3.5) suffisante pour vérifier l'équation (3.4) :

$$\forall y \in [\underline{\xi}, \overline{\xi}]^p,\ \varphi_\lambda(y) \leq C_\mu(y) \tag{3.5}$$

L'équation (3.5) ne peut pas être résolue en posant un problème linéaire, alors l'équation (3.6) qui peut être résolue avec un problème linéaire est proposée :

$$\forall y \in \{\underline{\xi}, \overline{\xi}\}^p,\ \varphi_\lambda(y) \leq C_\mu(y) \tag{3.6}$$

Théorème 3.6.
Les équations (3.5) et (3.6) sont équivalentes :

$$\forall y \in [\underline{\xi}, \overline{\xi}]^p,\ \varphi_\lambda(y) \leq C_\mu(y) \Leftrightarrow \forall y \in \{\underline{\xi}, \overline{\xi}\}^p,\ \varphi_\lambda(y) \leq C_\mu(y)$$

Preuve du Théorème 3.6.
Soit $y \in [\underline{\xi}, \overline{\xi}]^p$ *un vecteur de coûts et* $(.)$ *la fonction de permutation telle que* $y_{(1)} \leq \cdots \leq y_{(p)}$. *On définit* $\beta \in \mathbb{R}^p$ *un vecteur de réels tel que :* $\forall k \in P$, $\beta_{(k)} = \mu(Y_{(k)}) - \mu(Y_{(k+1)}) - \lambda_{(k)}$ *et* $l \in \{1, \ldots, p+1\}$ *un entier tel que :*

- *si* $\min_{t \in P}\{\sum_{k=t}^{p} \beta_{(k)}\} \geq 0$ *alors* $l = p + 1$ *;*
- *si* $\min_{t \in P}\{\sum_{k=t}^{p} \beta_{(k)}\} < 0$ *alors* $l = \arg\min_{t \in P}\{\sum_{k=t}^{p} \beta_{(k)}\}$.

Selon la définition de l *on peut en déduire que :*

$$\forall t \in \{1, \ldots, l-1\}, \sum_{k=t}^{l-1} \beta_{(k)} \geq 0 \tag{3.7}$$

$$\forall t \in \{l, \ldots, p\}, \sum_{k=l}^{t} \beta_{(k)} \leq 0 \tag{3.8}$$

Les inéquations (3.7) et (3.8) sont démontrées par les points a) et b) respectivement :

a) Si $l = p + 1$ *alors* $\min_{t \in P}\{\sum_{k=t}^{p} \beta_{(k)}\} \geq 0$ *et on peut en déduire directement que* $\forall t \in P = \{1, \ldots, l-1\}$, $\sum_{k=t}^{l-1} \beta_{(k)} \geq 0$.

b) Si $l = \arg\min_{t \in P}\{\sum_{k=t}^{p} \beta_{(k)}\}$ *alors*
- $\forall t \in \{1, \ldots, l-1\}$, $\sum_{k=t}^{p} \beta_{(k)} = \sum_{k=l}^{p} \beta_{(k)} + \sum_{k=t}^{l-1} \beta_{(k)} \geq \sum_{k=l}^{p} \beta_{(k)}$ *et donc* $\sum_{k=t}^{l-1} \beta_{(k)} \geq 0$.
- $\forall t \in \{l, \ldots, p\}$, $\sum_{k=t}^{p} \beta_{(k)} = \sum_{k=l}^{p} \beta_{(k)} - \sum_{k=l}^{t} \beta_{(k)} \geq \sum_{k=l}^{p} \beta_{(k)}$ *et donc* $\sum_{k=l}^{t} \beta_{(k)} \leq 0$.

L'inéquation suivante permet de prendre en compte la borne $\underline{\xi}$ *pour les objectifs* (1) *à* $(l-1)$ *:*

$$\sum_{k=1}^{l-1} \beta_{(k)} \times y_{(k)} \geq \left(\sum_{k=1}^{l-1} \beta_{(k)}\right) \times y_{(1)} \geq \left(\sum_{k=1}^{l-1} \beta_{(k)}\right) \times \underline{\xi} \tag{3.9}$$

Notons que $y_{(1)} \geq \underline{\xi}$. *L'inéquation (3.9) est démontrée par induction avec les points c), d) et e) :*

c) Pour $t = l - 1$ *alors on peut en déduire directement l'inéquation suivante :* $\sum_{k=t}^{l-1} \beta_{(k)} \times y_{(k)} \geq \left(\sum_{k=t}^{l-1} \beta_{(k)}\right) \times y_{(l-1)}$.

d) Pour un objectif $t \in \{1, \ldots, l-2\}$ *on suppose que cette inéquation est vérifiée :* $\sum_{k=t}^{l-1} \beta_{(k)} \times y_{(k)} \geq \left(\sum_{k=t}^{l-1} \beta_{(k)}\right) \times y_{(t)}$

e) Pour $t - 1$ *:* $\sum_{k=t-1}^{l-1} \beta_{(k)} \times y_{(k)} \geq \beta_{(t-1)} \times y_{(t-1)} + \left(\sum_{k=t}^{l-1} \beta_{(k)}\right) \times y_{(t)}$ *alors selon (3.7) cette inéquation est aussi vérifiée* $\sum_{k=t-1}^{l-1} \beta_{(k)} \times y_{(k)} \geq \left(\sum_{k=t-1}^{l-1} \beta_{(k)}\right) \times y_{(t-1)}$.

L'inéquation suivante permet de prendre en compte la borne $\overline{\xi}$ *pour les objectifs* (l) *à* (p) *:*

$$\sum_{k=l}^{p} \beta_{(k)} \times y_{(k)} \geq \left(\sum_{k=l}^{p} \beta_{(k)}\right) \times y_{(p)} \geq \left(\sum_{k=l}^{p} \beta_{(k)}\right) \times \overline{\xi} \tag{3.10}$$

Notons que $y_{(p)} \leq \overline{\xi}$. *L'inéquation (3.10) est démontrée par induction avec les points f), g) et h) :*

f) Pour $t = l$ *alors on peut en déduire directement l'inéquation suivante :* $\sum_{k=l}^{t} \beta_{(k)} \times y_{(k)} \geq \left(\sum_{k=l}^{t} \beta_{(k)}\right) \times y_{(l)}$.

g) Pour un objectif $t \in \{l+1, \ldots, p\}$ *on suppose que cette inéquation est vérifiée :* $\sum_{k=l}^{t} \beta_{(k)} \times y_{(k)} \geq \left(\sum_{k=l}^{t} \beta_{(k)}\right) \times y_{(t)}$.

h) Pour $t+1$ *:* $\sum_{k=l}^{t+1} \beta_{(k)} \times y_{(k)} \geq \beta_{(t+1)} \times y_{(t+1)} + \left(\sum_{k=l}^{t} \beta_{(k)}\right) \times y_{(t)}$ *alors selon (3.8) cette inéquation est aussi vérifiée* $\sum_{k=l}^{t+1} \beta_{(k)} \times y_{(k)} \geq \left(\sum_{k=l}^{t+1} \beta_{(k)}\right) \times y_{(t+1)}$.

Le théorème 3.6 est démontré par les points i) et j) :

i) $\{\underline{\xi}, \overline{\xi}\}^p$ *est un sous ensemble de* $[\underline{\xi}, \overline{\xi}]^p$*, alors la première partie (*$\Leftarrow$*) de l'équivalence du théorème 3.6 est vérifiée :*

$$\forall y \in \{\underline{\xi}, \overline{\xi}\}^p,\ 0 \leq C_\mu(y) - \varphi_\lambda(y) \Leftarrow \forall y \in [\underline{\xi}, \overline{\xi}]^p,\ 0 \leq C_\mu(y) - \varphi_\lambda(y)$$

j) Pour tout vecteur de coûts $y \in [\underline{\xi}, \overline{\xi}]^p$*, on définit un deuxième vecteur de coûts* $y' \in \{\underline{\xi}, \overline{\xi}\}^p$ *tel que* $\forall k \in \{1, \ldots, l-1\},\ y'_{(k)} = \underline{\xi}$ *et* $\forall k \in \{l, \ldots, p\},\ y'_{(k)} = \overline{\xi}$*. Selon les inéquations (3.9) et (3.10), l'inéquation* $\sum_{k=1}^{p} \beta_{(k)} \times y_{(k)} \geq \sum_{k=1}^{p} \beta_{(k)} \times y'_{(k)}$ *est vérifiée, alors la deuxième partie (*$\Rightarrow$*) de l'équivalence du théorème 3.6 est vérifiée :*

$$\forall y \in \{\underline{\xi}, \overline{\xi}\}^p,\ 0 \leq C_\mu(y) - \varphi_\lambda(y) \Rightarrow \forall y \in [\underline{\xi}, \overline{\xi}]^p,\ 0 \leq C_\mu(y) - \varphi_\lambda(y)$$

□

Comme précédemment plusieurs vecteurs de poids donnant des bornes de diverses qualités peuvent être obtenues en résolvant le programme linéaire (3.11). Supposons qu'une fonction objectif f permette de sélectionner le meilleur vecteur de poids, on peut construire une somme pondérée φ_λ bornant C_μ en résolvant le problème linéaire (3.11) suivant.

$$\begin{array}{lll} \max & f(\lambda) & \\ \text{s.c.} & \varphi_\lambda(y) \leq C_\mu(y) & \forall y \in [\underline{\xi}, \overline{\xi}] \\ & \lambda_k \in]0, 1] & \forall k \in P \end{array} \tag{3.11}$$

On note que si $\underline{\xi} = 0$ et $0 < \overline{\xi}$ alors les problèmes linéaires (3.3) et (3.11) sont équivalents. Le problème de la détermination d'une fonction objectif pour (3.11) est étudié dans la sous-section 3.2.3.

Exemple 3.6. Des vecteurs de poids pour une somme pondérée bornant l'intégrale de Choquet.
Soient $\varphi_{\lambda^a}, \varphi_{\lambda^b}, \varphi_{\lambda^c}$ *trois sommes pondérées qui bornent inférieurement l'intégrale de Choquet* C_μ*. La table 3.1 donne la fonction de capacité* μ *et les vecteurs de poids* λ^a*,* λ^b*,* λ^c*. Pour calculer les vecteurs de poids la fonction objectif utilisée est la somme des poids :* $\forall y \in \mathbf{\Omega},\ f(y) = \varphi_\lambda(y) = \sum_{k \in P} \lambda_k^a$*. Le vecteur* λ^a *est calculé avec le problème linéaire (3.3). Le vecteur* λ^b *est calculé avec le problème linéaire (3.11), avec des bornes* $\overline{\xi} = 12$ *et* $\underline{\xi} = 69$ *(qui sont calculées avec les points nadir et idéal). Le vecteur* λ^c *est calculé avec le problème linéaire (3.11),* $\underline{\xi} = 20$ *et* $\overline{\xi} = 25$*. L'utilisation de bornes sur les objectifs,* $\underline{\xi}$ *et* $\overline{\xi}$ *permet d'améliorer considérablement la somme des poids. Ainsi la somme des poids de* λ^b *(*$0.3 + 0.074 + 0.274 = 0.648$*), utilisant les bornes sur les objectifs, représente une amélioration de* 30% *par rapport à la somme des poids de* λ^a *(*$0.3 + 0.2 = 0.5$*). Si des bornes plus strictes sur les objectifs peuvent être calculées alors l'amélioration est encore plus importante. Ainsi la somme des poids de* λ^c *est* $0.314 + 0.214 + 0.414 = 0.942$*, ce qui représente une amélioration de* 88% *par rapport à la somme de* λ^a*. On note que dans le cas du vecteur* λ^b*,* $0.3 + 0.079 = 0.379 > \mu(\{1, 2\}) = 0.3$ *ce qui implique que l'équation (3.1) n'est pas respectée. Cet exemple montre que quand la différence entre la borne supérieure (*$\underline{\xi}$*) et la borne inférieure (*$\overline{\xi}$*) sur les objectifs approche* 0 *alors la somme des poids, calculés en utilisant ces bornes avec le problème linéaire (3.11), approche* 1.

L'impact de la qualité de la borne inférieur BORNE_INF^c où la somme pondérée φ_λ bornant la fonction d'utilité Ψ_μ^u calculée avec le programme linéaire (3.11) est évaluée dans le paragraphe suivant.

ensemble	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	P
μ	0	0.3	0.2	0.2	0.3	0.5	0.4	1
λ^a	–	0.3	0.0	0.2	–	–	–	–
λ^b	–	0.3	0.079	0.279	–	–	–	–
λ^c	–	0.314	0.214	0.414	–	–	–	–

Table 3.1 – Une fonction de capacité μ définie sur chaque ensemble d'objectifs et des vecteurs de poids λ^a, λ^b, λ^c définis sur chaque objectif. Chaque somme pondérée φ_{λ^a}, φ_{λ^b}, φ_{λ^c} borne inférieurement l'intégrale de Choquet C_μ.

3.2.2.1 Évaluations numériques

Dans le but d'évaluer l'utilisation de bornes sur les objectifs on considère des graphes construit par le générateur *ggen*. Ce générateur a été utilisé pour évaluer l'algorithme RMC [GBR06] développé en collaboration avec Alcatel. Les graphes générés ont des coûts positifs sur les arcs définis entre 1 et 10. Le nombre de nœuds des graphes est : $n = 1000$ ou 3000 ou 6000 ou 10000. Le nombre d'arcs des graphes est défini selon le nombre de nœuds du graphe tel que : $m = 5$ ou 50 ou 500 $\times n$. Nous définissons la densité du graphe comme le nombre moyen d'arcs pour chaque nœud, i.e. 5 ou 50 ou 500. Pour chaque combinaison de taille et de densité, un ensemble de 20 graphes a été générer et est utilisé pour l'expérimentation. Nous ne testons pas les combinaisons de 10000 nœuds avec $m = 50 \times n$ ou $m = 500 \times n$ à cause de son coût prohibitif en espace mémoire.

Notre problème consiste à construire un chemin, dans les graphes définis ci-dessus, minimisant une intégrale de Choquet. Nous proposons de tester nos propositions sur 7 intégrales de Choquet : 4 intégrales de Choquet $C_{\mu^a}, C_{\mu^b}, C_{\mu^c}, C_{\mu^d}$ définies sur 3 objectifs (cf. Table 3.2, de la présente page) et 3 intégrales de Choquet $C_{\mu^e}, C_{\mu^f}, C_{\mu^g}$ définies sur 5 objectifs (cf. Table 3.3, de la présente page).

ensemble	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	P
μ^a	0.0	0.2	0.2	0.1	0.6	0.7	0.7	1.0
μ^b	0.0	0.3	0.1	0.2	0.9	0.3	0.8	1.0
μ^c	0.0	0.5	0.7	0.6	0.8	0.9	0.8	1.0
μ^d	0.0	0.4	0.3	0.3	0.4	0.4	0.3	1.0

Table 3.2 – Fonctions de capacité pour l'intégrale de Choquet avec 3 objectifs.

ensemble	$\lvert A\rvert = 0$	$\lvert A\rvert = 1$	$\lvert A\rvert = 2$	$\lvert A\rvert = 3$	$\lvert A\rvert = 4$	$\lvert A\rvert = 5$
μ^e	0.0	0.0	0.0	0.0	0.0	1.0
μ^f	0.0	1.0	1.0	1.0	1.0	1.0
μ^g	0.0	0.3	0.4	0.4	0.4	1.0

Table 3.3 – Fonctions de capacité pour l'intégrale de Choquet avec 5 objectifs.

L'algorithme d'étiquetage général (cf. Algorithme 2.1, page 51) est utilisé, la fonction de sélection est identique à celle utilisée dans l'algorithme de Martins (cf. §2.2.2, page 55) : une fonction lexicographique. Dans cet algorithme la règle de coupe de la définition 3.1 avec la borne inférieure BORNE_INF^c (cf. Définition 3.8, page 80) est utilisée afin de prendre en compte la fonction d'utilité. La

somme pondérée bornant inférieurement l'intégrale de Choquet est construite durant l'initialisation de l'algorithme d'étiquetage avec les programmes linéaires (LP1-$\sum$) ou (LP2-$\sum$) :

- (LP1-$\sum$) : utilisation du programme linéaire (3.3) avec une fonction objectif somme $f(\lambda) = \sum_{k=1}^{p} \lambda_k$.
- (LP2-$\sum$) : utilisation du programme linéaire (3.11) avec une fonction objectif somme $f(\lambda) = \sum_{k=1}^{p} \lambda_k$. Les bornes $\underline{\xi}$ et $\overline{\xi}$ sur les objectifs sont calculées à partir du point idéal et d'une approximation du point nadir.

Les tables 3.4 et 3.5 montrent les temps de calcul pour l'algorithme d'étiquetage décrit ci dessus, pour l'optimisation des intégrales de Choquet $C_{\mu^a}, C_{\mu^b}, C_{\mu^c}, C_{\mu^d}$ et $C_{\mu^e}, C_{\mu^f}, C_{\mu^g}$ où les sommes pondérées bornant ces intégrales de Choquet sont construites selon (LP1-$\sum$) ou (LP2-$\sum$).

préférences		temps de calcul (s)										déviation (%)	
		$m = 5 \times n$				$m = 50 \times n$			$m = 500 \times n$				
fct. cap.	programme linéaire	1000	3000	6000	10000	1000	3000	6000	1000	3000	6000	max	moy.
C_{μ^a}	(LP1-$\sum$)	0.087	0.240	2.56	3.60	0.24	1.22	4.21	2.86	6.79	21.87	652	241
	(LP2-$\sum$)	0.027	0.085	0.34	0.55	0.09	0.47	1.33	1.19	4.10	14.03	0.0	0.0
C_{μ^b}	(LP1-$\sum$)	0.300	1.850	9.40	32.92	0.47	8.84	29.87	7.93	32.44	137.41	374	125
	(LP2-$\sum$)	0.140	0.390	3.50	7.95	0.37	5.56	17.81	5.97	21.55	95.40	0.0	0.0
C_{μ^c}	(LP1-$\sum$)	0.011	0.030	0.07	0.13	0.08	0.27	0.65	0.89	3.90	11.86	0.0	0.0
	(LP2-$\sum$)	0.014	0.035	0.08	0.14	0.09	0.28	0.71	0.93	4.05	12.09	27.27	10.16
C_{μ^d}	(LP1-$\sum$)	0.043	0.106	0.48	0.72	0.08	0.30	1.33	1.25	5.18	12.59	300	98.9
	(LP2-$\sum$)	0.016	0.053	0.12	0.23	0.05	0.25	0.82	1.05	4.40	9.75	0.0	0.0
Algorithme de Martins		0.466	4.657	19.00	63.99	2.35	26.54	99.26	9.80	58.63	204.18	–	–

Table 3.4 – Expérimentations sur des problèmes de plus court chemin d'un nœud à un nœud ($R_{s,t}$) avec 3 objectifs additifs. Temps de calcul en secondes pour l'algorithme d'étiquetage 2.1 avec la règle de coupe 3.1 et la borne inférieure 3.8 où les sommes pondérées bornant les intégrales de Choquet $C_{\mu^a}, C_{\mu^b}, C_{\mu^c}, C_{\mu^d}$ sont construites selon les programmes linéaires (LP1-$\sum$) ou (LP2-$\sum$) et pour l'algorithme de Martins (cf. §2.2.2, page 55). La déviation est le pourcentage de temps de calcul supplémentaire pour chaque programme linéaire et chaque intégrale de Choquet par rapport au meilleur temps de calcul entre (LP1-$\sum$) et (LP2-$\sum$) pour la même intégrale de Choquet.

Les temps de calcul montrent que l'ajout de la règle de coupe 3.1 avec la borne inférieure 3.8 dans l'algorithme d'étiquetage permet une amélioration importante relativement au même algorithme sans la règle de coupe, i.e. l'algorithme de Martins. On note que l'utilisation de cette règle de coupe est moins intéressante quand l'intégrale de Choquet C_μ est concave que quand l'intégrale de Choquet est convexe (cf. Définition 1.15, page 32). On note que dans ce cas, la somme des poids de la somme pondérée bornant l'intégrale de Choquet est inférieure à 1. L'utilisation de bornes sur les objectifs (LP2-$\sum$) permet alors d'améliorer de manière significative les temps de calcul par rapport à (LP1-$\sum$). Quand la fonction de capacité est submodulaire (μ^c) alors l'utilisation de bornes sur les objectifs (LP2-$\sum$) souffre d'un temps de calcul légèrement plus important par rapport à (LP1-$\sum$) à cause du temps de calcul des bornes $\underline{\xi}$ et $\overline{\xi}$.

3.2.3 Fonctions objectif d'un problème linéaire pour construire une somme pondérée

Les problèmes linéaires (3.3) et (3.11) définissent le domaine de définition des poids, mais clairement tous les vecteurs de poids respectant les contraintes de ces problèmes linéaires ne sont pas équivalents (cf. Exemple 3.7, page 92). Pour cette raison il est important d'utiliser la bonne fonction objectif afin

préférences		temps de calcul (s) $m = 5 \times n$				déviation (%)	
fct. cap.	problème linéaire	1000	3000	6000	10000	max	moy.
C_{μ^e}	(LP1-$\sum$)	1.04	10.28	45.10	148.13	1.21	0.28
	(LP2-$\sum$)	1.06	10.63	44.60	149.24	3.40	1.52
C_{μ^f}	(LP1-$\sum$)	0.02	0.10	0.19	0.47	0.0	0.0
	(LP2-$\sum$)	0.02	0.10	0.20	0.49	5.26	2.38
C_{μ^g}	(LP1-$\sum$)	0.04	0.51	0.76	1.56	240	167
	(LP2-$\sum$)	0.02	0.15	0.33	0.52	0.0	0.0
Algorithme de Martins		1.24	12.66	57.59	197.78	–	–

Table 3.5 – Expérimentations sur des problèmes de plus court chemin d'un nœud à un nœud ($R_{s,t}$) avec 5 objectifs additifs. Temps de calcul en secondes pour l'algorithme d'étiquetage 2.1 avec la règle de coupe 3.1 et la borne inférieure 3.8 où les sommes pondérées bornant les intégrales de Choquet $C_{\mu^e}, C_{\mu^f}, C_{\mu^g}$ sont construites selon les programmes linéaires (LP1-$\sum$) ou (LP2-$\sum$) et pour l'algorithme de Martins (cf. §2.2.2, page 55). La déviation est le pourcentage de temps de calcul supplémentaire par rapport à la meilleure méthode (LP1-$\sum$) ou (LP2-$\sum$) pour construire la somme pondérée bornant une même intégrale de Choquet.

de sélectionner le vecteur de poids qui permettra de supprimer un maximum de labels dans la règle de coupe 3.1.

La borne inférieure BORNE_INF^c (cf. Définition 3.8, page 80) doit être la plus haute possible pour chaque label. Alors chaque poids λ_k, $k \in P$, de la somme pondérée φ_λ utilisée dans cette borne inférieure doit être maximisé. Il faut donc construire un vecteur de poids λ qui soit Pareto efficace par rapport aux autres vecteurs de poids admissibles selon les problèmes linéaires (3.3) ou (3.11). La fonction $f : [0,1]^p \to \mathbb{R}_+$ doit alors être strictement croissante (cf. Définition 1.9, page 18) de manière à calculer un vecteur de poids efficace. La fonction f doit de plus être linéaire de manière à être intégrée dans les programmes linéaires (3.3) ou (3.11). Les fonctions objectif proposées † ci dessous sont toutes croissantes et linéaires. Comme la fonction objectif f doit être maximisée, alors elle peut contenir des agrégations $\min$ et $\sum$.

- La somme des poids :
$$f(\lambda) = \sum_{k=1}^{p} \lambda_k \tag{3.12}$$
La somme des poids est la fonction objectif la plus simple proposée, elle est utilisée comme fonction de référence.
- Une agrégation minimale plus une somme marginale :
$$f(\lambda) = \min_{k \in P} \lambda_k + \epsilon \times \sum_{k=1}^{p} \lambda_k \tag{3.13}$$
$\epsilon > 0$ une petite valeur positive. Cette fonction permet de calculer un vecteur équilibré de poids grâce à l'agrégation $\min$. La fonction objectif contient une somme pondérée marginale, ainsi la fonction objectif est strictement croissante.
- Une agrégation minimale par rapport à l'indice de Shapley plus une somme marginale :
$$f(\lambda) = \min_{k \in P}(\lambda_k - \phi_k(\mu)) + \epsilon \times \sum_{k=1}^{p} \lambda_k \tag{3.14}$$

†. Ces travaux ont fait l'objet d'une publication [FGL11a].

avec $\epsilon > 0$ une petite valeur positive et $\phi(\mu)$ le vecteur de valeurs de Shapley (cf. Définition 1.20, page 35). L'utilisation des indices de Shapley dans la fonction objectif permet de prendre en compte le fait que les objectifs les plus importants pour l'intégrale de Choquet doivent aussi être plus importants dans la somme pondérée. Ainsi la somme pondérée φ_λ est plus proche de l'intégrale de Choquet C_μ.

- Une somme de poids plus une fonction d'agrégation minimale marginale :

$$f(\lambda) = \epsilon \times \min_{k \in P}(\lambda_k - \phi_i) + \sum_{k=1}^{p} \lambda_k \tag{3.15}$$

avec $\epsilon > 0$ une petite valeur positive et $\phi(\mu)$ le vecteur de valeurs de Shapley (cf. Définition 1.20, page 35). Cette fonction permet, grâce à l'agrégation $\min$, d'obtenir un vecteur de poids équilibré parmi l'ensemble des vecteurs de poids qui maximisent la somme des poids.

Exemple 3.7. Analyse des différentes fonctions objectif pour calculer les vecteurs de poids.
Soient φ_{λ^a},φ_{λ^b},φ_{λ^c} trois sommes pondérées qui bornent inférieurement l'intégrale de Choquet C_μ. La table 3.6 donne les fonctions de capacité μ et les vecteurs de poids λ^a, λ^b, λ^c. Le vecteur λ^a est calculé avec le problème linéaire (3.3) et la fonction objectif (3.12). Le vecteur λ^b est calculé avec le problème linéaire (3.3) et la fonction objectif (3.13). Le vecteur λ^c est calculé avec le problème linéaire (3.3) et la fonction objectif (3.14). Le vecteur de poids λ^b est plus équilibré que les vecteurs de poids λ^a et λ^c. De plus l'objectif 2 n'est pas pris en compte dans le vecteur de poids λ^a. Comme supposé, le vecteur de poids λ^c est le plus proche des indices de Shapley.

ensemble	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	P
μ	0	0.3	0.2	0.2	0.3	0.5	0.4	1
λ^a	–	0.3	0.0	0.2	–	–	–	–
λ^b	–	0.15	0.15	0.2	–	–	–	–
λ^c	–	0.2	0.1	0.2	–	–	–	–
$\phi(\mu)$	–	0.36	0.26	0.36	–	–	–	–

Table 3.6 – Une fonction de capacité μ définie sur chaque ensemble d'objectifs et des vecteurs de poids λ^a, λ^b, λ^c définis sur chaque objectif. Chaque somme pondérée φ_{λ^a}, φ_{λ^b}, φ_{λ^c} borne inférieurement l'intégrale de Choquet C_μ. $\phi(\mu)$ est l'indice de Shapley de μ.

3.2.3.1 Évaluations numériques

Dans le but d'évaluer l'utilisation des différentes fonctions objectif (3.12), (3.13), (3.14) et (3.15) on considère l'optimisation des différentes intégrales de Choquet présentées précédemment (cf. §3.2.2.1, page 89) : les intégrales de Choquet $C_{\mu^a}, C_{\mu^b}, C_{\mu^c}, C_{\mu^d}$ pour 3 objectifs (cf. Figure 3.2, page 89) et les intégrales de Choquet $C_{\mu^e}, C_{\mu^f}, C_{\mu^g}$ pour 5 objectifs (cf. Figure 3.3, page 89). Les graphes utilisés sont identiques à ceux présentés dans le paragraphe 3.2.2.1.

L'algorithme d'étiquetage utilisé est identique à celui décrit dans le paragraphe 3.2.2.1. Pour construire les sommes pondérées bornant inférieurement les intégrales de Choquet $C_{\mu^a}, C_{\mu^b}, C_{\mu^c}, C_{\mu^d}$ et $C_{\mu^e}, C_{\mu^f}, C_{\mu^g}$ on considère uniquement la résolution du programme linéaire (3.11) L'utilisation de ce programme linéaire permettant de meilleurs temps de calcul en général que l'utilisation du programme linéaire (3.3). Les bornes $\underline{\xi}$ et $\overline{\xi}$ sur les objectifs sont calculées à partir du point idéal et d'une approximation du point nadir. Les différentes fonctions objectif (3.12), (3.13), (3.14) et (3.15) permettent de définir plusieurs programmes linéaires :

- $(\sum)$: utilisation du programme linéaire (3.11) avec la fonction objectif somme (3.12).
- (min) : utilisation du programme linéaire (3.11) avec la fonction objectif minimale (3.13).
- $(\min\text{-}\epsilon\text{-}\sum)$: utilisation du programme linéaire (3.11) avec la fonction objectif (3.14).
- $(\sum\text{-}\epsilon\text{-}\min)$: utilisation du programme linéaire (3.11) avec la fonction objectif (3.15).

préférences		temps de calcul (s)										déviation (%)	
		$m = 5 \times n$				$m = 50 \times n$			$m = 500 \times n$				
fct. cap.	programme linéaire	1000	3000	6000	10000	1000	3000	6000	1000	3000	6000	max	moy.
C_{μ^a}	$(\sum)$	0.029	0.081	0.40	0.61	0.095	0.48	1.38	1.28	4.39	14.29	2.40	0.86
	(min)	0.193	0.874	10.95	19.56	0.247	10.20	24.77	5.32	12.92	104.40	3106	1234
	$(\min\text{-}\epsilon\text{-}\sum)$	0.185	0.863	12.38	21.80	0.246	10.18	25.34	5.31	13.08	106.60	3473	1308
	$(\sum\text{-}\epsilon\text{-}\min)$	0.032	0.083	0.43	0.61	0.093	0.51	1.42	1.25	4.30	14.01	10.34	2.94
C_{μ^b}	$(\sum)$	0.178	0.446	7.71	13.87	0.403	6.22	18.25	5.97	21.55	95.40	97.54	27.64
	(min)	0.202	0.817	13.34	20.49	0.489	7.94	22.78	5.16	18.36	89.88	139.7	57.9
	$(\min\text{-}\epsilon\text{-}\sum)$	0.474	4.904	18.50	59.51	1.743	15.88	67.26	11.93	63.95	239.79	1126	386
	$(\sum\text{-}\epsilon\text{-}\min)$	0.153	0.400	7.11	12.81	0.204	5.61	13.70	5.04	19.24	61.76	4.79	0.48
C_{μ^c}	$(\sum)$	0.018	0.056	0.13	0.22	0.091	0.31	0.70	0.97	4.25	12.09	8.33	1.15
	(min)	0.018	0.056	0.12	0.22	0.091	0.31	0.70	0.95	4.30	12.30	1.73	0.39
	$(\min\text{-}\epsilon\text{-}\sum)$	0.018	0.057	0.12	0.22	0.091	0.31	0.70	0.98	4.84	12.50	13.88	2.33
	$(\sum\text{-}\epsilon\text{-}\min)$	0.018	0.058	0.12	0.22	0.091	0.31	0.70	0.94	5.08	12.87	19.52	2.95
C_{μ^d}	$(\sum)$	0.021	0.059	0.16	0.26	0.092	0.31	0.77	1.05	4.40	11.75	8.64	2.04
	(min)	0.026	0.069	0.23	0.37	0.092	0.33	0.85	1.39	4.78	12.96	48	21.98
	$(\min\text{-}\epsilon\text{-}\sum)$	0.020	0.060	0.16	0.26	0.092	0.32	0.77	1.21	4.18	12.14	17.47	3.38
	$(\sum\text{-}\epsilon\text{-}\min)$	0.020	0.059	0.16	0.25	0.092	0.32	0.77	1.03	4.05	11.65	3.22	0.32
Algorithme de Martins		0.466	4.657	19.00	63.99	2.35	26.54	99.26	9.80	58.63	204.18	–	–

Table 3.7 – Expérimentations sur des problèmes de plus court chemin d'un nœud à un nœud ($R_{s,t}$) avec 3 objectifs additifs. Temps de calcul en secondes pour l'algorithme d'étiquetage 2.1 avec la règle de coupe 3.1 et la borne inférieure 3.8 où les sommes pondérées bornant les intégrales de Choquet $C_{\mu^a}, C_{\mu^b}, C_{\mu^c}, C_{\mu^d}$ sont construites par la résolution des programmes linéaires $(\sum)$, (min), $(\min\text{-}\epsilon\text{-}\sum)$, $(\sum\text{-}\epsilon\text{-}\min)$ et pour l'algorithme de Martins (cf. §2.2.2, page 55). La déviation est le pourcentage de temps de calcul supplémentaire par rapport au meilleur programme linéaire utilisé pour construire la somme pondérée bornant une même intégrale de Choquet.

Pour les intégrales de Choquet C_{μ^c}, C_{μ^d}, C_{μ^e}, C_{μ^g} les différentes fonctions objectif proposées pour construire la somme pondérée induisent des temps de calcul similaires. Pour l'intégrale de Choquet C_{μ^a} l'utilisation des programmes linéaires (min) et $(\min\text{-}\epsilon\text{-}\sum)$ induit des temps de calcul moins bon que les temps de calcul induit par les programmes linéaires $(\sum)$ et $(\sum\text{-}\epsilon\text{-}\min)$. Pour l'intégrale de Choquet C_{μ^b} l'utilisation des programmes linéaires $(\sum)$, (min) et $(\min\text{-}\epsilon\text{-}\sum)$ induit des temps de calcul moins bon que les temps de calcul induit par le programme linéaire $(\sum\text{-}\epsilon\text{-}\min)$. Pour l'intégrale de Choquet C_{μ^f} l'utilisation du programme linéaire $(\sum)$ induit des temps de calcul moins bon que les temps de calcul induit par les autres programmes linéaires (min), $(\min\text{-}\epsilon\text{-}\sum)$ et $(\sum\text{-}\epsilon\text{-}\min)$. Ces différences de temps de calcul mettent en évidence deux problème majeur pour le calcul d'un vecteur de poids λ telle que la somme pondérée φ_λ borne inférieurement une intégrale de Choquet C_μ :

- L'utilisation des programmes linéaires (min) et $(\min\text{-}\epsilon\text{-}\sum)$ n'est pas efficace à cause de la proéminence de l'agrégation $\min$ dans ces deux fonctions. L'exemple 3.8 illustre ce problème.
- L'utilisation du programme linéaire $(\sum)$ n'est pas efficace quand la fonction de capacité est submodulaire (cf. Définition 1.18, page 34) sur de nombreux ensembles d'objectifs. Pour une telle intégrale

préférences		temps de calcul (s) $m = 5 \times n$				déviation (%)	
fct. cap.	problème linéaire	1000	3000	6000	10000	max	moy.
C_{μ^e}	($\sum$)	1.06	10.63	44.60	143.53	6.77	1.70
	(min)	1.17	12.23	47.36	134.43	15.05	7.91
	(min-ϵ-$\sum$)	1.06	10.87	44.90	141.37	5.16	2.03
	($\sum$-ϵ-min)	1.10	11.12	46.99	144.30	7.34	5.27
C_{μ^f}	($\sum$)	0.036	0.20	0.82	0.64	355	163
	(min)	0.025	0.08	0.18	0.33	4.17	1.82
	(min-ϵ-$\sum$)	0.024	0.08	0.19	0.32	5.57	1.39
	($\sum$-ϵ-min)	0.025	0.08	0.19	0.32	5.57	2.43
C_{μ^g}	($\sum$)	0.029	0.10	0.37	0.47	4.44	1.81
	(min)	0.029	0.10	0.37	0.46	2.78	1.25
	(min-ϵ-$\sum$)	0.030	0.10	0.36	0.45	3.48	0.86
	($\sum$-ϵ-min)	0.031	0.10	0.37	0.47	6.89	3.53
Algorithme de Martins		1.24	12.66	57.59	197.78	–	–

Table 3.8 – Expérimentations sur des problèmes de plus court chemin d'un nœud à un nœud ($R_{s,t}$) avec 5 objectifs additifs. Temps de calcul en secondes pour l'algorithme d'étiquetage 2.1 avec la règle de coupe 3.1 et la borne inférieure 3.8 où les sommes pondérées bornant les intégrales de Choquet $C_{\mu^e}, C_{\mu^f}, C_{\mu^g}$ sont construites par la résolution des programmes linéaires ($\sum$), (min), (min-ϵ-$\sum$), ($\sum$-ϵ-min) et pour l'algorithme de Martins (cf. §2.2.2, page 55). La déviation est le pourcentage de temps de calcul supplémentaire par rapport au meilleur programme linéaire utilisé pour construire la somme pondérée bornant une même intégrale de Choquet.

de Choquet les poids $\lambda_1, \ldots, \lambda_p$ calculés grâce au programme linéaire ($\sum$) ne sont pas équilibrés (cf. Exemple 3.7, page 92). Par exemple pour une intégrale de Choquet C_μ telle que $\forall A \subseteq P$, $A \neq \emptyset$, $\mu(A) = 1$ alors le programme linéaire ($\sum$) donnera un vecteur de poids $\lambda = (1, 0, \ldots, 0)$ totalement déséquilibré. Alors la règle de coupe 3.1 utilisant la borne inférieure 3.8 ne permet de supprimer que des labels qui ont un coût élevé sur le premier objectif.

Ainsi le programme linéaire ($\sum$-ϵ-min) est ainsi le seul programme linéaire qui permette à l'algorithme d'étiquetage de ne souffrir d'aucune perte importante de performance quelque soit l'intégrale de Choquet optimisée.

Exemple 3.8. Problème lié à l'utilisation d'une fonction objectif min.
Soient une borne inférieure $\underline{\xi} = 10$ *et une borne supérieure* $\overline{\xi} = 100$ *sur les 3 objectifs additifs et une fonction de capacité* μ *telle que* $\mu(\emptyset) = 0$, $\mu(\{1\}) = 0.6$, $\mu(\{2\}) = 0.1$, $\mu(\{3\}) = 0.3$, $\mu(\{1,2\}) = 0.9$, $\mu(\{1,3\}) = 1.0$, $\mu(\{2,3\}) = 0.4$, $\mu(\{1,2,3\}) = 1.0$. *Le programme linéaire (3.11) permettant de construire les vecteurs de poids* λ, λ' *est défini selon* μ, $\underline{\xi}$, $\overline{\xi}$ *et possède la contrainte suivante :* $\lambda_1 \times 10 + \lambda_2 \times 100 + \lambda_3 \times 10 \leq C_\mu((10, 200, 10)) = 100 \times 0.1 + 10 \times 0.1 + 10 \times 0.9 = 19$. *Considérons que l'on cherche à optimiser une fonction* min *dans le programme linéaire (3.11) alors on cherche un vecteur de poids équilibré. Un vecteur de poids parfaitement équilibré est* $\lambda_1 = \lambda_2 = \lambda_3 \approxeq 0.158$. *Il n'existe pas de meilleur vecteur de poids parfaitement équilibré car :* $10 \times 0.158 + 100 \times 0.158 + 10 \times 0.158 \approxeq 19$. *Considérons que l'on cherche à optimiser une fonction* $\sum$ *dans le programme linéaire (3.11) alors on peut obtenir :* $\lambda'_1 = \lambda'_3 = 0.45$ *et* $\lambda'_3 = 0.1$. *En effet ce vecteur de poids vérifie la contrainte :* $0.45 \times 10 + 0.1 \times 100 + 0.45 \times 10 = 19$. *Ce deuxième vecteur de poids* (λ') *bien que moins équilibré que le premier vecteur de poids* (λ) *donne pour la plupart des labels une meilleure borne inférieure 3.8 et permet donc de supprimer plus de labels que le premier vecteur de poids. Notons que ce problème est dû à l'utilisation de bornes* $\underline{\xi}$,$\overline{\xi}$ *sur les objectifs.*

3.2.4 Définition de fonctions linéaires ou affines bornant les fonctions d'utilité partielles

Nous avons vu précédemment comment construire une somme pondérée $\varphi_{\lambda'}$ qui borne inférieurement l'intégrale de Choquet C_μ qui est la fonction d'agrégation de la fonction d'utilité Ψ^u_μ. Afin de proposer une somme pondérée φ_λ, qui borne inférieurement la fonction d'utilité Ψ^u_μ, il faut pouvoir prendre en compte les fonctions d'utilité partielles. Ceci implique de calculer des fonctions linéaires bornant inférieurement les fonctions d'utilité partielles. Ces fonctions peuvent alors être définies de la manière suivante : $\forall y \in \mathbf{\Omega}, \forall k \in P, \underline{u}_k(y_k) = \alpha_k \times y_k$. Elle doivent aussi borner les fonctions d'utilité partielles telles que : $\forall y \in \mathbf{\Omega}, \forall k \in P, \underline{u}_k(y) \leq u_k(y)$. Alors la somme pondérée φ_λ bornant Ψ^u_μ est définie telle que : $\forall k \in P, \lambda_k = \lambda'_k \times \alpha_k$. Dans le paragraphe 3.2.4.1 on propose une méthode pour construire des fonctions linéaires bornant des fonctions d'utilité partielles linéaires par morceaux.

Dans le cadre du problème MOSP Q-$\sum$ une somme pondérée φ_λ est linéaire et donc monotone. Une fonction fonction affine : $\underline{U} : \mathbf{\Omega} \to \xi, \underline{U}(y) = \varphi_\lambda(y) + \boldsymbol{\gamma}$ composé d'une somme pondérée φ_λ et d'une valeur fixe $\gamma \in \mathbb{R}$ est aussi monotone. En effet selon le théorème 3.2 la fonction affine $\underline{U}$ est monotone car elle peut être décomposée par une fonction d'agrégation linéaire φ_λ et une fonction croissante g définie telle que $\forall y \in \xi, g(y) = y + \boldsymbol{\gamma}$. Ainsi il est possible d'utiliser une fonction affine $\underline{U}$ bornant la fonction d'utilité $U = \Psi^u_\mu$ composé d'une intégrale de Choquet et de fonctions d'utilité partielles $u_1, \ldots, u_p$ pour calculer la borne inférieure BORNE_INF^c (cf. Définition 3.8, page 80). Nous avons vu précédemment comme construire une somme pondérée $\varphi_{\lambda'}$ bornant l'intégrale de Choquet : $\forall y \in \mathbf{\Omega}$, $\varphi_\lambda(y) \leq C_\mu(y)$. Dans le paragraphe 3.2.4.2 nous montrons comment construire des fonctions affines ($\underline{u}_k(y) = \alpha_k \times y_k + \gamma_k, \gamma_k \in \mathbb{R}$) bornant les fonctions d'utilité partielles : $\forall y \in \mathbf{\Omega}, \underline{u}_k(y) \leq u_k(y)$. Ces fonctions affines peuvent alors être utilisées pour calculer les bornes inférieures BORNE_INF^c (cf. Définition 3.8, page 80) et BORNE_INF^d (cf. Définition 3.9, page 82) de la même façon que des fonctions linéaires. Alors la fonction affine $\underline{U} = \varphi_{\lambda'} + \boldsymbol{\gamma}$ bornant Ψ^u_μ est définie telle que : $\lambda' = \lambda \times \lambda$ et $\boldsymbol{\gamma} = \sum_{k=1}^{p} \lambda_k \times \gamma_k$.

3.2.4.1 Construction de fonctions linéaires bornant les fonctions d'utilité partielles

Soit un objectif $k \in P$ et une fonction linéaire par morceaux u_k. Soit Θ_k l'ensemble des coefficients d'inclinaison de la fonction d'utilité linéaire par morceaux u_k. Si le degré d'inclinaison α_k de la fonction linéaire $\underline{u}_k(y_k) = \alpha_k \times y_k$ est plus petit que le plus petit degré d'inclinaison de la fonction linéaire par morceaux u_k :

$$\alpha_k = \min_{\theta_k \in \Theta_k} \theta_k \tag{3.16}$$

alors la fonction linéaire $\underline{u}_k$ est par définition inférieure à u_k pour chaque valeur $y_k \in \mathbb{R}_+ : \underline{u}_k(y_k) = \theta_k \times y_k \leq u_k(y_k)$.

Exemple 3.9. Fonction linéaire bornant une fonction d'utilité partielle.
Soit une fonction d'utilité linéaire par morceaux u_k définie telle que $\overline{z}^1_k = 0$, $\overline{z}^2_k = 0.5$, $\overline{z}^3_k = 1$, $\overline{z}^4_k = 1.5$, $\overline{z}^5_k = 2$ et $\Theta_k = \{0.5, 2, 1, 0.25\}$. La fonction linéaire $\underline{u}_k(y_k) = \alpha_k \times y_k$ bornant u_k, calculée grâce à l'équation (3.16), est définie par $\alpha_k = 0.25$. Ces deux fonctions u_k et $\underline{u}_k$ sont illustrées sur la figure 3.13.

3.2.4.2 Construction de fonctions affines bornant les fonctions d'utilité partielles

Considérons un objectif $k \in N$ avec $u_k : \Omega_k \to \xi$ la fonction d'utilité partielle associée. Soient $\underline{\xi}_k$ et $\overline{\xi}_k$ des bornes supérieures et inférieures sur chaque objectif $k \in P$ définie par rapport à une solution optimale $\mathbf{r}_{s,t} \in R_{s,t}$ selon U telle que : $\underline{\xi}_k \leq z_k(\mathbf{r}_{s,t})$ et $\overline{\xi}_k \geq z_k(\mathbf{r}_{s,t})$.

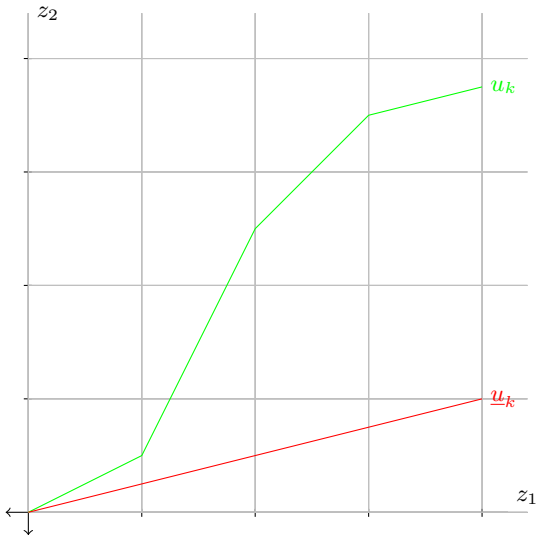

Figure 3.13 – Une fonction linéaire (−) bornant inférieurement une fonction d'utilité partielle (−) linéaire par morceaux.

Soit u_k une fonction d'utilité partielle linéaire par morceaux et croissante (cf. Définition 1.12, page 28), τ_k est le nombre de segments de la fonction d'utilité u_k. Les segments sur lesquels sont définis la fonction u_k sont : $[z_k^1, z_k^2], \ldots, [z_k^{\tau_k}, z_k^{\tau_k+1}]$. Avec $z_k^1, \ldots, z_k^{\tau_k}$ les bornes des segments. Une fonction affine $\underline{u}_k(y_k) = \alpha_k \times y_k + \gamma_k$ bornant inférieurement u_k uniquement entre $\underline{\xi}_k$ et $\overline{\xi}_k$. Soient $\overline{z}_k^{\underline{l}_k}$ et $\overline{z}_k^{\bar{l}_k}$ deux bornes des segments de u_k telles que $[\overline{z}_k^{\underline{l}_k}, \overline{z}_k^{\bar{l}_k}]$ est le plus grand sous-segment de $[\underline{\xi}_k, \overline{\xi}_k]$, i.e. :

$$\begin{aligned} \underline{l}_k &= \min\{l : u_k(\underline{\xi}_k) \leq u_k(\overline{z}_k^l), l \in \{1, \ldots, \tau_k\}\} \\ \bar{l}_k &= \max\{l : u_k(\overline{z}_k^l) \leq u_k(\overline{\xi}_k), l \in \{1, \ldots, \tau_k\}\} \end{aligned}$$

Alors la fonction affine $\underline{u}_k = \alpha_k \times y_k + \gamma_k$ bornant u_k peut être construite avec le problème linéaire (3.17) suivant :

$$\begin{array}{lll} \max & \alpha_k \times ((\underline{\xi}_k + \overline{\xi}_k)/2) + \gamma_k & (a) \\ \text{s.c.} & \alpha_k \times \overline{z}_k^l + \gamma_k \leq u_k(\overline{z}_k^l) \quad \forall l \in \{\underline{l}_k, \ldots, \bar{l}_k\} & (b) \\ & \alpha_k \times \overline{\xi}_k + \gamma_k \leq u_k(\overline{\xi}_k) & (c) \\ & \alpha_k \times \underline{\xi}_k + \gamma_k \leq u_k(\underline{\xi}_k) & (d) \\ & \alpha_k \in \mathbb{R}_+ & \\ & \gamma_k \in \mathbb{R} & \end{array} \tag{3.17}$$

Soit $[\overline{z}_k^l, \overline{z}_k^{l+1}]$ un segment de u_k, u_k est donc linéaire entre $\overline{z}_k^l$ et $\overline{z}_k^{l+1}$. Pour s'assurer que $\underline{u}_k$, qui est une fonction affine, est inférieure à u_k sur le segment $[\overline{z}_k^l, \overline{z}_k^{l+1}]$ alors il suffit de vérifier que $\underline{u}_k$ est inférieure à u_k sur les extrémités $\overline{z}_k^l$ et $\overline{z}_k^{l+1}$ de ce segment. Ainsi le problème linéaire (3.17) exploite le fait qu'il suffit de vérifier que $\underline{u}_k$ borne inférieurement u_k uniquement sur les extrémités des segments, car u_k est linéaire par morceaux. La fonction objectif (a) permet au problème linéaire (3.17) de calculer la fonction affine $\underline{u}_k$ maximale sur la valeur $(\underline{\xi}_k + \overline{\xi}_k)/2$ qui est située entre les deux bornes $\underline{\xi}_k$ et $\overline{\xi}_k$. Alors la fonction d'agrégation $\underline{U}$ composée de la fonction affine $\underline{u}_k$ sera la plus proche possible de la fonction d'utilité U quand la valeur sur le k-ème objectif atteint $(\underline{\xi}_k + \overline{\xi}_k)/2$. Ceci permet d'une certaine manière de maximiser la borne inférieure BORNE_INF^c (cf. Définition 3.8, page 80). La contrainte (b)

du problème linéaire (3.17) assure que la fonction $\underline{u}_k$ est inférieure à u_k entre $\overline{z}_k^{l_k}$ et $\overline{z}_k^{\bar{l}_k}$. Les contraintes (c) et (d) assurent que la fonction $\underline{u}_k$ est inférieure sur $[\underline{\xi}_k, \overline{z}_k^{l_k}]$ et $[\overline{\xi}_k, \overline{z}_k^{\bar{l}_k}]$.

Exemple 3.10. Fonction affine bornant une fonction d'utilité partielle.
Soit une fonction d'utilité linéaire par morceaux u_k définie telle que $\overline{z}_k^1 = 0$, $\overline{z}_k^2 = 0.5$, $\overline{z}_k^3 = 1$, $\overline{z}_k^4 = 1.5$, $\overline{z}_k^5 = 2$ et $\Theta_k = \{0.5, 2, 1, 0.25\}$. La fonction linéaire $\underline{u}_k(y_k) = \alpha_k \times y_k + \gamma_k$ bornant u_k, calculée grâce à l'équation (3.17), est définie par $\alpha_k = 3/2$ et $\gamma_k = -5$. Ces deux fonctions u_k et $\underline{u}_k$ sont illustrées sur la figure 3.14.

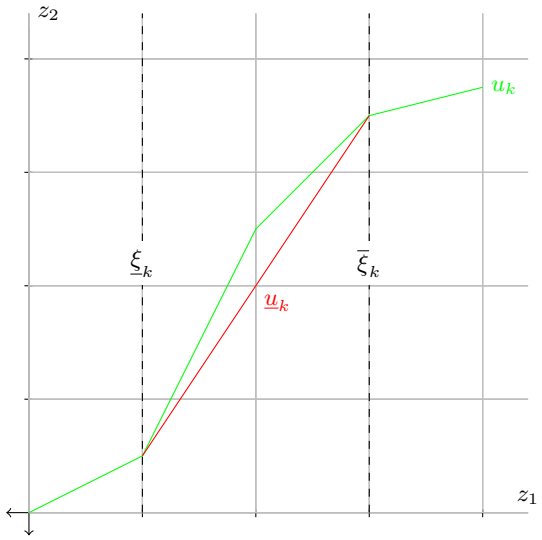

Figure 3.14 – Une fonction affine $(-)$ bornant inférieurement, sur l'intervalle $[\underline{\xi}_k, \overline{\xi}_k]$, une fonction d'utilité partielle $(-)$ linéaire par morceaux.

3.2.5 Conclusion : construction d'une fonction d'agrégation monotone bornant une fonction d'utilité basée sur l'intégrale de Choquet

Nous avons proposé différents programmes linéaires permettant de calculer une somme pondérée φ_λ, qui est une fonction linéaire et donc monotone, bornant une fonction d'utilité basée sur l'intégrale de Choquet. Cette somme pondérée est ensuite utilisée pour calculer la borne inférieure BORNE_INF^c (cf. Définition 3.8, page 80) de chaque label et ainsi de supprimer un certain nombre de ces labels dans l'algorithme d'étiquetage grâce à la règle de coupe générale (cf. Définition 3.1, page 74). L'objectif de nos travaux est que la règle de coupe permette de supprimer un maximum de label afin de diminuer le temps de calcul de l'algorithme d'étiquetage. Dans ce but, la borne inférieure BORNE_INF^c doit être la plus haute possible pour chaque label évalué. Ainsi, il est naturel de chercher à maximiser les poids $\lambda_1, \ldots, \lambda_p$ de la somme pondérée φ_λ.

Nous avons vu dans un premier temps, comment construire une somme pondérée bornant une intégrale de Choquet quelconque. Malheureusement, en général la somme des poids d'une somme pondérée bornant l'intégrale de Choquet est strictement inférieure à 1. Nous avons alors proposer d'utiliser des bornes sur les objectifs afin d'améliorer la somme pondérée bornant l'intégrale de Choquet, i.e. augmenter les poids $\lambda_1, \ldots, \lambda_p$. Les résultats numériques nous montrent une amélioration importante des temps de calcul pour des intégrales de Choquet très différentes. Ensuite nous avons proposé et étudié différentes fonctions objectif pour calculer les poids de la somme pondérée. Ainsi nous avons montré qu'il

est préférable d'utiliser une fonction objectif permettant de calculer un vecteur de poids $(\lambda_1, \ldots, \lambda_p)$ équilibré.

Pour finir nous avons proposé deux méthode pour construire une fonction d'agrégation linéaire (i.e. une somme pondérée), ou une fonction d'agrégation affine, bornant une fonction d'utilité. En effet, nous avons montré (cf. Théorème 3.2, page 76) qu'une fonction d'agrégation affine peut également être utilisée dans la borne inférieure BORNE_INF^c.

3.3 Conclusion

Dans ce chapitre nous avons présenté les différentes bornes inférieures pouvant être utilisées afin de comparer les labels avec une solution connue du problème durant l'exécution de l'algorithme d'étiquetage. Cette règle de coupe permet de supprimer des labels durant l'exécution de l'algorithme d'étiquetage pour le problème MOSP d'un sommet à un autre sommet ($R_{s,t}$). Les expérimentations confirment l'importance d'un calcul d'une borne inférieure de bonne qualité. Deux types de bornes utilisées dans la littérature ont été identifiées. Nous avons généralisé ces différentes bornes afin de pouvoir les utiliser autant que possible dans le cadre général de l'optimisation d'une fonction d'utilité dans un problème MOSP Q-$\otimes$.

Dans le cadre particulier de l'optimisation d'une fonction d'utilité basée sur l'intégrale de Choquet pour le problème MOSP Q-$\sum$, plusieurs propositions ont été faites, notamment pour l'optimisation d'une intégrale de Choquet convexe. Dans la littérature il n'existe pas de méthode pour calculer une somme pondérée bornant une intégrale de Choquet quelconque. Nous avons proposé différents problèmes linéaires pour calculer une somme pondérée bornant une intégrale de Choquet quelconque. En particulier nous proposons d'utiliser des bornes sur les objectifs afin d'améliorer significativement la somme pondérée calculée et donc les temps de calcul de l'algorithme d'étiquetage. Nous avons aussi vu que la fonction objectif utilisée dans le programme linéaire a une forte influence sur la qualité de la somme pondérée calculée.

Dans la littérature il n'existe pas de méthode pour calculer une fonction d'agrégation monotone bornant inférieurement une fonction d'utilité contenant des fonctions d'utilité linéaires par morceaux. Nous avons proposé deux méthodes pour construire des fonctions linéaires ou affines bornant les fonctions d'utilité partielles linéaires par morceaux. Ces fonctions linéaires ou affines permettent ensuite de construire une fonction d'agrégation monotone bornant la fonction d'utilité.

CHAPITRE 4

Dominance selon une fonction d'utilité basée sur l'intégrale de Choquet

Nous avons présenté dans le chapitre précédent une règle de coupe générale nous permettant de supprimer des labels dans un algorithme d'étiquetage pour optimiser une fonction d'utilité U dans le problème de plus court chemin $R_{s,t}$ d'un nœud source $s \in N$ à un nœud puits $t \in N$. Cette règle de coupe ne peut être utilisée pour optimiser une fonction d'utilité dans le cadre du problème de plus court chemin $R_{s,\bullet}$ d'un nœud source $s \in N$ à tous les nœuds du graphe. Ainsi à l'heure actuelle seule la dominance de Pareto est utilisée pour résoudre ce problème dans le cas multi-critère, même en présence de préférences. Néanmoins on sait que si la fonction d'utilité U (problème Q-$\sum$) est une somme pondérée, alors la dominance de Pareto peut être remplacée par une comparaison entre les valeurs des labels sur U puisque U est monotone (cf. Définition 2.12, page 67). Alors le problème est réduit à un problème mono-objectif. Nous avons montré précédemment (cf. §3.1.1, page 75) qu'il n'est pas possible d'utiliser une telle comparaison si la fonction d'utilité n'est pas équivalente à une somme pondérée. Dans ce cas là, la littérature ne propose que d'utiliser la dominance de Pareto pour supprimer des labels. Ainsi la fonction d'utilité n'est pas prise en compte. Dans ce chapitre nous proposons une méthode pour supprimer plus de labels que la dominance de Pareto en prenant en compte la fonction d'utilité.

Dans le cadre du problème $R_{s,\bullet}$ il n'est pas possible d'utiliser des informations en rapport avec le nœud final, puisque tous les nœuds sont des nœuds finaux. Ainsi les bornes proposées dans le chapitre précédent ne sont plus valides. On propose dans ce chapitre un nouveau type de dominance qui prend en compte une fonction d'utilité U : la U-dominance. Notons que la U-dominance n'utilise pas de fonction d'évaluation. De plus cette dominance est une définition "plus forte" de la dominance selon Pareto. Si un premier label domine selon Pareto un deuxième label alors le premier label U-domine le deuxième label.

Dans la section 4.1, on propose une définition générale de la U-dominance pour une fonction d'utilité U générale. Dans la section 4.2, on s'intéresse à la C_μ-dominance ou la fonction d'utilité est limitée à une intégrale de Choquet. La C_μ-dominance ne pouvant être vérifiée facilement on présente dans la sous-section 4.2.1 une condition suffisante pour la C_μ-dominance. Dans la section 4.3, on propose une généralisation de cette condition suffisante pour la dominance selon une fonction d'utilité composée de fonctions d'utilité partielles linéaires par morceaux et d'une intégrale de Choquet dans un problème de plus court chemin avec des objectifs additifs, bottleneck ou multiplicatifs. Dans la section 4.4, on

évalue l'impact de l'utilisation de la U-dominance (à l'aide de ces deux conditions suffisantes) dans un algorithme de label setting (cf. §2.2, page 50) par rapport à l'utilisation de la dominance de Pareto.

4.1 Dominance selon une fonction d'utilité

On considère un graphe $G = (N, A, c)$, avec N l'ensemble des nœuds, A l'ensemble des arcs et $c : A \rightarrow \mathbf{\Omega}$ la fonction de coûts. $\mathbf{\Omega} \subset \mathbb{R}^p_+$ est l'espace des objectifs. On considère p objectifs avec $P = \{1, \ldots, p\}$ l'ensemble des objectifs. On note $s \in N$ le nœud source. Soit une fonction d'utilité $U : \mathbf{\Omega} \rightarrow \xi$ croissante ou décroissante sur chaque objectif modélisant les préférences du décideur définie a priori, avec $\xi \subset \mathbb{R}^p_+$ une échelle commune de satisfaction (cf. §1.2.3, page 26). On cherche à construire pour chaque nœud $i \in N$ un chemin $r_{s,i} \in R_{s,i}$ entre s et i optimal selon U. Chaque chemin du nœud source à un autre nœud est une solution de ce problème, avec $R_{s,i}$ l'ensemble des solutions.

Dans un algorithme d'étiquetage (cf. §2.1, page 51), considérons deux labels $\ell_i, \ell'_i \in L_i$ associé à un même nœud $i \in N$. Ces deux labels correspondent chacun à un chemin du nœud source $s \in N$ au nœud i. Une dominance, notée $\prec$, permet de comparer les vecteurs de coûts de ces deux labels. Si le label ℓ'_i domine le label ℓ_i ($z(\ell'_i) \prec z(\ell_i)$) alors le label ℓ_i peut être supprimé. Ce label peut être supprimé car toute propagation (cf. §2.2, page 50) du label ℓ'_i est meilleure que la même propagation identique du label ℓ'_i. Une propagation correspond à un chemin du nœud i vers un (autre) nœud du graphe. Ainsi pour chaque chemin $r_{s,j}$ généré à partir du label ℓ'_i, il existe un meilleur chemin $r_{s,j}$ généré à partir du label ℓ_i.

Dans la littérature peu de types de dominance (ou relation de dominance) ont été proposés. Sans information préférentielle uniquement la notion d'efficacité et de dominance au sens de Pareto (cf. Définition 1.1, page 13) peuvent être utilisées pour savoir si un chemin est meilleur qu'un autre. Habituellement cette dominance, noté $\prec_P$ est utilisée. Dans le cadre du problème MOSP avec des objectifs additifs ou multiplicatifs Q-$\sum$ $-Q$-$\sum$ (cf. Chapitre 2, page 43), cette dominance permet de calculer l'ensemble maximal complet des solutions efficaces. Dans le cadre du problème MOSP avec en plus des objectifs bottleneck, Gandibleux et al [GBR06] proposent une généralisation de la dominance de Pareto (cf. Définition 2.6, page 54), noté $\prec_{GBR}$, permettant de calculer l'ensemble maximal complet des solutions efficaces.

Ces dominances ne prennent en compte aucune information sur la fonction d'utilité. Ainsi, l'ensemble maximal complet des solutions efficaces qu'elles permettent de construire, contient non seulement les solutions optimales mais aussi un grand nombre de solutions inutilement construites. Si la fonction d'utilité est monotone (cf. Théorème 3.2, page 76) alors une dominance simple, qu'on note $<_U$, peut être utilisée. Cette dominance peut être définie telle que ℓ'_i domine ℓ_i selon la fonction d'utilité U,i.e. $\ell'_i <_U \ell_i$, ssi $U(\ell'_i) < U(\ell_i)$. La fonction d'utilité monotone la plus connue est la somme pondérée. L'hypothèse de monotonie de la fonction d'utilité est une hypothèse en général considérée comme trop limitative dans le cadre de la modélisation des préférences (cf. §1.2, page 22). Carraway et al [CMM90] ont émis l'idée d'utiliser une dominance $\prec_{U,i}$ prenant en compte la fonction d'utilité U et un nœud du graphe $i \in N$. Les auteurs ne proposent pas de méthode pour vérifier une telle dominance entre deux labels. Une certaine forme de dominance prenant en compte une fonction OWA convexe a été proposée par Galand et Spanjaard [GS07], les auteurs proposent de comparer les deux labels ℓ_i et ℓ'_i par rapport à la fonction OWA convexe : $\Psi^{owa}(\ell_i) \leq \Psi^{owa}(\ell'_i) \Rightarrow \ell_i$ "*domine*" ℓ'_i. Aucun cadre général pour la dominance selon une fonction d'utilité n'est proposé dans la littérature. Aucune méthode n'est proposée pour la dominance selon une intégrale de Choquet. Pour ces raisons nous proposons de calculer une dominance $\prec_U$ prenant en compte la fonction d'utilité U.

4.1.1 Définition de la dominance selon une fonction d'utilité

Pour un label ℓ_i associé au nœud $i \in N$ et correspondant à un chemin $r_{s,i} \in R_{s,i}$ du nœud source s au nœud i, une propagation de ce chemin vers un nœud $t \in N$ est un label ℓ_t correspondant à un chemin $r_{s,i} \Diamond r_{i,t}$ du nœud source s au nœud $j$avec un chemin $r_{i,t} \in R_{i,t}$. Le vecteur de coûts du label ℓ_j est égal à $z(\ell_i) \otimes z(r_{i,j})$, avec $\otimes$ le vecteur des opérateurs binaires (cf. §2, page 43). La notion de dominance selon une fonction d'utilité * dans le cadre du problème de plus court chemin multi-objectif peut être définie de la manière suivante.

Définition 4.1. Dominance selon une fonction d'utilité pour le problème MOSP.
Soient deux labels $\ell_i, \ell'_i \in L_i$ associé avec un nœud $i \in N$ et U une fonction d'utilité. Le label ℓ'_i domine ℓ_i selon la fonction d'utilité, ℓ'_i U-domine ℓ_i, i.e. $\ell'_i \prec_U \ell_i$ s.s.i. :

$$\forall t \in N,\ \forall r_{i,t} \in R_{i,t},\ U(z(\ell'_i) \otimes z(r_{i,t})) < U(z(\ell_i) \otimes z(r_{i,t}))$$

Le label ℓ'_i domine faiblement ℓ_i selon la fonction d'utilité (i.e. $\ell'_i \precsim_U \ell_i$) s.s.i. :

$$\forall t \in N,\ \forall r_{i,t} \in R_{i,t},\ U(z(\ell'_i) \otimes z(r_{i,t})) \leq U(z(\ell_i) \otimes z(r_{i,t}))$$

Soient deux labels, associés à un nœud $i \in N$, que l'on souhaite comparer selon la U-dominance. Les propagations possibles de ces deux labels sont l'ensemble des chemins de i à tous les nœuds : i.e. $R_{i,\bullet} = \cup_{t \in N} R_{i,t}$. Notons que le chemin $\langle i \rangle \in R_{i,\bullet}$ est un chemin du nœud i au nœud i composé d'aucun arc. L'ensemble des vecteurs de coûts de ces extensions, i.e. $\cup_{t \in N} z(R_{i,t})$, sont donc un sous-ensemble assez large de $\mathbf{\Omega}$. Notons que dans le cadre du problème de plus court chemin d'un nœud à un autre nœud les extensions possibles d'un label sont limitées aux chemins vers l'unique nœud puits. Alors l'ensemble des vecteurs de coûts de ces extensions est un sous-ensemble de $\mathbf{\Omega}$ plus petit 3.2.2, nous exploitons notamment cette propriété pour calculer des bornes sur les objectifs (cf. §3.2.2, page 85). Pour cette raison dans le cadre du problème de plus court chemin d'un nœud à tous les nœuds, on considère que l'ensemble des vecteurs de coûts des extensions possibles des labels est simplement $\mathbf{\Omega}$. Alors on peut définir la U-dominance de manière plus générale qui pourrait être exploitée dans d'autres problèmes que celui de plus court chemin. Plutôt que de définir la U-dominance entre deux labels on peut la définir entre les deux vecteurs de coûts de ces labels. De cette manière la U-dominance peut être utilisée avec d'autres structures de données que l'on peut associer à des vecteurs de coûts.

Définition 4.2. Dominance selon une fonction d'utilité.
Soient $y^a, y^b \in \mathbf{\Omega}$ deux vecteurs de coûts de labels. Si pour tout vecteur de coûts $y^c \in \mathbf{\Omega}$ l'utilité de $z(y^a) \otimes z(y^c)$ est plus petite que l'utilité de $z(y^b) \otimes z(y^c)$ alors y^a U-domine y^b :

$$y^a \precsim_U y^b \iff \forall y^c \in \mathbf{\Omega},\ U(z(y^a) \otimes z(y^c)) < U(z(y^b) \otimes z(y^c))$$

Si pour tout vecteur de coûts $y^c \in \mathbf{\Omega}$ l'utilité de $z(y^a) \otimes z(y^c)$ est plus petite ou égale à l'utilité de $z(y^b) \otimes z(y^c)$ alors y^a U-domine faiblement y^b :

$$y^a \precsim_U y^b \iff \forall y^c \in \mathbf{\Omega},\ U(z(y^a) \otimes z(y^c)) \leq U(z(y^b) \otimes z(y^c))$$

Différentes propriété de la U-dominance peuvent être définies. Ainsi la relation de dominance $\precsim_U$ est transitive puisque $\forall y^a, y^b, y^c \in \mathbf{\Omega}$ si $y^a \prec_U y^b$ et $y^b \prec_U y^c$ alors par définition $y^a \prec_u y^c$. Si la fonction d'utilité est strictement croissante ou strictement décroissante sur chaque objectif, alors la relation de

∗. Ce travail a été présenté à la conférence IEEE SSCI 2011 et est publié dans les actes de la conférence [FGL11b].

dominance $\prec_U$ est une approximation de la dominance selon Pareto : i.e. $\forall y^a, y^b \in \mathbf{\Omega}$, $y^a \prec_P y^b \Rightarrow y^a \prec_U y^b$. Si la fonction d'utilité est croissante ou décroissante sur chaque objectif, alors la relation de dominance $\precsim_U$ est une approximation de la dominance selon Pareto : i.e. $\forall y^a, y^b \in \mathbf{\Omega}$, $y^a \prec_P y^b \Rightarrow y^a \precsim_U y^b$. Si la fonction d'utilité U est monotone (cf. §3.1.1, page 75) alors la relation de dominance $\prec_U$ est équivalente de la relation de dominance $<_U$, en effet pour une paire de vecteurs de coûts $y^a, y^b \in \mathbf{\Omega}$ quelconques on obtient les inéquations suivantes :

$$y^a <_U y^b \Leftrightarrow U(y^a) < U(y^b) \Leftrightarrow \forall y^c \in \mathbf{\Omega},\ U(y^a \otimes y^c) < U(y^b \otimes y^c) \Leftrightarrow y^a \prec_U y^b$$

Nous avons vu précédemment (cf. §2.4.2, page 67) qu'un algorithme d'étiquetage utilisant la dominance la $<_U$ est équivalent à un algorithme d'étiquetage pour le problème de plus court chemin mono-objectif.

Pour une fonction d'utilité croissante ou décroissante sur chaque objectif, le problème d'optimisation suivant peut être résolu pour vérifier si un vecteur de coûts $y^a \in \mathbf{\Omega}$ domine (faiblement) le vecteur de coûts y^b :

$$\begin{aligned} \delta_U(y^a, y^b) = \quad & \max \quad U(y^a \otimes y^c) - U(y^b \otimes y^c) \\ & \text{s.c.} \quad y^c \in \mathbf{\Omega} \end{aligned} \tag{4.1}$$

Si $\delta_U(y^a, y^b) \leq 0$ alors par définition $y^a \precsim_U y^b$, de même si $\delta_U(y^a, y^b) < 0$ alors par définition $y^a \prec_U y^b$. Si U est une fonction linéaire et que les objectifs sont tous additifs alors le problème (4.1) est équivalent à un programme linéaire. Si la fonction d'utilité U n'est pas linéaire alors le problème (4.1) n'est pas nécessairement linéaire et dans certains cas ce problème est même non convexe. Pour cette raison, plutôt que de résoudre exactement le problème (4.1) on propose des conditions suffisantes pour la dominance entre deux vecteurs de coûts (correspondant à deux labels) selon une fonction d'utilité basée sur l'intégrale de Choquet.

4.1.2 Utilisation de la dominance selon une fonction d'utilité

Pour les raisons exposées dans le chapitre 2, on considère l'utilisation d'un algorithme d'étiquetage pour résoudre un problème de plus court chemin d'un nœud à tous les nœuds. Pour utiliser la dominance selon une fonction d'utilité U dans l'algorithme d'étiquetage général (cf. Algorithme 2.1, page 51), il suffit d'utiliser la relation de dominance $\prec_U$ dans la règle de dominance. Soit L_i l'ensemble des labels associés au nœud $i \in N$, si durant la dernière itération de l'algorithme d'étiquetage un ou plusieurs labels ont été ajoutés à L_i alors la règle de dominance doit être vérifiée sur L_i. Ainsi l'ensemble des labels de L_i U-dominés par un autre label de L_i sont supprimés :

$$L_i := L_i \setminus \left\{ \ell_i \in L_i, \exists \ell'_i \in L_i, z(\ell_j) \prec_U z(\ell'_j) \right\}$$

Les labels ouverts supprimés de L_i doivent aussi être supprimés de l'ensemble des labels ouverts $\mathfrak{L}$.

Avec une telle règle de dominance basée sur la relation de dominance $\prec_U$, l'ensemble des chemins optimaux selon U du nœud source à tous les nœuds sont nécessairement calculés. On peut considérer qu'il est uniquement nécessaire de calculer une solution optimale selon U du nœud source à tous les nœuds. Alors la relation de dominance $\precsim_U$ peut être utilisée à la place de la relation de dominance $\prec_U$. Néanmoins dans ce cas, tout les labels dominés selon la relation de dominance $\precsim_U$ ne doivent pas être supprimés car une solution optimale selon U de ce problème peut être faiblement U-dominée. Alors pour deux labels $\ell'_i, \ell_i \in L_i$ équivalents tels que $\ell'_i \precsim_U \ell_i$ et $\ell_i \precsim_U \ell'_i$, alors un de ces deux labels doit être conservé.

4.2 Dominance selon l'intégrale de Choquet

On considère dans cette section que la fonction d'utilité U se limite à une intégrale de Choquet C_μ et que les objectifs sont tous des objectifs additifs : $\otimes = (+, \ldots, +)$. On utilise alors le terme de C_μ-dominance ($\precsim_{C_\mu}$) à la place du terme U-dominance ($\precsim_U$). Un cas plus général où la fonction d'utilité U est composée d'une intégrale de Choquet et de fonctions d'utilité partielles linéaires par morceaux est présenté dans la section 4.3 suivante. Rappelons que les coûts des chemins sont positifs ou nuls : $\mathbf{\Omega} \subset \mathbb{R}^p_+$. Le problème (4.1) est alors défini de la manière suivante :

$$\delta_{C_\mu}(y^a, y^b) = \max_{y^c \in \mathbf{\Omega}} C_\mu(y^a + y^c) - C_\mu(y^b + y^c) \tag{4.2}$$

Avec une telle fonction d'utilité U, le problème (4.2) n'est pas équivalent à un programme linéaire, sauf si la fonction de capacité de l'intégrale de Choquet est linéaire, ce qui n'est pas le cas de manière générale. De plus, la fonction objectif du problème (4.2) n'est pas croissante, comme le montre l'exemple 4.1.

Exemple 4.1.
On considère le problème MOSP *avec* 3 *objectifs additifs* ($\otimes = (+,+,+)$)*, on cherche à vérifier la dominance selon une fonction d'utilité U entre deux vecteurs de coûts y^a, y^b. Soient $y^a, y^b \in \mathbf{\Omega} = [0,100]^3$ deux vecteurs de coûts tels que $y^a = (100,0,100)$, $y^b = (0,0,50)$ et μ une fonction de capacité telle que $\mu(\{1\}) = 0$, $\mu(\{2\}) = \mu(\{3\}) = 0.5$, $\mu(\{1,2\}) = \mu(\{1,3\}) = 0.6$ et $\mu(\{2,3\}) = 0.7$. La fonction d'utilité U est l'intégrale de Choquet C_μ : $U = C_\mu$. Soit $f_{y^a,y^b} : \mathbf{\Omega} \to \xi$ une fonction vectorielle telle que $f_{y^a,y^b}(y^c) = C_\mu(y^a \otimes y^c) - C_\mu(y^b \otimes y^c)$. La fonction f_{y^a,y^b} est la fonction objectif du problème (4.2) : $\delta_{C_\mu}(y^a, y^b) = \max_{y^c \in \mathbf{\Omega}} f_{y^a,y^b}(y^c)$. On étudie la fonction f_{y^a,y^b} selon le vecteur de coûts $y^c \in \mathbf{\Omega}$. On considère que le premier coût $y^c_1 \in [0,100]$ de y^c varie entre 0 et 100 et les deux autres coûts $y^c_2 = y^c_3 = 0$ sont fixés à 0.*

- *Pour $y^c_1 \in [0,50]$ on obtient :*

$$\begin{aligned} f_{y^a,y^b}(y^c) &= [0.0 \times (100+0) + 0.4 \times (0+y^c_1) + 0.6 \times (100+0)] \\ &\quad - [0.3 \times (0+0) + 0.2 \times (0+y^c_1) + 0.5 \times (50+0)] \\ &= 0.2 \times y^c_1 + 35 \end{aligned}$$

  Alors f_{y^a,y^b} est strictement croissante sur $[0,50] \times [0] \times [0]$.
- *Pour $y^c_1 \in [50,100]$ on obtient : $f_{y^a,y^b}(y^c) = -0.1 \times w_1 + 50$, ce qui implique que f_{y^a,y^b} est strictement décroissante $[50,100] \times [0] \times [0]$.*

La fonction f_{y^a,y^b} n'est donc ni croissante ni décroissante sur le premier objectif. Cet exemple peut aussi être appliqué aux autres objectifs. Cet exemple illustre la complexité de calculer $\delta_U(y^a, y^b)$ par rapport aux problèmes (4.1) et (4.2) pour une fonction de capacité μ et des vecteurs de coûts $y^a, y^b \in \mathbf{\Omega}$.

4.2.1 Condition suffisante de dominance selon une intégrale de Choquet

Le théorème 4.1 ci-dessous, définit une condition suffisante de la dominance selon une intégrale de Choquet. Pour ce théorème le lemme suivant est démontré.

Lemme 4.1.
Pour tout vecteurs de coûts $y^a, y^b \in \mathbf{\Omega}$, et tout ensemble d'objectifs $A \in \mathfrak{P}$ l'inéquation suivante est vérifiée :

$$\min_{k \in A}(y^a_k - y^b_k) \leq \min_{k \in A}(y^a_k) - \min_{k \in A}(y^b_k) \leq \max_{k \in A}(y^a_k - y^b_k) \tag{4.3}$$

Preuve du Lemme 4.1.
Soient $t, l \in A$ *tel que* $t = \arg\min_{k\in A}(y^a_k)$ *et* $l = \arg\min_{k\in A} y^b_k$.

(min) *Comme* $-y^b_t \leq -y^b_l$ *alors* $y^a_t - y^b_t \leq y^a_t - \min_{k\in A} y^a_k$. *La partie gauche de l'équation (4.3) est prouvée :* $\min_{k\in A}(y^a_k - y^b_k) \leq (\min_{k\in A} y^a_k) - (\min_{k\in A} y^b_k)$.

(max) *Comme* $y^a_t \leq y^a_l$ *alors* $(\min_{k\in A} y^a_k) - y^b_l \leq y^a_l - y^b_l$. *La partie droite de l'équation (4.3) est prouvée :* $(\min_{k\in A} y^a_k) - (\min_{k\in A} y^b_k) \leq \max_{k\in A} y^a_k - y^b_k$.

□

Soient $y^a, y^b \in \mathbf{\Omega}$ deux vecteurs de coûts, on chercher à vérifier si y^a C_μ-domine y^b : $\delta_{C_\mu}(y^a, y^b) < 0$. Pour cela on définit une approximation supérieure $\overline{\delta}_{C_\mu}$ de δ_{C_μ} peut être utilisée :

$$\forall y^a, y^b \in \mathbf{\Omega},\ \overline{\delta}_{C_\mu}(y^a, y^b) \geq \delta_{C_\mu}(y^a, y^b) = \max_{y^c\in\mathbf{\Omega}} C_\mu(y^a + y^c) - C_\mu(y^b + y^c)$$

Ainsi pour deux labels ℓ^a_i, ℓ^b_i associés au nœud $i \in N$, si $\overline{\delta}_{C_\mu}(\ell^a, \ell^b) < 0$ alors ℓ^a_i U-domine ℓ^b_i et le label ℓ^b_i peut alors être supprimé. Le théorème suivant utilise une approximation supérieure $\overline{\delta}_{C_\mu}$ de δ_{C_μ} pour définir une condition suffisante de la dominance selon une intégrale de Choquet.

Théorème 4.1. Condition suffisante de la dominance selon l'intégrale de Choquet.
Soient deux vecteurs de coûts $y^a, y^b \in \mathbf{\Omega}$, *et une intégrale de Choquet* C_μ. y^a *domine* y^b *selon l'intégrale de Choquet* ($y^a \prec_{C_\mu} y^b$) *si :*

$$\overline{\delta}_{C_\mu}(y^a, y^b) = \sum_{A\in\mathfrak{P}^+} m(A) \times (\max_{k\in A} y^a_k - y^b_k) + \sum_{A\in\mathfrak{P}^-} m(A) \times (\min_{k\in A} y^a_k - y^b_k) < 0$$

Avec $m : \mathfrak{P}(P) \to \mathbb{R}$ *la représentation de Möbius (cf. Définition 1.19, page 35) de la fonction de capacité* μ.

Preuve du Théorème 4.1.
Si $\overline{\delta}_{C_\mu}$ *est une approximation inférieure de* δ_{C_μ} *telle que définie par l'équation (4.2) alors le théorème 4.1 est valide. On montre dans cette preuve que* $\overline{\delta}_{C_\mu}$ *telle que définie dans le théorème 4.1 est bien une approximation inférieure de* δ_{C_μ}. *Pour calculer* $\overline{\delta}_{C_\mu}$ *on utilise la représentation de Möbius* $m : \mathfrak{P}(P) \to \mathbb{R}$ *de l'intégrale de Choquet (cf. Définition 1.19, page 35). On note que la transformation de Möbius, contrairement à la fonction de capacité, peut être négative pour certains ensembles d'objectifs. En utilisant cette représentation l'équation (4.2) peut être définie de la manière suivante :*

$$\delta_{C_\mu}(y^a, y^b) = \max_{y^c\in\mathbf{\Omega}} \sum_{A\in\mathfrak{P}} m(A) \times \left[\min_{k\in A}(y^a_k + y^c_k) - \min_{k\in A}(y^b_k + y^c_k)\right] \tag{4.4}$$

avec $\mathfrak{P}$ *l'ensemble des parties de l'ensemble des objectifs* P. *Puisque l'on connait* y^a, y^b, *la variable de ce problème est* $y^c \in \mathbf{\Omega}$, *pour pouvoir résoudre cette équation on cherche à y retirer cette variable. Dans un premier temps on définit l'ensemble des ensembles d'objectifs* $\mathfrak{P}^+ = \{A \subset P : 0 \leq m(A)\}$ *qui ont un Möbius positif ou nul, et l'ensemble des ensembles d'objectifs* $\mathfrak{P}^- = \{B \subset P : m(B) < 0\}$ *dont le Möbius est négatif. L'équation (4.4) peut alors être présentée de la manière suivante :*

$$\delta_{C_\mu}(y^a, y^b) = \max_{y^c\in\mathbf{\Omega}} \left\{ \sum_{A\in\mathfrak{P}^+} m(A) \times \left[\min_{k\in A}(y^a_k + y^c_k) - \min_{k\in A}(y^b_k + y^c_k)\right] + \sum_{A\in\mathfrak{P}^-} m(A) \times \left[\min_{k\in A}(y^a_k + y^c_k) - \min_{k\in A}(y^b_k + y^c_k)\right] \right\}$$

On peut alors définir une approximation inférieure $\overline{\delta}_{C_\mu}$ de δ_{C_μ} en décomposant l'équation précédente en 2^p composantes, chaque composante correspondant à un sous-ensemble d'objectifs. Pour un ensemble d'objectifs $A \in \mathfrak{P}$ la composante associée est $[\min_{k\in A}(y^a_k + y^c_k) - \min_{k\in A}(y^b_k + y^c_k)]$. Les composantes associées à un ensemble d'objectifs dont le Möbius est positif ou nul ($\mathfrak{P}^+$), doivent être maximisées par rapport à $y^c \in \mathbf{\Omega}$. Les composantes associées à un ensemble d'objectifs dont le Möbius est négatif ($\mathfrak{P}^-$) doivent être minimisées par rapport à $y^c \in \mathbf{\Omega}$. Alors l'approximation supérieure $\overline{\delta}_{C_\mu}$ de δ_{C_μ} peut être définie de la manière suivante :

$$\begin{aligned}\delta_{C_\mu}(y^a, y^b) \leq \overline{\delta}_{C_\mu}(y^a, y^b) = &\sum_{A\in\mathfrak{P}^+} m(A) \times \max_{y^c\in\mathbf{\Omega}} \left[\min_{k\in A}(y^a_k + y^c_k) - \min_{k\in A}(y^b_k + y^c_k)\right] \\ &+ \sum_{A\in\mathfrak{P}^-} m(A) \times \min_{y^c\in\mathbf{\Omega}} \left[\min_{k\in A}(y^a_k + y^c_k) - \min_{k\in A}(y^b_k + y^c_k)\right]\end{aligned} \tag{4.5}$$

Considérons un ensemble d'objectifs $A \subseteq \mathfrak{P}^+$ dont le Möbius est positif ou nul. En utilisant le lemme 4.1 on obtient l'inégalité suivante :

$$\forall y^c \in \mathbf{\Omega},\ \min_{k\in A}(y^a_k + y^c_k) - \min_{k\in A}(y^b_k + y^c_k) \leq \max_{k\in A}(y^a_k + y^c_k - y^b_k - y^c_k) = \max_{k\in P}(y^a_k - y^b_k) \tag{4.6}$$

Soit un objectif $t \in A$ tel que $t = \arg\max_{k\in A}(y^a_k - y^b_k)$. On définit alors un vecteur de coûts $y^c \in \mathbf{\Omega}$ tel que $\forall k \in A \setminus \{t\}$, $y^c_k \sim +\infty$ et $y^c_t = 0$. On peut en déduire que $\min_{k\in A}(y^a_k + y^c_k) - \min_{k\in A}(y^b_k + y^c_k) = (y^a_t + y^c_t) - (y^b_t + y^c_t) = y^a_t - y^b_t$. Ainsi le vecteur y^c peut être retiré de la composante associée avec l'ensemble d'objectifs A, mais aussi avec n'importe quelle composante associée à un ensemble d'objectifs dont le Möbius est positif :

$$\max_{y^c\in\mathbf{\Omega}} \left[\min_{k\in A}(y^a_k + y^c_k) - \min_{k\in A}\left(y^b_k + y^c_k\right)\right] = \max_{k\in A}\left(y^a_k - y^b_k\right) \tag{4.7}$$

De manière identique, avec l'aide du lemme 4.1 le vecteur y^c peut être retiré de chaque composante associée avec un Möbius négatif. Ainsi pour un ensemble d'objectifs $B \in \mathfrak{P}^-$ dont le Möbius est négatif on obtient :

$$\min_{y^c\in\mathbf{\Omega}} \left[\min_{k\in B}(y^a_k + y^c_k) - \min_{k\in B}\left(y^b_k + y^c_k\right)\right] = \min_{k\in B}\left(y^a_k - y^b_k\right) \tag{4.8}$$

En utilisant les composantes définies par (4.7) et (4.8) dans l'équation (4.5) on obtient :

$$\overline{\delta}_{C_\mu}(y^a, y^b) = \sum_{A\in\mathfrak{P}^+} m(A) \times \max_{k\in A}\left(y^a_k - y^b_k\right) + \sum_{A\in\mathfrak{P}^-} m(A) \times \min_{k\in A}\left(y^a_k - y^b_k\right)$$

Le théorème 4.1 est démontré.

□

L'exemple suivant illustre l'utilisation de la condition suffisante de la C_μ-dominance (cf. Théorème 4.1, page ci-contre) dans un algorithme d'étiquetage pour le problème de l'exemple 3.1.

Exemple 4.2. Dominance selon une intégrale de Choquet, suite de l'exemple 3.1.
Soient trois labels ℓ^a_2, ℓ^b_2, ℓ^c_2 tels que $z(\ell^a_2) = (4, 5)$, $z(\ell^b_2) = (0, 9)$, $z(\ell^c_2) = (11, 4)$. On cherche à optimiser l'intégrale de Choquet C_μ définie par la fonction de capacité μ : $\mu(\emptyset) = 0$, $\mu(\{1\}) = 0.7$, $\mu(\{2\}) = 0.7$ $\mu(\{1, 2\}) = 1$. La représentation de Möbius $m : N \to [-1, 1]$ de la fonction de capacité est : $m(\emptyset) = 0$, $m(\{1\}) = 0.7$, $m(\{2\}) = 0.7$ et $m(\{1, 2\}) = -0.4$. On compare le label ℓ^a aux labels

ℓ^b et ℓ^c selon la C_μ-dominance. Sur les figures 4.1 et 4.2, la zone bleu désigne l'espace des points non C_μ-dominés par le label ℓ^a : $\{y \in \mathbf{\Omega} : 0 < \overline{\delta}_{C_\mu}(z(\ell^a), y)\}$. La condition suffisante de C_μ-dominance (cf. Théorème 4.1, page 104) entre le label ℓ_2^a et le label ℓ_2^b 4.1 n'est pas vérifiée car $\overline{\delta}_{C_\mu}(z(\ell^a), z(\ell^b)) \geq 0$:

$$\overline{\delta}_{C_\mu}(z(\ell^a), z(\ell^b)) = 0.7 \times (4-0) + 0.7 \times (5-9) - 0.4 \times (5-9) = 2.8 - 2.8 + 1.6 = 1.6$$

Par contre, grâce cette condition suffisante on peut montrer que le label ℓ_2^a C_μ-domine le label ℓ_2^c 4.2, en effet $\overline{\delta}_{C_\mu}(z(\ell^a), z(\ell^c)) < 0$:

$$\overline{\delta}_{C_\mu}(z(\ell^a), z(\ell^c)) = 0.7 \times (4-11) + 0.7 \times (5-4) - 0.4 \times (4-11) = -4.9 + 0.7 - 2.8 = -1.4$$

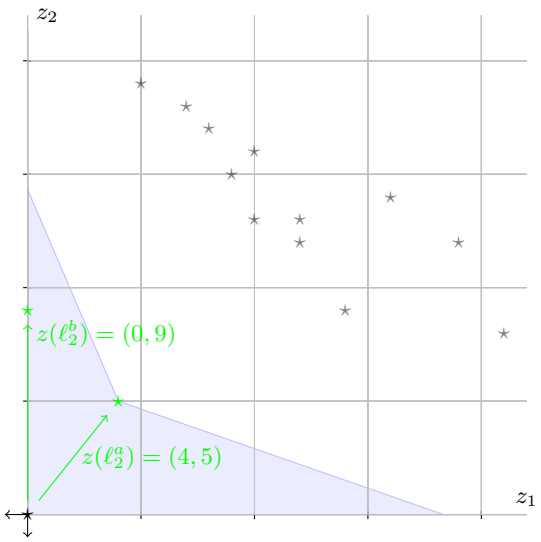

Figure 4.1 – Le vecteur vert $(0, 9)$ correspond au label ℓ_2^b, le vecteur vert $(4, 5)$ correspond au label ℓ_2^a.

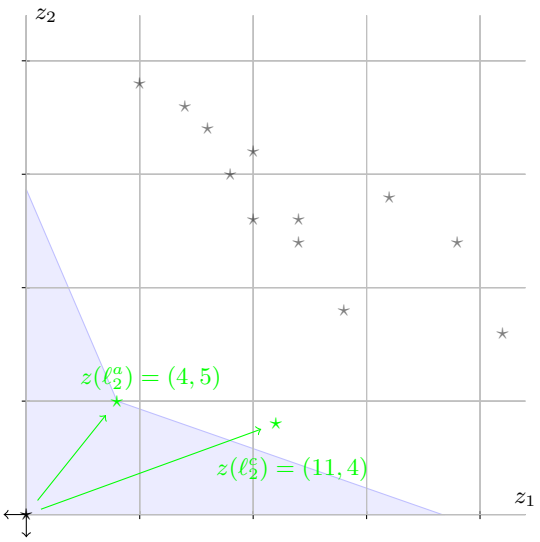

Figure 4.2 – Le vecteur vert $(11, 4)$ correspond au label ℓ_2^c, le vecteur vert $(4, 5)$ correspond au label ℓ_2^a.

4.2.2 Conclusion

Nous avons présenté dans cette section une condition suffisante de la dominance selon une intégrale de Choquet (C_μ-dominance). Cette condition suffisante peut être vérifiée rapidement si le nombre d'objectifs n'est pas trop élevé. Elle permet de supprimer plus de labels dans un algorithme d'étiquetage que la simple utilisation de la Dominance de Pareto comme le montre l'exemple 4.2. Les temps de calcul d'un algorithme d'étiquetage utilisant cette condition suffisante de la C_μ-dominance pour supprimer des labels sont présentés dans la section 4.4.

4.3 Relation de dominance avec différents types d'objectifs et des fonctions d'utilité linéaires par morceaux

Considérons maintenant une fonction d'utilité $U = \Psi_\mu^u$ composée d'une intégrale de Choquet C_μ et de fonctions d'utilité partielles linéaires par morceaux $u_1, \ldots, u_p$ (cf. Définition 1.12, page 28).

$$\forall y \in \mathbf{\Omega},\ \Psi_\mu^u(y) = C_\mu(u_1(y_1), \ldots, u_p(y_p))$$

Dans cette section on présente une condition suffisante de la dominance selon cette fonction d'utilité (Ψ_μ^u-dominance). Pour chaque critère $k \in P$ la fonction d'utilité u_k est croissante si la fonction objectif z_k doit être maximisée ou décroissante si la fonction objectif z_k doit être minimisée. On considère dans cette section des objectifs de type somme, bottleneck ou multiplicatif et des coûts entiers positifs $\mathbf{\Omega} = \mathbb{R}_+^p$.

Pour deux vecteurs de coûts $y^a, y^b \in \mathbf{\Omega}$, le premier vecteur de coûts y^a Ψ_μ^u-domine le deuxième vecteur de coûts y^b si $\delta_{\Psi_\mu^u} < 0$, avec $\delta_{\Psi_\mu^u}$ défini tel que :

$$\forall y^a, y^b \in \mathbf{\Omega},\ \delta_{\Psi_\mu^u}(y^a, y^b) = \max_{y^c \in \mathbf{\Omega}} \Psi_\mu^u(y^a + y^c) - \Psi_\mu^u(y^b + y^c)$$

De la même manière que pour la dominance selon une intégrale de Choquet, le théorème suivant utilise une approximation supérieure $\overline{\delta}_{\Psi_\mu^u}$ de $\delta_{\Psi_\mu^u}$ pour définir une condition suffisante de la Ψ_μ^u-dominance ainsi .

Théorème 4.2. Condition suffisante de la dominance selon une fonction d'utilité composée de fonctions d'utilité partielle et d'une intégrale de Choquet
Soient deux vecteurs de coûts $y^a, y^b \in \mathbf{\Omega}$ et une fonction d'utilité Ψ_μ^u avec $u_1, \ldots, u_p$ des fonctions d'utilité partielles linéaires par morceaux et μ une fonction de capacité. y^a domine y^b selon la fonction d'utilité Ψ_μ^u ($y^a \prec_{\Psi_u^\mu} y^b$) si :

$$\overline{\delta}_{\Psi_\mu^u}(y^a, y^b) = \sum_{A \in \mathfrak{P}^+} m(A).\max_{\theta \in \Theta_k} \theta_k.g_k^{\max}(y_k^a, y_k^b) + \sum_{A \in \mathfrak{P}^-} m(A).\min_{\theta \in \Theta_k} \theta_k.g_k^{\min}(y_k^a, y_k^b) < 0 \qquad (4.9)$$

Avec Θ_k l'ensemble des coefficients d'inclinaison (cf. Définition 1.12, page 28) la fonction d'utilité partielle linéaire par morceaux u_k associée avec l'objectif $k \in P$. Pour chaque objectif $k \in P$, la valeur de $g_k^{\max}(y_k^a, y_k^b)$ et $g_k^{\min}(y_k^a, y_k^b)$ dépend de la fonction objectif. Si la fonction objectif z_k est additive ($\otimes_k = +$) et l'objectif k est à minimiser alors g_k est défini telle que :

$$g_k^{\max}(y_k^a, y_k^b) = g_k^{\min}(y_k^a, y_k^b) = y_k^a - y_k^b$$

Si la fonction objectif z_k est une fonction minimale ($\otimes_k = \min$) et l'objectif k est à maximiser alors g_k est défini telle que :

$$g_k^{\max}(y_k^a, y_k^b) = \max\left\{y_k^a - y_k^b, 0\right\}$$
$$g_k^{\min}(y_k^a, y_k^b) = \min\left\{y_k^a - y_k^b, 0\right\}$$

Si la fonction objectif z_k est une fonction maximale ($\otimes_k = \max$) et l'objectif k est à minimiser alors g_k est défini telle que :

$$g_k^{\max}(y_k^a, y_k^b) = \min\left\{y_k^a - y_k^b, 0\right\}$$
$$g_k^{\min}(y_k^a, y_k^b) = \max\left\{y_k^a - y_k^b, 0\right\}$$

Si la fonction objectif z_k est multiplicative ($\otimes_k = \times$) et l'objectif k est à minimiser alors g_k est défini telle que :

$$g_k^{\max}(y_k^a, y_k^b) = \max\left\{\underline{\xi}_k.(y_k^a - y_k^b), \overline{\xi}_k.(y_k^a - y_k^b)\right\}$$
$$g_k^{\min}(y_k^a, y_k^b) = \min\left\{\underline{\xi}_k.(y_k^a - y_k^b), \overline{\xi}_k.(y_k^a - \overline{\xi}_k.y_k^b)\right\}$$

Avec $\underline{\xi}_k = \min \Omega_k$ une borne inférieure et $\overline{\xi}_k = \max \Omega_k$ une borne supérieure de l'objectif $k \in P$.

Preuve du Théorème 4.2.
En utilisant la représentation de Möbius de l'intégrale de Choquet alors la fonction $\delta_{\Psi_\mu^u}$ peut être définie telle que :

$$\begin{aligned}\delta_{\Psi_\mu^u}(y^a, y^b) = &\sum_{A\in\mathfrak{P}^+} m(A).\max_{y^c\in\mathbf{\Omega}}\left[\min_{k\in A} u_k(y_k^a \otimes_k y_k^c) - \min_{k\in A} u_k(y_k^b \otimes_k y_k^c)\right] \\ &+ \sum_{A\in\mathfrak{P}^-} m(A).\min_{y^c\in\mathbf{\Omega}}\left[\min_{k\in A} u_k(y_k^a \otimes_k y_k^c) - \min_{k\in A} u_k(y_k^b \otimes_k y_k^c)\right]\end{aligned} \tag{4.10}$$

On démontre que $\overline{\delta}_{\Psi_\mu^u}$, telle que définie dans le théorème 4.2, est une approximation supérieure de $\delta_{\Psi_\mu^u}$. De même que dans la preuve 4.1 précédente on peut diviser l'équation précédente en 2^p composantes. Pour un ensemble d'objectifs $A \in \mathfrak{P}$, la composante associée avec cet ensemble est $[\min_{k\in A} u_k(y_k^a \otimes_k y_k^c) - \min_{k\in A} u_k(y_k^b \otimes_k y_k^c)]$. Grâce au lemme 4.1 on obtient l'équation :

$$\begin{aligned}\min_{k\in A}\left[u_k(y_k^a \otimes_k y_k^c) - u_k(y_k^b \otimes_k y_k^c)\right] &\\ \leq \min_{k\in A} u_k(y_k^a \otimes_k y_k^c) - \min_{k\in A} u_k(y_k^b \otimes_k y_k^c) &\\ \leq \max_{k\in A}\left[u_k(y_k^a \otimes_k y_k^c) - u_k(y_k^b \otimes_k y_k^c)\right] &\end{aligned}$$

Considérons que le Möbius de l'ensemble d'objectifs A est positif ou nul : $0 \geq m(A)$. Soit $l \in A$ un objectif de A tel que $l = \arg\max_{k\in A}\left[u_k(y_k^a \otimes_k y_k^c) - u_k(y_k^b \otimes_k y_k^c)\right]$. Alors on définit un vecteur de coûts $y^c \in \mathbf{\Omega}$ tel que $y_l^c = \mathbf{0}_k$ et $\forall k \in P \setminus \{l\}$, $y_k^c \sim +\infty$. Pour ce vecteur de coûts l'égalité suivante est vérifiée : $\min_{k\in A} u_k(y_k^a \otimes_k y_k^c) - \min_{k\in A} u_k(y_k^b \otimes_k y_k^c) = u_l(y_l^a \otimes_l y_l^c) - u_l(y_l^b \otimes_l y_l^c)$, La composante associée avec A, ou avec tout ensemble d'objectifs dont le Möbius est positif ou nul, peut être définie de la manière suivante :

$$\begin{aligned}\max_{y^c\in\mathbf{\Omega}}\left[\min_{k\in A} u_k(y_k^a \otimes_k y_k^c) - \min_{k\in A} u_k(y_k^b \otimes_k y_k^c)\right] &\\ = \max_{k\in A}\max_{y_k^c\in\Omega_k}\left[u_k(y_k^a \otimes_k y_k^c) - u_k(y_k^b \otimes_k y_k^c)\right] &\end{aligned} \tag{4.11}$$

De manière identique, on peut définir la composante associée avec un ensemble d'objectifs $B \in \mathfrak{P}^+$ dont le Möbius est négatif :

$$\begin{aligned}\min_{y^c\in\mathbf{\Omega}}\left[(\min_{k\in A} u_k(y_k^a \otimes_k y_k^c)) - (\min_{k\in A} u_k(y_k^b \otimes_k y_k^c))\right] &\\ = \min_{k\in A}\min_{y_k^c\in\Omega_k}\left[u_k(y_k^a \otimes_k y_k^c) - u_k(y_k^b \otimes_k y_k^c)\right] &\end{aligned} \tag{4.12}$$

On s'intéresse à résoudre les équations $\max_{y_k^c\in\Omega_k}[u_k(y_k^a \otimes_k y_k^c) - u_k(y_k^b \otimes_k y_k^c)]$ et $\min_{y_k^c\in\Omega_k}[u_k(y_k^a \otimes_k y_k^c) - u_k(y_k^b \otimes_k y_k^c)]$. Comme les fonctions d'utilité partielles $u_1, \ldots, u_p$ sont des fonctions linéaires par

morceaux, alors la différence maximale entre les deux vecteurs $y^a \otimes y^c$ et $y^b \otimes y^c$ selon u_k peut être bornée en utilisant la dérivée maximale $\max_{\theta_k \in \Theta_k} \theta_k$ et la dérivée minimale $\min_{\theta_k \in \Theta_k} \theta_k$ de la fonction d'utilité partielle u_k. Alors pour tout objectif $k \in P$ la différence entre $u_k(y_k^a \otimes_k y_k^c)$ et $u_k(y_k^b \otimes_k y_k^c)$ est bornée par :

$$\begin{aligned}\min_{\theta_k \in \Theta_k} \theta_k . \left[(y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right] \\ \leq u_k(y_k^a \otimes_k y_k^c) - u_k(y_k^b \otimes_k y_k^c) \\ \leq \max_{\theta_k \in \Theta_k} \theta_k . \left[(y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right]\end{aligned}$$

Par conséquent pour chaque objectif $k \in P$, en utilisant les inéquations suivantes on peut retirer la fonction d'utilité u_k des équations (4.11) et (4.12) :

$$\begin{aligned}\max_{y_k^c \in \mathbf{\Omega}} u_k(y_k^a \otimes_k y_k^c) - u_k(y_k^b \otimes_k y_k^c) \leq \max_{y^c \in \mathbf{\Omega}} \max_{\theta \in \Theta_k} \theta_k . \left[(y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right] \\ \min_{y^c \in \mathbf{\Omega}} \min_{\theta_k \in \Theta_k} \theta_k . \left[(y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right] \leq \min_{y_k^c \in \mathbf{\Omega}} u_k(y_k^a \otimes_k y_k^c) - u_k(y_k^b \otimes_k y_k^c)\end{aligned}$$

Selon la définition de la fonction d'utilité Ψ_μ^u, si la fonction objectif z_k est à minimiser alors les coefficients d'inclinaison $\{\theta_k^1, \ldots, \theta_k^{\tau_k}\}$ sont tous positifs :

$$\begin{aligned}\max_{y_k^c \in \Omega_k} \max_{\theta \in \Theta_k} \theta_k . \left[(y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right] = \max_{\theta_k \in \Theta_k} \theta_k . \left[\max_{y_k^c \in \Omega_k} (y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right] \\ \min_{y_k^c \in \Omega_k} \min_{\theta \in \Theta_k} \theta_k . \left[(y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right] = \min_{\theta_k \in \Theta_k} \theta_k . \left[\min_{y_k^c \in \Omega_k} (y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right]\end{aligned} \tag{4.13}$$

Selon la définition de la fonction d'utilité Ψ_μ^u, si la fonction objectif z_k est à maximiser alors les coefficients d'inclinaison $\{\theta_k^1, \ldots, \theta_k^{\tau_k}\}$ sont tous négatifs :

$$\begin{aligned}\max_{y_k^c \in \Omega_k} \max_{\theta \in \Theta_k} \theta_k . ((y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)) = \max_{\theta_k \in \Theta_k} \theta_k . \left[\min_{y_k^c \in \Omega_k} (y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right] \\ \min_{y_k^c \in \Omega_k} \min_{\theta \in \Theta_k} \theta_k . \left[(y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right] = \min_{\theta_k \in \Theta_k} \theta_k . \left[\max_{y_k^c \in \Omega_k} (y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c)\right]\end{aligned} \tag{4.14}$$

Considérons des bornes inférieures $\underline{\xi}_1, \ldots, \underline{\xi}_p$ et supérieures $\overline{\xi}_1, \ldots, \overline{\xi}_p$ sur les objectifs, telles que que $\forall k \in P$, $\underline{\xi}_k = \min \Omega_k$ et $\overline{\xi}_k = \max \Omega_k$. Ainsi tout vecteurs de coûts $y^c \in \mathbf{\Omega}$ est défini entre ces bornes, en effet $[\underline{\xi}_1, \overline{\xi}_p] \times \ldots [\underline{\xi}_p, \overline{\xi}_p] = \mathbf{\Omega}$. Alors selon la définition des opérateurs binaires $\otimes_1, \ldots, \otimes_p$, pour chaque objectif $k \in P$ une des deux inéquations suivantes est vérifiée :

$$\begin{aligned}(y_k^a \otimes_k \underline{\xi}_k) - (y_k^b \otimes_k \underline{\xi}_k) \leq (y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c) \leq (y_k^a \otimes_k \overline{\xi}_k) - (y_k^b \otimes_k \overline{\xi}_k) \\ \textit{ou} \\ (y_k^a \otimes_k \overline{\xi}_k) - (y_k^b \otimes_k \overline{\xi}_k) \leq (y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c) \leq (y_k^a \otimes_k \underline{\xi}_k) - (y_k^b \otimes_k \underline{\xi}_k)\end{aligned}$$

On peut alors retirer le vecteur de coûts y^c des équations (4.13) et (4.14) en utilisant les inéquations précédentes :

$$\begin{aligned}\max_{y_k^c \in \Omega_k} (y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c) = \max \left\{(y_k^a \otimes_k \underline{\xi}_k) - (y_k^b \otimes_k \underline{\xi}_k), (y_k^a \otimes_k \overline{\xi}_k) - (y_k^b \otimes_k \overline{\xi}_k)\right\} \\ \min_{y_k^c \in \Omega_k} (y_k^a \otimes_k y_k^c) - (y_k^b \otimes_k y_k^c) = \min \left\{(y_k^a \otimes_k \underline{\xi}_k) - (y_k^b \otimes_k \underline{\xi}_k), (y_k^a \otimes_k \overline{\xi}_k) - (y_k^b \otimes_k \overline{\xi}_k)\right\}\end{aligned}$$

L'équation précédente peut être clarifiée selon le type d'objectif $(+, \times, \min, \max)$ *considéré. Si* $\otimes_k = +$ *alors la fonction objectif* z_k *est à minimiser et on obtient :*

$$\max_{\theta\in\Theta_k} \theta_k . \max\left\{(y_k^a \otimes_k \underline{\xi}_k) - (y_k^b \otimes_k \underline{\xi}_k), (y_k^a \otimes_k \overline{\xi}_k) - (y_k^b \otimes_k \overline{\xi}_k)\right\} = \max_{\theta\in\Theta_k} \theta_k.(y_k^a - y_k^b)$$

Si $\otimes_k = \min$ *alors la fonction objectif* z_k *est à maximiser et on obtient :*

$$\max_{\theta\in\Theta_k} \theta_k . \max\left\{(y_k^a \otimes_k \underline{\xi}_k) - (y_k^b \otimes_k \underline{\xi}_k); (y_k^a \otimes_k \overline{\xi}_k) - (y_k^b \otimes_k \overline{\xi}_k)\right\} = \max_{\theta\in\Theta_k} \theta_k . \min\left\{y_k^a - y_k^b, 0\right\}$$

Si $\otimes_k = \max$ *alors la fonction objectif* z_k *est à minimiser et on obtient :*

$$\max_{\theta\in\Theta_k} \theta_k . \max\left\{(y_k^a \otimes_k \underline{\xi}_k) - (y_k^b \otimes_k \underline{\xi}_k); (y_k^a \otimes_k \overline{\xi}_k) - (y_k^b \otimes_k \overline{\xi}_k)\right\} = \max_{\theta\in\Theta_k} \theta_k . \max\left\{y_k^a - y_k^b, 0\right\}$$

Si $\otimes_k = \times$ *alors la fonction objectif* z_k *est à minimiser et considérons que* $\underline{\xi}_p = \mathbf{0}_p = 1$*, on obtient :*

$$\begin{aligned}\max_{\theta\in\Theta_k} \theta_k . \max&\left\{(y_k^a \otimes_k \underline{\xi}_k) - (y_k^b \otimes_k \underline{\xi}_k); (y_k^a \otimes_k \overline{\xi}_k) - (y_k^b \otimes_k \overline{\xi}_k)\right\} \\ &= \max_{\theta\in\Theta_k} \theta_k . \max\left\{\underline{\xi}.(y_k^a - y_k^b), \overline{\xi}_k.(y_k^a - y_k^b)\right\}\end{aligned}$$

Le théorème 4.2 est prouvé.

□

4.3.1 Conclusion

Nous avons présenté dans cette section comment calculer une approximation supérieure $\overline{\delta}_{\Psi_\mu^u}$ de $\delta_{\Psi_\mu^u}$ permettant d'établir une condition suffisante pour la dominance selon une fonction $U = \Psi_\mu^u$ composée d'une intégrale de Choquet C_μ et de fonctions d'utilité partielles $u_1, \ldots, u_p$. Ainsi cette méthode permet calculer un chemin optimal selon une fonction d'utilité Ψ_μ^u entre un nœud source et tous les nœuds du graphe, tout en supportant des objectifs de type additif, multiplicatif ou bottleneck. Aucune autre méthode pour ce problème n'est disponible dans la littérature. Ce problème représente de nombreuses difficultés par rapport à un problème de plus court chemin mono-objectif qui sont : les interactions entre les objectifs, les fonctions d'utilité partielles linéaires par morceaux et les objectifs non additifs. Aucune autre méthode pour résoudre ce problème n'est disponible dans la littérature. Les temps de calcul, pour quelques instances de ce problème, d'un algorithme d'étiquetage utilisant cette condition suffisante pour supprimer des labels sont présentés dans la section suivante.

Une perspective existe concernant l'intégration de fonctions d'utilité partielles linéaires par morceaux dans le calcul de $\overline{\delta}_{\Psi_\mu^u}$. L'intégration des fonctions d'utilité dans l'équation 4.9 est effectuée en utilisant les dérivées $\theta_k^1, \ldots, \theta_k^{\tau_k}$ de chaque fonction d'utilité u_k, $k \in P$. Ces dérivées ont été définies comme les coefficients d'inclinaison (cf. Définition 1.12, page 28). On utilise ensuite dans cette équation la plus petite et la plus grande des dérivées. Il serait possible de considérer uniquement les dérivées de la fonction u_k uniquement entre $\min\{y_k^a, y_k^b\}$ et $\overline{\xi}_k$. En effet toute propagation de y^a et y^b est définie entre $\min\{y_k^a, y_k^b\}$ et $\overline{\xi}_k$ sur l'objectif $k \in P$.

4.4 Évaluation numérique

Dans le but d'évaluer l'utilisation de la dominance selon une fonction d'utilité on considère des graphes construits par le générateur *ggen*. Les graphes générés ont des coûts positifs sur les arcs définis

entre 1 et 10. Le nombre de nœuds des graphes est : $n = 1000$ ou 3000 ou 6000 ou 10000. Le nombre d'arcs des graphes est défini selon le nombre de nœuds du graphe tel que : $m = 5 \times n$ ou $50 \times n$ ou $500 \times n$. Nous définissons la densité du graphe comme le nombre moyen d'arcs pour chaque nœud, i.e. 5 ou 50 ou 500. Pour chaque combinaison de taille et de densité, un ensemble de 50 graphes a été généré et est utilisé pour l'expérimentation. Nous ne testons pas les combinaisons de 10000 nœuds avec $m = 50 \times n$ ou $m = 500 \times n$ à cause de son coût prohibitif en espace mémoire. Le nœud source est choisi aléatoirement.

La dominance selon une fonction d'utilité est utilisée dans un algorithme d'étiquetage (cf. Algorithme 2.1, page 51). Cet algorithme d'étiquetage utilise la fonction lexicographique comme fonction de sélection.

4.4.1 Spécificité de l'algorithme d'étiquetage

Afin d'utiliser de manière efficace la U-dominance dans un algorithme d'étiquetage, il faut éviter de vérifier cette U-dominance pour chaque paire de labels ℓ_i^a, ℓ_i^b associés à un même nœud. En effet l'évaluation des conditions suffisantes pour la U-dominance des théorèmes 4.1 et 4.2 est plus complexe que l'évaluation de la dominance de Pareto. Afin de limiter le nombre de tests de vérification des conditions suffisantes pour la U-dominance, pour une paire de labels ℓ_i^a et ℓ_i^b, on vérifie dans un premier temps la dominance de Pareto entre ℓ_i^a et ℓ_i^b qui est aussi une condition suffisante pour la U-dominance (cf. §4.1.1, page 101). Ensuite, si la dominance de Pareto n'est pas vérifiée alors on évalue la condition suffisante du théorème 4.1 ou 4.2 uniquement si :

$$\beta \leq \sum_{k \in P} z_k(\ell_i^a) - z_k(\ell_i^b)$$

Avec β le seuil de différence pour vérifier la U-dominance, par défaut on utilise $\beta = 10$. La valeur de β peut aussi être modifiée en fonction de la taille du graphe de la fonction d'utilité ou des valeurs des coûts sur les arcs. Quelque soit le seuil de différence utilisé l'algorithme d'étiquetage, l'algorithme d'étiquetage reste exacte puisque aucun label non U-dominé n'est supprimé.

4.4.2 Dominance selon une intégrale de Choquet avec des objectifs additifs

On cherche ici à évaluer l'efficacité de la condition suffisante de la dominance selon une intégrale de Choquet C_μ définie par le théorème 4.1. Cette condition suffisante est utilisée pour évaluer la C_μ-dominance avec une intégrale de Choquet C_μ dans l'algorithme d'étiquetage général (cf. Algorithme 2.1, page 51). La fonction de sélection du label courant est la fonction lexicographique (cf. Définition 1.4, page 15). Cet algorithme d'étiquetage permet de résoudre exactement l'optimisation d'une intégrale de Choquet dans un problème de plus court chemin multi-objectif d'un nœud à tous les nœuds. De manière à évaluer l'algorithme pour différents profils de décideurs, plusieurs intégrales de Choquet sont définies par les fonctions de capacité μ^a, μ^b, μ^c, μ^d, μ^e, μ^f, μ^g de la table 4.1. Les fonctions de capacité μ^a et μ^c sont des fonctions (quasi)-linéaires (peu ou aucune interactions) (cf. §1.2.5, page 32), les intégrales de Choquet C_{μ^a} et C_{μ^c} sont équivalentes à des sommes pondérées. Les fonctions de capacité μ^f et μ^g sont des fonctions modélisent des interactions très forte entre les objectifs. Les fonctions de capacité μ^b, μ^d, μ^e et μ^h sont des fonctions intermédiaires. Les temps de calcul pour la résolution exacte du problème MOSP sont donnés par la table 4.2.

Les tables 4.2 et 4.3 montre que l'utilisation de la C_μ-dominance permet de diminuer les temps de calcul de manière significative pour de nombreuses intégrales de Choquet par rapport au même algorithme utilisant uniquement la dominance de Pareto, i.e. l'algorithme de Martins [Mar84a]. Ainsi avec

fonctions de capacité	ensembles d'objectifs							
	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	P
μ^a	0.0	0.333	0.333	0.333	0.666	0.666	0.666	1.0
μ^b	0.0	0.3	0.3	0.3	0.7	0.7	0.7	1.0
μ^c	0.0	0.6	0.2	0.2	0.8	0.8	0.4	1.0
μ^d	0.0	0.3	0.3	0.3	0.8	0.4	0.6	1.0
μ^e	0.0	0.4	0.3	0.3	0.4	0.4	0.3	1.0
μ^f	0.0	0.4	0.1	0.4	1.0	0.8	1.0	1.0
μ^g	0.0	0.6	0.1	0.3	0.9	1.0	0.4	1.0
μ^h	0.0	0.3	0.3	0.3	0.8	0.4	0.9	1.0

Table 4.1 – Fonctions de capacité pour l'intégrale de Choquet avec 3 objectifs.

une intégrale de Choquet linéaire (avec une fonction de capacité additive) C_{μ^a}, C_{μ^c} l'algorithme d'étiquetage utilisant la condition suffisante obtient des temps de calcul 90% plus rapide en moyenne que le même algorithme avec la dominance de Pareto. Quand l'intégrale de Choquet est proche d'une fonction linéaire, i.e. avec peu d'interactions, telles C_{μ^b}, C_{μ^d}, la C_μ-dominance permet aussi de faire baisser les temps de calcul de manière importante : une réduction de $90\% - 80\%$ est obtenue par rapport à l'utilisation seule de la dominance de Pareto. Pour une intégrale de Choquet avec des interactions significatives, telle C_{μ^g} et C_{μ^h}, la C_μ-dominance permet de diminuer légèrement les temps de calcul : $60\% - 70\%$ de réduction par rapport à l'utilisation seule de la dominance de Pareto .

Par contre, quand l'intégrale de Choquet modélise des interactions fortes, négatives ou positives, ce qui est le cas pour C_{μ^e} et C_{μ^f} alors les temps de calcul obtenus en utilisant la C_μ-dominance souffrent d'une perte de performance par rapport à l'utilisation uniquement de la dominance de Pareto.

On note aussi que en général, plus la taille des graphes et/ou plus la densité des graphes augmente plus la C_μ-dominance permet de faire baisser les temps de calcul. Par exemple pour un graphe avec 6000 nœud et 300000 arcs, la déviation de temps de calcul par rapport à l'algorithme de Martins est de 2.3% pour l'intégrale de Choquet C_{μ^a}, de 9.37% pour l'intégrale de Choquet C_{μ^b} et de 26.0% pour l'intégrale de Choquet C_{μ^h} alors que leurs déviations moyennes sont 7.720%, 20.395% et 41.575% respectivement.

4.4.2.1 Analyse pour les cas fortement concave ou convexe de l'intégrale de Choquet

Dans le cas de l'intégrale de Choquet C_{μ^e} cette fonction modélise de fortes interactions positives alors on dit qu'elle est fortement convexe. Pour C_{μ^f} cette fonction modélise de fortes interactions négatives, on dit également qu'elle est fortement concave. On peut montrer que analytiquement que dans ces deux cas il est difficile de supprimer des labels. Ce problème n'est pas dû à la condition nécessaire du théorème 4.1, mais à la définition générale de la dominance selon une fonction d'utilité (cf. Définition 4.2, page 101).

Si la fonction d'utilité est totalement concave $U = \max$ alors il n'est pas possible de supprimer plus de labels avec la U-dominance qu'avec la dominance de Pareto : soit $y^a, y^b \in \mathbf{\Omega}$ deux vecteurs de coûts correspondant à deux labels. Considérons que y^a ne domine y^b selon Pareto et que $y^a \neq y^b$, on peut alors montrer que y^a ne U-domine pas y^b. Il existe un objectif $l \in P$ tel que $y^b_l < y^a_l$. On peut donc construire un vecteur de coûts $y^c \in \mathbf{\Omega}$ tel que $y^c_l = \epsilon$ et $\forall k \in P \setminus \{l\}$, $y^c_k = 0$, avec $\epsilon \in \mathbb{R}_+$ un nombre suffisamment grand. Alors $C_\mu(y^a + y^c) = \max_{k \in P} y^a_k + y^c_k = y^a_l + \epsilon$ et $C_\mu(y^b + y^c) = \max_{k \in P} y^b_k + y^c_k = y^b_l + \epsilon$.

intégrales de Choquet	temps de calcul (s)						déviation (%)	
	$m = 5 \times n$			$m = 50 \times n$		$m = 500 \times n$		
	1000	6000	10000	1000	6000	1000	max.	moy.
C_{μ^a}	0.004	0.038	0.073	0.062	0.509	0.796	12.820	7.720
C_{μ^b}	0.010	0.100	0.197	0.196	2.037	1.780	32.051	20.395
C_{μ^c}	0.006	0.052	0.100	0.085	0.734	0.955	19.230	10.592
C_{μ^d}	0.006	0.057	0.111	0.091	0.829	0.970	19.230	11.145
C_{μ^e}	0.035	0.461	1.086	1.411	24.798	8.553	135.896	113.002
C_{μ^f}	0.036	0.468	1.106	1.529	23.091	8.847	137.261	116.911
C_{μ^g}	0.015	0.152	0.301	0.399	4.336	3.493	48.076	34.915
C_{μ^h}	0.017	0.175	0.347	0.496	5.652	4.263	57.233	41.575
Martins	0.031	0.462	0.957	1.318	21.729	7.448	100	100

Table 4.2 – Expérimentations sur un problème de plus courts chemins d'un nœud à tous les nœuds ($R_{s,\bullet}$) avec 3 objectifs additifs. Comparaison des temps de calcul entre un algorithme d'étiquetage 2.1 utilisant la C_μ-dominance et l'algorithme de Martins qui utilise la dominance de Pareto. Les fonctions de capacité μ^a,...,μ^h sont définies dans la table 4.1. La déviation (exprimée en pourcentage) est la moyenne des temps de calcul ou le temps de calcul maximum par rapport au temps de calcul de l'algorithme de Martins.

intégrales de Choquet	temps de calcul (s)		déviation (%)	
	20×20	40×10	max.	moy.
C_{μ^a}	0.002	0.002	0.19	0.25
C_{μ^b}	0.332	0.199	31.46	32.30
C_{μ^c}	0.018	0.014	1.79	2.00
C_{μ^d}	0.042	0.027	4.19	4.23
C_{μ^e}	1.289	0.978	138.82	156.88
C_{μ^f}	1.729	1.109	171.75	177.38
C_{μ^g}	0.383	0.216	34.15	36.18
C_{μ^f}	1.034	0.627	99.14	101.16
Martins	1.002	0.632	100	100

Table 4.3 – Expérimentations sur un problème de plus courts chemins d'un nœud à tous les nœuds ($R_{s,\bullet}$) avec 3 objectifs additifs. Comparaison des temps de calcul entre un algorithme d'étiquetage 2.1 utilisant la C_μ-dominance et l'algorithme de Martins qui utilise la dominance de Pareto. Les fonctions de capacité μ^a,...,μ^h sont définies dans la table 4.1. La déviation (exprimée en pourcentage) est la moyenne des temps de calcul ou le temps de calcul maximum par rapport au temps de calcul de l'algorithme de Martins. Les instances sont des graphes ayant une topologie en grille, 20×20 indique une grille de 20 nœuds de largeur et de 20 nœuds de longueur.

Ainsi la U-dominance selon une fonction d'utilité totalement convexe ne nous permet pas de supprimer le label correspondant au vecteur de coûts y^b puisque par définition $\delta_U(y^a, y^b) = y^a_l - y^b_l > 0$. Cette remarque est illustrée sur la figure 4.3.

Si la fonction d'utilité est totalement concave $U = \min$ alors il n'est pas non plus possible de supprimer plus de labels avec la U-dominance qu'avec la dominance de Pareto : Soient $y^a, y^b \in \mathbf{\Omega}$ deux vecteurs de coûts correspondant à deux labels. Considérons que y^a ne domine y^b selon Pareto et que $y^a \neq y^b$, on peut alors montrer que y^a ne U-domine par y^b. Il existe un objectif $l \in P$ tel que $y^b_l < y^a_l$. On peut donc construire un vecteur de coûts $y^c \in \mathbf{\Omega}$ tel que $y^c_l = 0$ et $\forall k \in P \setminus \{l\}$, $y^c_k = \epsilon$, avec $\epsilon \in \mathbb{R}_+$ un nombre suffisamment grand. Alors $C_\mu(y^a + y^c) = \min_{k \in P} y^a_k + y^c_k = y^a_l$ et $C_\mu(y^b + y^c) = \min_{k \in P} y^b_k + y^c_k = y^b_l$. Ainsi la U-dominance selon la fonction d'utilité totalement concave ne nous permet pas de supprimer le label correspondant au vecteur de coûts y^b puisque par définition $\delta_U(y^a, y^b) = y^a_l - y^b_l > 0$. Cette remarque est illustrée sur la figure 4.4. Pour les cas extrêmes

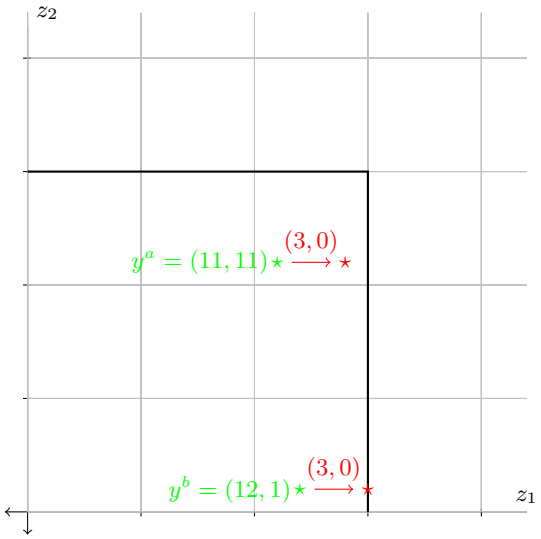

Figure 4.3 – Le trait en gras représente la courbe de niveau de la fonction d'agrégation max. Les deux points verts représentent des vecteurs de coûts y^a, y^b, Les deux vecteurs rouges représentent le vecteur de coûts $y^c = (3, 0)$.

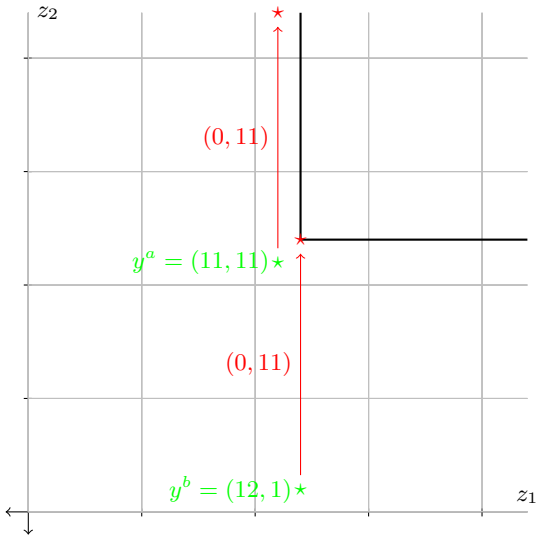

Figure 4.4 – Le trait en gras représente la courbe de niveau de la fonction d'agrégation min. Les deux points verts représentent des vecteurs de coûts y^a, y^b, Les deux vecteurs rouges représentent le vecteur de coûts $y^c = (0, 11)$.

de l'intégrale de Choquet (C_{μ^e} et C_{μ^f}) la C_μ-dominance ne permet de supprimer que des labels déjà Pareto dominé. La condition suffisante pour la C_μ-dominance (cf. Théorème 4.1, page 104) étant plus coûteuse à vérifiée que la dominance de Pareto une perte de performance est constatée dans ces cas de l'intégrale de Choquet sur les expérimentations numériques.

4.4.3 Dominance selon une fonction d'utilité avec des fonctions d'utilité partielles

On cherche dans ce paragraphe à évaluer l'efficacité de la dominance selon une fonction d'utilité Ψ^u_μ comprenant une intégrale de Choquet C_μ et des fonctions d'utilité partielles linéaires par morceaux $u_1, \ldots, u_p$. Dans ce but on utilise (cf. Définition 4.4.1, page 111) la condition suffisante de la Ψ^u_μ-

fonction utilité	temps de calcul (s)						déviation (%)
	$m = 5 \times n$			$m = 50 \times n$		$m = 500 \times n$	
	1000	6000	10000	1000	6000	1000	moyenne
$\Psi_{\mu^a}^{u^a}$	0.11	3.63	10.6	0.28	6.0	1.39	12.90
$\Psi_{\mu^a}^{u^b}$	0.31	17.89	60.58	2.25	68.8	9.74	72.30
$\Psi_{\mu^a}^{u^c}$	0.42	21.58	78.6	2.34	76.8	10.12	84.97
$\Psi_{\mu^a}^{u^d}$	0.62	23.75	80.4	2.28	120.8	11.58	101.44
Martins	0.52	23.52	77.5	2.93	115.4	11.28	0

Table 4.4 – Expérimentations sur un problème de plus courts chemins d'un nœud à tous les nœuds ($R_{s,\bullet}$) avec 3 objectifs additifs. Comparaison des temps de calcul entre un algorithme d'étiquetage 2.1 utilisant la Ψ_μ^u-dominance et l'algorithme de Martins qui utilise la dominance de Pareto. Les fonctions de capacité μ^a,...,μ^h sont définies dans la table 4.1. Le vecteur de fonctions d'utilité partielles u est un vecteur de fonctions d'identité. La déviation (exprimée en pourcentage) est la moyenne des temps de calcul ou le temps de calcul maximum par rapport au temps de calcul de l'algorithme de Martins.

dominance définie par le théorème 4.2 dans l'algorithme d'étiquetage général (cf. Algorithme 2.1, page 51). La fonction de sélection du label courant est la fonction lexicographique. Afin d'évaluer l'impact de l'utilisation de fonctions d'utilité partielles linéaires par morceaux on considère une seule intégrale de Choquet (C_{μ^a} définie par la table 4.1). Les vecteurs de fonctions d'utilité partielles u^a, u^b, u^c et u^d sont définis de telle sorte que : $u_1^a, \ldots, u_p^a$ sont toutes des fonctions linéaires, $u_1^b, \ldots, u_p^b$ sont des fonctions linéaires par morceaux convexes, $u_1^c, \ldots, u_p^c$ sont des fonctions linéaires par morceaux concaves, $u_1^d, \ldots, u_p^d$ sont des fonctions linéaires par morceaux pour une part concaves pour l'autre convexes. Les temps de calcul de cet algorithme d'étiquetage sont donnés par la table 4.4.

La table 4.4 montre que la méthode proposée pour supprimer des labels permet de diminuer de manière significative les temps de calcul pour l'algorithme d'étiquetage, avec des fonctions concaves ou convexes. Comme dans la sous-section 4.4.2 précédente, pour certaines fonctions d'utilité notre méthode provoque de légères pertes de performance. Ainsi avec les fonctions d'utilité partielles $u_1^d, \ldots, u_p^d$ un temps de calcul supplémentaire par rapport à l'algorithme de Martins.

4.5 Conclusion

Nous avons proposé dans ce chapitre une dominance généralisée pour l'optimisation d'une fonction d'utilité dans un problème de plus court chemin d'un nœud à tous les nœuds. Dans le chapitre 2 nous avons vu que si la fonction d'utilité vérifie la monotonie (cf. Définition 2.12, page 67) alors cette dominance consiste à comparer les labels sur la base de leur valeur sur la fonction U. Dans le cas contraire la dominance de Pareto était utilisée pour comparer les labels. La dominance de Pareto qui est largement utilisée en optimisation multi-objectif ne prend pas en compte les préférences du décideur. Ce chapitre montre qu'il est possible de construire une relation de dominance prenant en compte les préférences du décideur, même si la fonction d'utilité n'est pas monotone.

Nous avons également mis en évidence que si cette fonction d'utilité est basée sur une fonction d'agrégation $\max$ ou $\min$ alors seule la dominance de Pareto peut être utilisée pour comparer deux labels

associés à un même nœud. On rappelle qu'en minimisation, une fonction d'agrégation $\min$ modélise une interaction négative totale entre les objectifs, alors qu'une fonction d'agrégation $\max$ modélise une interaction positive totale et que la somme pondérée modélise une absence totale d'interaction entre les objectifs. Avec l'intégrale de Choquet qui permet de modéliser différents niveaux *intermédiaires* d'interaction entre la somme pondérée et les fonctions d'agrégation $\max$ et $\min$, nous montrons qu'une dominance intermédiaire, entre la comparaison des labels sur U et la dominance de Pareto, peut être utilisée.

Pour une intégrale de Choquet quelconque C_μ nous proposons la C_μ-dominance qui est une généralisation de la dominance de Pareto. La C_μ-dominance étant difficile à calculer on propose une condition suffisante de la C_μ-dominance. Cette condition suffisante utilisée (cf. §4.4.1, page 111) dans un algorithme d'étiquetage permet de supprimer plus de labels que l'utilisation de la dominance de Pareto dans un algorithme d'étiquetage et ainsi de réduire de manière significative les temps de calcul pour de nombreuses intégrales de Choquet. Les résultats numériques confirment que, moins l'intégrale de Choquet modélise des interactions positives ou négatives plus la condition suffisante de l'utilisation de la C_μ-dominance permet de réduire les temps de calcul par rapport à la dominance de Pareto. La condition suffisante de la C_μ-dominance est plus difficile à évaluer que la dominance de Pareto. Quand la C_μ-dominance ne supprime pas plus de labels que la dominance de Pareto alors l'algorithme d'étiquetage utilisant cette condition suffisante souffre de temps de calcul plus long. La principale perspective de ces travaux est de proposer une méthode permettant de différencier les intégrales de Choquet pour lesquelles il est intéressant d'utiliser la C_μ-dominance et celles pour lesquelles la C_μ-dominance n'est pas intéressante. Dans ce deuxième cas uniquement la dominance de Pareto doit être utilisée. Nous pensons que le degré de *orness* d'une intégrale de Choquet [Mar98] permettrait de différencier ces deux cas.

Dans un deuxième temps nous avons travaillé sur l'optimisation d'une fonction d'utilité Ψ_μ^u composée d'une intégrale de Choquet et de fonctions d'utilité partielles dans un problème de plus court chemin multi-objectif avec objectifs additifs, bottleneck ou multiplicatifs. Pour cela nous proposons la Ψ_μ^u-dominance qui est une généralisation de la C_μ-dominance. Les expérimentations numériques montrent aussi que la Ψ_μ^u-dominance peut aussi être utilisée de manière efficace par rapport à la dominance de Pareto pour certaines fonctions d'utilité.

À notre connaissance nos travaux sur la U-dominance sont les seuls travaux à prendre en compte une fonction d'utilité non-monotone dans la dominance utilisée dans un algorithme d'étiquetage. Les méthodes proposées sont aussi les seules existantes pour optimiser une fonction d'utilité dans un problème de plus court chemin d'un nœud à tous les nœuds ou un problème de plus court chemin avec des objectifs non additifs.

chapterProposition d'algorithmes de routage pour les réseaux internet

La section 4.6 présente le cadre général du routage dans les réseaux internet et plus précisément du routage prenant en compte la Qualité de Service (routage QOS). De nombreuses méthodologies existent dans la littérature pour le routage QOS [XN99]. Le routage par contraintes est l'une d'entres elles. Cette méthodologie permet de prendre en compte plusieurs objectifs de qualité de service (ou de coût) pour le calcul des routes, des contraintes étant définies sur ces objectifs. Ainsi le routage par contraintes est un type de routage multi-objectif. Différents algorithmes de routage par contraintes de la littérature sont présentés dans la sous-section 4.6.3. Dans la section 4.7, nous proposons de définir, dans le cadre du routage multi-objectif, une fonction d'utilité agrégeant les objectifs plutôt que de définir des contraintes sur les objectifs. On présente ensuite l'application, pour le routage multi-objectif, de nos travaux du chapitre précédent concernant l'optimisation d'une fonction d'utilité dans des algorithmes d'étiquetage.

4.6 Routage dans les réseaux internet et qualité de service

Un réseau internet (ou réseau IP) est un réseau de télécommunication basé sur le *protocole internet*. Un tel réseau est composé de routeurs et de liens entre ces routeurs. Il peut être modélisé par un graphe où les liens sont modélisés par les arcs du graphe et les routeurs par les nœuds du graphe. Les attributs de qualité de service ou de coût des liens du réseau sont modélisées par des valeurs de performance ou de coût sur les arcs. Ces attributs correspondent par exemple à la bande passante d'un lien, au délai de transmission d'un lien, ...

Le service produit par un réseau IP est le transport de paquets de données entre les différents routeurs du réseau. Un paquet arrive sur le réseau depuis un seul et unique routeur source et doit sortir du réseau par un ou plusieurs routeurs puits. Si le paquet doit sortir vers un unique routeur puits alors on parle de routage "*unicast*". Si le paquet doit sortir du réseau par plusieurs routeurs puits alors on parle de routage "*multicast*". Un cas particulier du routage "*multicast*" est le routage "*broadcast*" quand le paquet doit sortir du réseau par l'ensemble des routeurs du réseaux. On considère dans ce chapitre le routage *unicast*.

4.6.1 Le routage dans les réseaux internet

Différents types de protocoles et de techniques sont utilisés simultanément à différents niveaux (physique, liaison, réseau,...) pour transporter les paquets dans les réseaux internet. On s'intéresse dans le cadre de cette thèse aux protocoles et techniques du niveau réseau et plus précisément au routage. Le routage désigne les protocoles et techniques permettant aux routeurs de choisir le "meilleur" chemin pour router chaque paquet transporté dans le réseau. *Le réseau internet* est composé d'un ensemble de systèmes autonomes (AS). Un AS est un sous-réseau du réseau internet où les routeurs partagent des protocoles de routage communs et un administrateur unique à l'ensemble des routeurs de l'AS. Un AS est composé de routeurs intérieurs et de routeurs de bordure, les routeurs de bordure étant connectés aux autres AS. Les protocoles de routage qui permettent de router les paquets à l'intérieur d'un AS sont dits IGP (*Interior Gateway Protocol*) et les protocoles de routage permettant de router les paquets entre les routeurs de bordure d'AS adjacents sont dits BGP (*Border Gateway Protocol*). Dans le cadre de cette thèse on s'intéresse aux protocoles de routage IGP. On note qu'un protocole de routage IGP s'intéresse uniquement au routage à l'intérieur de AS où il est déployé, quand bien même le routeur d'origine et le routeur de destination du paquet sur le réseau internet sont tous deux situés en dehors de l'AS.

Le routage des paquets dans le réseau est basé sur un *protocole de routage* et un *algorithme de routage*. Le protocole de routage définit comment les routeurs du réseau sont informés de la topologie et des attributs des liens du réseau. Les attributs des liens du réseau permettent de définir des métriques.

Par exemple, un attribut de bande passante sur chaque lien du réseau permet de définir une métrique pour les chemins du réseau. Ces métriques peuvent être considérées comme des objectifs [YF03], en effet, naturellement l'administrateur du réseau, pour chaque métrique, va chercher soit à la maximiser soit à la minimiser. Par exemple on va chercher à minimiser une métrique de coût† et maximiser une métrique de capacité. Si plusieurs métriques/objectifs sont pris en compte on peut parler de routage multi-objectif. Quand une métrique de qualité de service est prise en compte, on parle de routage QoS, si uniquement une métrique de coût est prise en compte on parle de routage "*best-effort*". Les algorithmes de routage pour le routage QoS ou pour le routage *best-effort* sont des algorithmes de plus court chemin mono-objectif ou multi-objectif si plusieurs métriques sont considérées simultanément. Ces algorithmes s'appuient sur les informations produites par le protocole de routage. Dans le cas du routage *best-effort*, la métrique de coût est en général équivalente à un objectif additif (1-$\sum$) à minimiser. L'algorithme de routage *best-effort* est alors un algorithme de plus court chemin mono-objectif avec un objectif additif, tel l'algorithme de Dijkstra [Dij59].

Dans les protocoles de routage dit "*link-state*" [YF03], tel OSPF (*Open Shortest Path First*) et IS-IS (*Intermediate System To Intermediate System*), chaque routeur informe l'ensemble des autres routeurs du réseau de l'existence et des attributs des liens qui lui sont adjacents. Ces informations permettent à chaque routeur de connaître la topologie et les attributs des liens du réseau. Dans les protocoles de routage dit "*distance-vector*" [YF03] tels que RIP (*Routing Information Protocol*) et EIGRP (*Enhanced Interior Gateway Routing Protocol*) aucun algorithme de routage à proprement parler n'est utilisé. Néanmoins, le fonctionnement d'un tel protocole de routage est similaire à l'algorithme de plus court chemin $R_{\bullet,\bullet}$ de Roy-Floyd-Warshall [Roy59] discuté dans le chapitre 2. En effet chaque routeur informe régulièrement ses routeurs voisins des attributs (associés aux métriques) de l'ensemble des chemins qu'il connait vers les autres routeurs du réseau. Grâce aux informations obtenues par les routeurs de bordure à l'aide des protocoles de routage BGP, les routeurs de l'AS sont informés de l'existence de routeurs à l'extérieur du réseau et des attributs des chemins pour atteindre ces routeurs. Notons que les protocoles *link-state* OSPF et IS-IS sont les protocoles de routage les plus couramment utilisés dans les réseaux IP [FT02]. Dans la suite de ce chapitre on considère qu'un protocole de réseau *link-state*, prenant en compte plusieurs métriques, permet à chaque routeur de connaître la topologie exacte du réseau ainsi que les attributs des liens. Ainsi on s'intéresse dans ce chapitre uniquement aux algorithmes de routage. Le routage d'un paquet dans un réseau peut être fait de manière centralisé ou distribué. Le *routage centralisé* (ou par la source) [Beu06, YF03] implique que le routeur source du paquet calcule le chemin jusqu'à sa sortie du réseau, ensuite le chemin du paquet est encodé dans l'entête du paquet [YF03] puis chaque routeur le long du chemin *fait suivre* (*forward*) le paquet en fonction de ce chemin. Ce type de routage peut par exemple être utilisé pour construire un "*label-switching path*" dans le cadre de l'utilisation du "*MultiProtocol Label Switching*" (MPLS).

Le routage dans un réseau peut être de type *proactif* ou de type *réactif* (aussi appelé *on-demand*) [YF03]. Si le routage est proactif alors les chemins entre toutes les paires de routeurs sont construis qu'ils soient utilisés ou non pour router des paquets. Ainsi, pour un routeur, le routage proactif correspond à la construction de l'ensemble des chemins de lui même à l'ensemble des routeurs du réseau, ce qui est équivalent au problème $R_{s,\bullet}$ (cf. §2.1.1.1, page 46). Si le routage est réactif alors les chemins empruntés par les paquets sont calculés uniquement pour un flux de paquets précis entre deux routeurs et pour une durée de temps précise, ces chemins sont donc calculés uniquement quand le réseau en a besoin. Pour un routeur et un flux de paquets, le routage réactif correspond à la construction d'un chemin de lui même vers le routeur destination du flux de paquets, ce qui est équivalent au problème $R_{s,t}$ (cf. §2.1.1.1,

†. Dans cette thèse on considère uniquement le coût administratif.

page 46). Dans les réseaux IP [YF03], le routage proactif est plus couramment utilisé que le routage réactif.

Le routage d'un paquet dans un réseau peut être fait de manière centralisé ou distribué, voir [YF03]. Généralement dans le réseaux IP le routage est distribué ou "*hop-by-hop routing*" [NM99]. Comme c'est le cas du routage *best-effort* généralement utilisé avec les protocoles OSPF et IS-IS. Dans le cadre du *routage distribué*, le calcul du plus court chemin n'est pas réalisé uniquement par le routeur source mais aussi l'ensemble des routeurs composant le chemin emprunté par le paquet. Chaque routeur calcule un plus court chemin vers le routeur destination du paquet. Ensuite chaque routeur utilise ce plus court chemin pour fait suivre le paquet ‡. Le routage distribué est en général utilisé dans le cadre du routage proactif ($R_{s,\bullet}$) [WC96].

4.6.2 Prise en compte de la Qualité de Service

Initialement les réseaux IP appliquaient uniquement le routage *best-effort* pour l'ensemble des paquets transitant dans le réseau [XN99]. Désormais avec la commercialisation de services dans les réseaux IP, il est nécessaire de prendre en compte également la Qualité de Service (QoS) [XN99]. De nombreux protocoles et techniques ont été proposés dans la littérature pour prendre en compte la qualité de service [XN99].

Afin d'améliorer la qualité de service pour l'ensemble des paquets transitant sur un réseau IP, les techniques de "*traffic engineering*" permettent d'équilibrer la charge globale du réseau sur les différents liens du réseau [XN99, FRT02]. La charge d'un lien pour une période de temps donnée correspond à la taille totale des paquets transitant sur ce lien (le *trafic*) durant cette période divisé par la capacité (non réservée) du lien pour cette période. Un déséquilibre de la charge globale du réseau implique que certains liens seront congestionnés (ou surchargés) alors que d'autre liens sont au repos. La congestion d'un lien implique un délai de transmission important pour ce lien ainsi qu'une perte possible de certains paquets qui transitent sur ce lien. Une meilleure répartition de la charge globale permet ainsi, non seulement d'améliorer la qualité de service pour les paquets transitant sur les liens congestionnés, mais aussi de réduire les coûts d'exploitation de l'infrastructure du réseau [XN99]. Parmi les méthodes de *traffic engineering* liées au calcul des plus courts chemins on note la méthode proposée par Fortz et al [FRT02, FT02] qui s'applique dans le cadre du routage *best-effort* avec les protocoles OSPF ou IS-IS. Les auteurs proposent de modifier les coûts d'un ensemble de liens parmi les liens les plus chargés du réseau. Ainsi les chemins entre paires de routeurs, une fois réactualisés, éviteront de passer par ces liens et donc la charge de ces liens diminuera. Cette méthode permet une amélioration significative de la répartition de la charge du réseau [FT02]. Un autre exemple est l'option "*equal-cost multipath*" (ECMP §) [XN99] du protocole OSPF, qui permet aux routeurs du réseau de faire suivre les paquets équitablement entre l'ensemble des chemins dont le coût est égal. Cette option nécessite donc le calcul, pour chaque paire de routeurs, de l'ensemble des chemins de coût minimal.

Exemple 4.3. *Traffic engineering.*
Soit un AS *défini par un ensemble de 6 routeurs représentés par les nœuds :* $a, \ldots, f$ *(cf. Figure 4.5). Les routeurs correspondant aux nœuds* a, d *et* f *sont des routeurs de bordure. On suppose que les liens du réseau sont déchargés à l'exception d'un unique flux de paquets du routeur* a *au routeur* d*. Le protocole de routage prend en compte uniquement la métrique de coût définie par un attribut de valeur* 1 *sur*

‡. Si la destination est située en dehors de l'AS, alors le paquet est envoyé vers le routeur de bordure de l'AS le plus proche de la destination.

§. Notons que l'utilisation de l'option ECMP peut provoquer une rupture de l'ordre de réception des paquets dans les flux TCP.

chaque lien. Supposons dans un premier temps que la stratégie de routage best-effort soit appliquée à

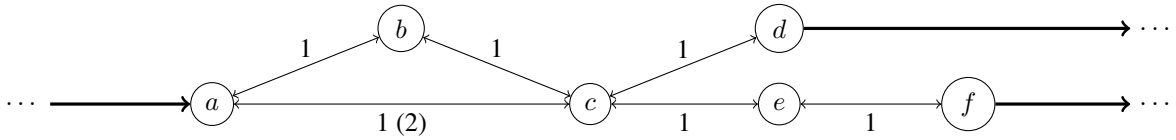

Figure 4.5 – Un réseau internet autonome défini par un ensemble de routeurs (nœuds) $a, b, \ldots, f$, avec a, d et f les routeurs de bordure. Seul l'attribut de coût est défini sur les liens (arcs).

l'ensemble des paquets transitant sur le réseau. Afin de router le flux de paquets du routeur a au routeur d un chemin entre a et d est calculé à l'aide d'un algorithme de plus court chemin mono-objectif en prenant en compte uniquement la métrique de coût : $\langle a, c, d\rangle$. Ainsi la totalité du trafic entre les nœuds a et c passe par le lien (a, c) tandis que les liens (a, b) et (b, c) sont au repos : la charge globale n'est pas répartie équitablement sur le réseau.

En utilisant les méthodes de traffic engineering présentées précédemment il est possible de mieux répartir la charge globale. Par exemple la méthode de Fortz et al [FRT02] permet d'augmenter le coût de l'arc (a, c) tel que $c(a, c) = 2$ car cet arc est surchargé. Considérons qu'une stratégie (par exemple l'option ECMP*) est utilisée par le routeur a pour repartir les paquets équitablement entre les deux chemins $\langle a, b, c\rangle$ et $\langle a, b\rangle$ entre a et c qui ont un coût égal. Alors un algorithme de plus court chemin mono-objectif est appliqué afin de router les paquets du flux de paquets entre le routeur a et le routeur d et seule la métrique de coût est prise en compte (avec la modification $c(a, c) = 2$ par rapport à la figure 4.5). Selon cette métrique, deux chemins de a à d sont optimaux : $\langle a, b, c, d\rangle$ et $\langle a, c, d\rangle$, avec un coût de 3 pour ces deux chemins, $z(\langle a, b, c, d\rangle) = z(\langle a, c, d\rangle) = 3$. Grâce à la stratégie* ECMP *le flux entre a et d est reparti équitablement entre les deux chemins optimaux de a et d. Ainsi la moitié du trafic entre a et c passe par le chemin $\langle a, c\rangle$ et l'autre moitié du trafic passe par le chemin $\langle a, b, c\rangle$: le trafic est alors réparti équitablement entre les liens (a, b), (b, c) et (a, c).*

Ainsi les techniques de *traffic engineering* consistent à une prise en compte implicite des besoins de QoS, puisque ces techniques permettent une amélioration globale de la QoS sans prendre en compte des besoins particuliers. Les autres techniques de QoS présentées ci-dessous permettent de prendre en compte de manière explicite des besoins de QoS.

Des modèles tels que le *service différentié* (DIFFSERV) [XN99] permettent de définir plusieurs classes de qualité de service (classes QoS). Chaque paquet est affecté à avec une classe QoS. La classe QoS associée à un paquet (ou un flux de paquets) est définie par le "*service level agreement*" (SLA) qui lie l'administrateur du réseau avec le client qui envoie ce paquet sur le réseau. Les paquets qui ne bénéficient pas d'un SLA sont affectés à une classe *best-effort*. Une classe QoS peut par exemple être définie par un délai de transmission maximum du paquet ou l'assurance que le paquet sera délivré au routeur puits (même si le réseau est surchargé). La classe QoS associée à un paquet est inscrite dans l'entête du protocole réseau du paquet (par exemple IP ou MPLS) par le routeur source. Chaque paquet est ensuite routé par les routeurs du réseau en fonction de sa classe QoS. Certaines classes QoS correspondent à l'activation de méthodes ou protocoles QoS, tel que le protocole RSVP ou le routage par contraintes.

Le protocole "*Resource ReSerVation Protocol*" (RSVP) [XN99] permet de réserver la bande passante sur un chemin entre deux routeurs du réseau. Seuls les paquets bénéficiant de la classe QoS correspondante peuvent alors utiliser la bande passante réservée. Il existe aussi différentes stratégies de gestion des files d'attente des paquets dans les routeurs permettant de prendre en compte différentes classes QoS.

Ces stratégies de gestion de file permettent par exemple de faire suivre en priorité les paquets bénéficiant d'une classe QOS supérieure.

Le routage par contraintes (*constraint-based routing*, CBR) [XN99] est une autre méthodologie permettant de prendre en compte la qualité de service.

4.6.3 Le routage par contraintes

Le routage par contraintes consiste à construire, pour certaines classes QOS, des chemins respectant des contraintes (définies par des bornes inférieures ou supérieures) sur des objectifs de qualité de service. Les paquets associés avec ces classes QOS sont alors assurés par exemple d'emprunter des chemins dont les liens sont peu chargés ou avec un faible délai de transmission. La contrainte d'un objectif à minimiser est définie par une borne supérieure et la contrainte d'un objectif à maximiser est définie par une borne inférieure. Le routage par contraintes, si plus d'un objectif est pris en compte, nécessite la résolution d'un problème de plus court chemin multi-objectif sous contraintes. On note que le routage par contraintes est particulièrement utile en collaboration avec le mécanisme MPLS pour faire suivre les paquets ou/et le protocole RSVP [XN99].

Younis et Fahmi [YF03] proposent un état de l'art sur le problème de routage par contraintes. Kuipers et al [KMKK02, KKKM04] ainsi que Van Mieghem et al [MKK+03] présentent des états de l'art sur les algorithmes de routage par contraintes.

4.6.3.1 Définition des problèmes de plus court chemin multi-objectif sous contraintes

Dans le cadre du problème de plus court chemin multi-objectif sous contraintes, on conserve les notations présentées dans le cadre du problème MOSP (cf. Chapitre 2, page 43). Soit un graphe $G = (N, A, c)$, avec N l'ensemble des nœuds, A l'ensemble des arcs et $c : A \rightarrow \mathbb{R}_+^p$ la fonction qui définit les attributs de coût ou de QOS sur les arcs.

Soient $s \in N$ un routeur source pour un paquet donné et $t \in N$ le routeur puits de ce même paquet. Chaque lien du graphe est associé avec un vecteur de p attributs : $c(a) \in \mathbb{R}_+^p, \forall a \in A$. On considère p objectifs, chacun correspond à un attribut. Dans la littérature sur le routage par contraintes [XN99, YF03] trois types d'objectifs sont considérés. Un objectif $k \in P$ est dit additif si l'opérateur binaire associé est une addition ($\otimes_k = +$), il est dit multiplicatif si l'opérateur binaire est une multiplication ($\otimes_k = \times$), dans les deux cas cet objectif est à minimiser ($\leq_k \Leftrightarrow \leq$). Un objectif $k \in P$ est dit bottleneck si l'opérateur binaire est une fonction minimale ($\otimes_k = \min$), cet objectif doit être maximisé ($\leq_k \Leftrightarrow \geq$).

Les attributs de Qualité de Service [YF03] des liens peuvent être :

- La bande passante maximale est associé à un objectif bottleneck.
- La bande passante non réservée est associé à un objectif bottleneck.
- La bande passante résiduelle définie comme la bande passante maximale non utilisée du lien, cet attribut est associé à un objectif bottleneck.
- Le délai de transmission (ou délai) défini comme le temps de latence du lien, cet attribut est associé à un objectif additif.
- La variation du délai de transmission (ou jitter) définie comme la variation du temps de latence du lien, cet attribut est un objectif additif.
- Le nombre de *hops* (ou nombre de sauts) défini comme le nombre de liens d'un chemin, cet attribut est associé à un objectif additif. La valeur de l'attribut est de 1 pour chaque lien.

- La probabilité de perte de paquet est associé à un objectif multiplicatif. Notons que cet attribut peut dans certains cas être remplacé par l'attribut du logarithme de la probabilité de perte, ce deuxième attribut est alors associé à un objectif additif [RP11].

On désigne par $z_k(r) \leq_k \beta_k$ la contrainte d'un objectif $k \in P$, avec β_k la borne inférieure ou supérieure. Dans le cas du problème MOSP sous contraintes, une *solution admissible* est une solution vérifiant l'ensemble des contraintes du problème. Dans le cadre général du problème de plus court chemin multi-objectif sous contraintes, des contraintes sont définies sur certains objectifs, d'autres objectifs non contraints doivent être optimisés. On définit le problème de plus court chemin $\overline{l\text{-}\otimes}\ t\text{-}\otimes$ (avec $p = l + t$) comme le problème de plus court chemin multi-objectif avec l objectifs quelconques sur lesquels sont définis des contraintes et t objectifs quelconques à optimiser.

Tout problème MOSP sous contraintes contenant au moins deux objectifs additifs (ou multiplicatifs) est NP-complet [WC96], que ces objectifs soient contraints ou à optimiser. Notons que des objectifs bottleneck contraints peuvent être aisément ajoutés à un problème MOSP. Il suffit de supprimer, a priori de la résolution du problème MOSP, les liens du graphe ne respectant pas les contraintes [KKKM04, Beu06].

Plusieurs problèmes MOSP sous contraintes ont été étudiés dans la littérature. Le problème dit "*Restricted Shortest Path*" (RSP) [KKKM04] est un problème MOSP comportant un objectif additif de délai de transmission contraint par une borne supérieure et un objectif additif de coût à minimiser : $\overline{1\text{-}\sum}\ 1\text{-}\sum$. Le problème dit "*Multi-Constrainted Path*" (MCP) [KMKK02] est un problème MOSP comportant des objectifs additifs contraints : $\overline{Q\text{-}\sum}$. Des états de l'art pour le problème MCP sont proposés dans les articles suivants [KKKM04, MKK+03, YF03, KMKK02]. Le problème dit "*Multi-Constrainted Optimal Path*" (MCOP) [MKK+03] est un problème MOSP comportant des objectifs additifs contraints ainsi qu'un objectif additif à minimiser : $\overline{Q\text{-}\sum}\ 1\text{-}\sum$.

4.6.3.2 Des algorithmes pour des problèmes de plus court chemin sous contraintes

On présente dans cette sous-section 4 algorithmes pour des problèmes de plus court chemin sous contraintes. Ces algorithmes permettent de résoudre le problème MOSP ($R_{s,t}$) d'un nœud s à un nœud t. Pour illustrer ces algorithmes on considère un problème de plus court chemin avec deux objectifs et trois solutions : r^a, r^b et r^c. Soient $z(r^a) = (4, 11)$, $z(r^b) = (8, 8)$ et $z(r^c) = (12, 4)$ les trois points associés à ces solutions. Une borne supérieure est définie sur chaque objectif telle que : $\beta_1 = 10$ et $\beta_2 = 10$. Uniquement la solution r^b est admissible.

Jaffe propose [Jaf84] un algorithme pour le problème $\overline{Q\text{-}\sum}$, cet algorithme construit une solution qui n'est pas nécessairement admissible. En effet cet algorithme optimise une somme pondérée φ_λ dans le problème équivalent $Q\text{-}\sum$ qui ne prend pas en compte les contraintes sur les objectifs. Ainsi dans l'exemple illustré par la figure 4.6, la solution construite par l'algorithme de Jaffe est r^a et cette solution n'est pas admissible (elle ne respecte pas les contraintes sur les objectifs), alors qu'il existe une solution admissible r^b. Les poids $\lambda_k, k \in P$ de la somme pondérée φ_λ sont calculés en fonction des bornes des contraintes $\beta_1, \ldots, \beta_p$: $\lambda_k = \frac{1}{\beta_k}$ [MKK+03]. Dans l'exemple de la figure 4.6, le vecteur de poids est $\lambda = (0.25, 0.25)$. La somme pondérée vérifiant la monotonie dans le cadre du problème $Q\text{-}\sum$ (cf. Définition 2.12, page 67), un algorithme d'étiquetage mono-objectif peut être utilisé. On note qu'une variante de cet algorithme [LR01] consiste à utiliser un algorithme de ranking mono-objectif afin de construire les solutions en ordre croissant selon la somme pondérée φ_λ jusqu'à ce qu'une solution respectant toutes les contraintes soit construite.

L'algorithme TAMCRA [NM99] est proposé pour le problème MCP, $\overline{Q\text{-}\sum}$. Il s'agit d'un algorithme de label setting (LS). Comme dans l'algorithme de Jaffe, les auteurs proposent d'optimiser une fonction d'agrégation dans le problème $Q\text{-}\sum$ équivalent sans prendre en compte les contraintes. Plutôt que

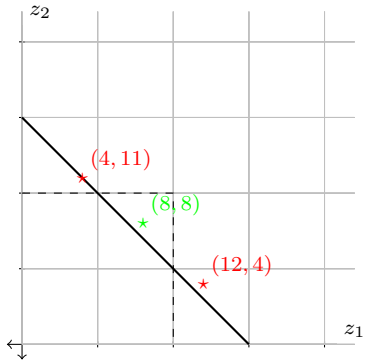

Figure 4.6 – Minimisation d'une somme pondérée $\varphi_{(0.25,0.25)}$, utilisée dans l'algorithme de Jaffe. Le trait en gras représente la courbe de niveau de la somme pondérée. Les étoiles ($\star$) représentent les points des solutions efficaces.

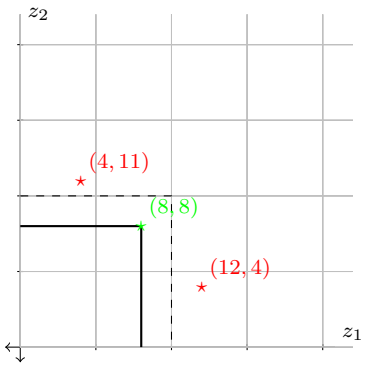

Figure 4.7 – Minimisation d'une fonction d'agrégation maximale pondérée avec $\lambda = (0.5, 0.5)$, utilisée dans les algorithmes TAMCRA, SAMCRA. Le trait en gras représente la courbe de niveau de la fonction d'agrégation maximale pondérée. Les étoiles ($\star$) représentent les points des solutions efficaces.

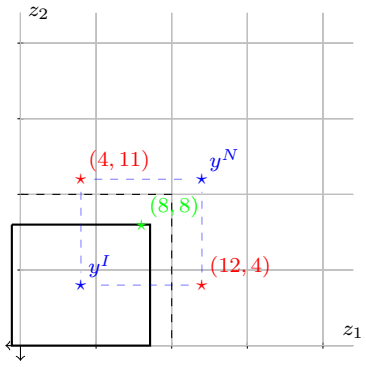

Figure 4.8 – Minimisation d'une norme de Tchebychev $\Upsilon_{(\frac{1}{14},\frac{1}{16})}$ par rapport au point idéal $y^I = (4, 4)$ des solutions efficaces, utilisée dans l'algorithme RMC. Le trait en gras représente la courbe de niveau de la norme de Tchebychev. Les étoiles ($\star$) représentent les points des solutions efficaces. y^N est le point nadir des solutions efficaces.

d'optimiser une somme pondérée, les auteurs proposent d'optimiser une fonction maximale pondérée $V : \mathbb{R}^p_+ \to \mathbb{R}_+ : V(y) = \max_{k \in P} \frac{y_k}{\beta_k}, \forall y \in \mathbb{R}^p$. Dans notre exemple, illustré par la figure 4.7, la solution optimale selon V est r^b. Cette solution est admissible. Les auteurs montrent que toute solution optimale selon V est admissible. L'algorithme TAMCRA peut être défini par rapport à l'algorithme d'étiquetage général (cf. Algorithme 2.1, page 51). Considérons $\mathfrak{L}$ l'ensemble des labels ouverts, L_i l'ensemble des labels (ouverts ou fermés) associés avec le nœud $i \in N$ et ℓ_i un label associé avec le nœud $i \in N$. On défini l'algorithme TAMCRA de la manière suivante :

1) La fonction de sélection h est la fonction d'agrégation $V : h = V$.
2) Une règle de coupe (cf. Définition 2.9, page 59) est utilisée telle que un label ℓ est supprimé si $V(\ell) > 1$. Cette règle de coupe implique que tous les labels ne vérifiant pas une contrainte sont supprimés.
3) L'algorithme propage au plus l labels par nœud, avec $l \in \mathbb{N}_+$ un paramètre défini a priori. Quand l labels sont déjà associés au nœud $i \in N$ ($|L_i| = l$), alors un nouveau label ℓ_i est ajouté à l'ensemble des labels associé avec le nœud i, L_i et à l'ensemble des labels ouverts $\mathfrak{L}$ s.s.i. il existe un label ouvert $\ell_i \in L_i \cap \mathfrak{L}$ associé au nœud i qui soit plus mauvais que $\ell_i : V(\ell_i) < \max\{V(\ell'_i), \ell'_i \in L_i \cap \mathfrak{L}\}$. Si cette inégalité est vérifiée alors le plus mauvais label de L_i ($\arg\min_{\ell'_i \in L_i} V(\ell'_i)$) est supprimé sinon le nouveau label ℓ_i est supprimé.

Le troisième point implique que cet algorithme n'optimise pas de manière exacte la fonction V. Les solutions construites par l'algorithme TAMCRA ne sont pas nécessairement admissibles même si il existe une solution admissible. L'algorithme TAMCRA possède une complexité polynomiale par rapport au paramètre l. Notons que cet algorithme permet de résoudre un problème $R_{s,\bullet}$ et donc un problème $R_{s,t}$. Les évaluations numériques montrent qu'un petit paramètre $l \leq 5$ suffit pour que l'algorithme TAMCRA construise une solution optimale selon V dans 99% des cas.

L'algorithme SAMCRA [MNK01] est un algorithme de label setting (LS) exact basé sur l'algorithme TAMCRA permettant de résoudre un problème MCP ($\overline{Q\text{-}\sum}$). Cet algorithme propose d'optimiser une fonction maximale pondérée dans le problème Q-$\sum$ équivalent. La principale différence de l'algorithme SAMCRA, par rapport à l'algorithme TAMCRA, est que le nombre de labels propagés par nœud n'est pas limité (voir 3) ci-dessus). Comme TAMCRA, l'algorithme SAMCRA est une variation de l'algorithme d'étiquetage général (cf. Algorithme 2.1, page 51), la fonction de sélection des labels est identique à celle de TAMCRA (voir 1) ci-dessus). La règle de coupe est différente de celle utilisée dans TAMCRA (voir 2) ci-dessus) : un label ℓ est supprimé si $V(z(\ell)) \leq V(z(\ell^\star))$, avec $\ell^\star \in L_t$ le meilleur label connu associé avec le nœud puits t. Contrairement à TAMCRA, SAMCRA ne permet pas résoudre un problème $R_{s,\bullet}$, seulement un problème $R_{s,t}$.

L'algorithme RMC [Beu06] permet de résoudre exactement un problème MOSP avec un nombre indéfini d'objectifs additifs et un objectif bottleneck Q-$\sum$ 1-min. Sur chaque objectif une contrainte peut éventuellement être définie. Dans un premier temps, les arcs ne vérifiant pas la contrainte sur l'objectif bottleneck sont supprimés du graphe. L'ensemble maximal complet des solutions efficaces est calculées par l'algorithme de label setting de Gandibleux et al [GBR06] présenté dans la section 2.2.2.1. Les solutions efficaces ne vérifiant pas les contraintes sur les objectifs additifs (non admissibles) sont supprimées. Ensuite la solution optimale selon une norme de Tchebychev $\Upsilon_\alpha : \mathbb{R}^p_+ \to \mathbb{R}_+$ (cf. Définition 1.8, page 17) avec $\alpha \in \mathbb{R}$ est retournée. Cette norme de Tchebychev correspond à la distance par rapport au point idéal y^I (cf. Définition 1.3, page 13) des solutions efficaces et admissibles.

$$\Upsilon_\alpha(z(r)) = \max_{k \in P} \alpha_k.(z_k(r) - y^I_k)$$

avec α le vecteur de poids tel que $\alpha = (\frac{1}{p\times(y_1^N - y_1^I)}, \ldots, \frac{1}{p\times(y_p^N - y_p^I)})$, y^N le point nadir (cf. Définition 1.3, page 13). Notons que le poids associé avec l'objectif p bottleneck est négatif car $y_k^N > y_k^I$. Dans notre exemple, illustré par la figure 4.8, la solution optimale selon Υ_α est r^b. Contrairement aux algorithmes SAMCRA et TAMCRA, l'algorithme RMC prend en compte un objectif bottleneck dans la fonction d'agrégation optimisée, dans ce cas la norme de Tchebychev.

Les problèmes MCP peuvent également être résolus par la méthode MPS [CCP03] Il s'agit d'une méthode de ranking similaire à celles [CM82, MPRS07] présentées dans la section 2.3.2. On cite aussi l'algorithme de Hallam et al [HHW01] qui, bien que n'étant pas présenté dans le cadre du routage par contraintes, permet de résoudre un problème MCP. Cet algorithme est présenté dans la section 2.2.3.2.

4.6.3.3 Routage par contraintes et méthodes de traffic engineering

Pour Xiao et Ni [XN99] le routage par contraintes permet de mieux répartir la charge globale sur le réseau que le routage *best-effort*. Dans le cadre de l'utilisation du routage par contraintes avec plusieurs classe de QOS, les paquets appartenant à différentes classes de QOS mais partageant le même nœud source et puits, peuvent être routés sur des chemins différents. Ainsi la charge des paquets entre ce routeur source et puits est repartie entre les liens de ces chemins, ce qui peut alors permettre de mieux répartir la charge globale du réseau. La prise en compte de plusieurs objectifs de qualité de service permet de prendre en considération l'utilisation de liens qui sinon ne sembleraient pas intéressants du point de vue d'un (unique) objectif de coût. L'exemple suivant illustre ces arguments.

Exemple 4.4. Suite de l'exemple 4.3.
Soit un AS *défini par un ensemble de routeurs $a, \ldots, f$ (cf. Figure 4.9, de la présente page), avec a, d et f des routeurs de bordure. On considère un flux de paquets du routeur a au routeur d et un flux de paquets du routeur a au routeur f. Le protocole de routage prend en compte deux objectifs : le nombre de sauts et le délai.*

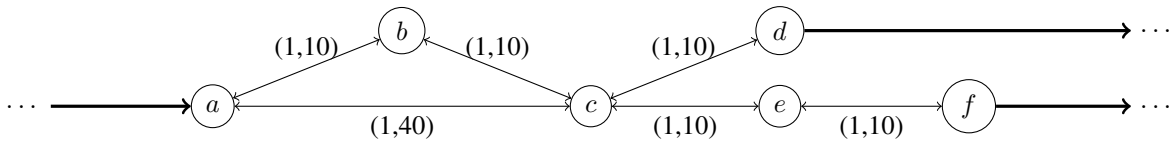

Figure 4.9 – Un réseau IP autonome défini par un ensemble de routeurs (nœuds) $a, \ldots, f$, avec a, d et f des routeurs de bordure. Deux attributs sont définis sur les liens du réseau : le premier est le nombre de sauts (1) et le second est le délai (2).

Comme vu précédemment, un routage prenant en compte uniquement le nombre de sauts, implique que la totalité du trafic (les flux de a à d et de a à f) entre les nœuds a et c passe par le lien (a, c) tandis que les liens (a, b) et (b, c) sont au repos. Dans l'exemple 4.3 nous montrons que l'utilisation de méthodes de traffic engineering permet de répartir équitablement le trafic sur les liens (a, b), (b, c) et (a, c).

La mise en place du routage par contraintes permet de répartir la charge du réseau grâce à :

- *La prise en compte de plusieurs objectifs : il n'existe pas un unique chemin "optimal" entre les deux nœuds a et c, mais un ensemble de chemins efficaces $\{\langle a, c\rangle, \langle a, b, c\rangle\}$. On considère une contrainte de $z(r) \leq 4$ sur le nombre de sauts et une contrainte de $z(r) \leq 50$ sur le délai. Alors parmi les paquets*

entrant par le routeur a, ceux sortant au routeur d seront routés sur le chemin $\langle a, c, d\rangle$ alors que ceux sortant au routeur f seront routés sur le chemin $\langle a, b, c, e, f\rangle$.

- *La prise en compte de plusieurs classes* QoS *: une première classe* QoS *consiste à minimiser le nombre de sauts. Une deuxième classe* QoS *définit des contraintes : $z(r) \leq 4$ sur le nombre de sauts et $z(r) \leq 40$ sur le délai. Alors pour les paquets des deux flux de a à d et de a à f, les paquets affectés à la première classe* QoS *sont routés sur le chemin $\langle a, c, ..\rangle$ et les paquets affectés à la deuxième classe* QoS *sont routés sur le chemin $\langle a, b, c, ...\rangle$.*

Dans ces deux cas la charge du réseau est répartie sur les arcs (a, b), (b, c) et (a, c). Néanmoins la charge du réseau n'est pas répartie de manière aussi équilibrée qu'avec les méthodes de traffic engineering *présentées dans l'exemple 4.3 précédent.*

Notons que le routage par contraintes ne permet pas de prendre en compte directement des informations sur les prévisions de trafic du réseau pour construire les chemins, contrairement aux méthodes de *traffic engineering* [FRT02].

4.7 Un algorithme pour le routage prenant en compte la qualité de service

Dans le cadre du routage prenant en compte la qualité de service, plusieurs objectifs de qualité de service peuvent être pris en compte en même temps. Des extensions des protocoles de routage OSPF et IS-IS sont capables d'informer chaque routeur du réseau de l'état du réseau, i.e. de la topologie du graphe et des attributs de qualité de service ou de coût de chaque arc. La plupart des paquets routés dans un réseau sont routés sur des chemins aux coûts minimaux (routage *best-effort*). Certains paquets bénéficiant d'un SLA doivent être routés sur des chemins de qualité supérieure, i.e. des chemins satisfaisant certaines exigences de qualité de service.

Pour construire ces chemins l'administrateur doit prendre en compte plusieurs objectifs de qualité de service tels que ceux décrit dans la sous-section précédente (cf. page 121) et éventuellement une métrique de coût. On peut alors parler de routage multi-objectif. Comme expliqué dans la sous-section 1.1.2, il est alors nécessaire de définir une méthode pour sélectionner une solution parmi l'ensemble des solutions efficaces de ce problème de routage multi-objectif. Dans ce cadre, le routage par contraintes propose de définir des contraintes sur les objectifs, puis de construire des chemins satisfaisant ces contraintes.

Nous proposons dans cette section d'utiliser une méthode d'aide à la décision pour définir une fonction d'utilité composée d'une intégrale de Choquet et de fonctions d'utilité partielles linéaires par morceaux plutôt que de définir des contraintes. Un algorithme d'étiquetage utilisant la dominance selon une fonction d'utilité peut alors être utilisé pour construire des solutions optimales selon cette fonction d'utilité.

4.7.1 Modélisation des préférences dans le cadre du routage prenant en compte des objectifs de qualité de service

Illustrons sur un objectif les avantages d'une fonction d'utilité partielle linéaire par morceaux par rapport à une contrainte. Soit un objectif de délai $k \in P$. On s'intéresse au routage d'un paquet de données émit par une application de vidéoconférence. Soit r le chemin emprunté par ce paquet. Dans le cadre du routage par contraintes, l'administrateur définit une contrainte sur l'objectif de délai : $z_k(r) \leq \beta_k$. Cette contrainte étant définie par l'administrateur, alors elle correspond à une modélisation (simple) de ses préférences sur cet objectif. Cette contrainte implique que r est pleinement satisfaisant sur cet objectif pour l'administrateur si il vérifie cette contrainte, mais n'est aucunement satisfaisant sur cet

objectif si il ne vérifie pas cette contrainte. La satisfaction l'administrateur sur cet objectif peut être modélisé par une fonction d'utilité partielle indicatrice : $u(y_k) = 0$ si $y^k \leq \beta_k$ et $u(y_k) = 1$ si $y^k > \beta_k$. Cette fonction indicatrice est (sous certaines conditions) équivalente à la contrainte $z_k(r) \leq \beta_k$. Ces fonctions indicatrices sont illustrées par les figures 4.18, 4.19, 4.20.

En général l'administrateur va définir une telle contrainte par rapport à une exigence de délai maximum de l'application de vidéoconférence émettant ce paquet [XN99]. Or les performances d'une application de vidéoconférence par rapport à un objectif de délai ne correspondent pas nécessairement à une fonction indicatrice. Ainsi on peut considérer qu'au dessus d'une certain valeur de délai β_k^a, l'application ne fonctionnera pas ou très mal, en dessous d'une autre valeur de délai β_k^b, le délai n'impactera pas les performances de l'application pour l'utilisateur. Si ces deux valeurs sont égales ($\beta_k^a = \beta_k^b$) alors cette valeur permet de définir clairement une contrainte (ou une fonction indicatrice) sur l'objectif de délai. Mais en général ces deux valeurs ne sont pas égales et il existe des valeurs de délai intermédiaires entre ces deux valeurs pour lesquelles l'application fonctionnera toujours mais pas de manière optimale. Les performances de l'application sur cet objectif de délai peuvent alors être définies plus précisément avec une fonction linéaire par morceaux qu'avec une contrainte. Une fonction linéaire par morceaux peut non seulement approximer une fonction indicatrice (et donc une contrainte), mais peut aussi approximer n'importe quelle autre fonction continue représentant les performances de l'application par rapport au délai, par exemple une fonction exponentielle. Les préférences de l'administrateur sur cet objectif peuvent être modélisées plus précisément avec une fonction d'utilité partielle linéaire par morceaux. Une telle modélisation des préférences de l'administrateur est illustrée par les fonctions d'utilité partielles des figures 4.15, 4.16, 4.17.

Considérons maintenant l'ensemble des objectifs. Dans le cadre du problème de routage MCP, une contrainte sur chaque objectif est définie. Pour qu'un chemin soit satisfaisant sur l'ensemble des objectifs, ce chemin doit vérifier la contrainte sur chaque objectif. C'est-à-dire, un chemin satisfaisant sur chaque objectif. Ainsi le non respect d'une contrainte sur un objectif (i.e. un objectif non satisfaisant) ne peut pas être compensé par une amélioration des performances sur tous les autres objectifs. Une telle modélisation des préférences est équivalente à la notion de complémentarité entre les objectifs (cf. §1.2.4, page 30) telle que définie en aide à la décision multi-critère. Nous avons vu que la minimisation d'une fonction d'agrégation maximale pondérée agrégeant les objectifs correspond à une complémentarité totale entre les objectifs, De Neve et Van Mieghem [NM99] montrent que la minimisation d'une telle fonction d'agrégation permet de construire une solution vérifiant l'ensemble des contraintes si une telle solution existe. Les algorithmes SAMCRA, TAMCRA utilisent cette méthode pour calculer un chemin vérifiant ces contraintes. Dans le cadre de l'algorithme RMC, l'ensemble des chemins efficaces est calculé, ensuite les chemins ne vérifiant pas les contraintes sont supprimés. Une norme de Tchebychev est ensuite utilisée pour sélectionner des chemins parmi les chemins vérifiant les contraintes. La norme de Tchebychev modélise aussi une complémentarité totale entre les objectifs. L'utilisation d'une norme de Tchebychev ou d'une fonction d'agrégation maximale pondérée indique que l'administrateur souhaite calculer un chemin de compromis sur l'ensemble des objectifs (cf. §1.1.2.3, page 16).

La recherche d'un chemin de compromis sur l'ensemble des objectifs ne permet pas toujours de calculer un chemin satisfaisant. Prenons un exemple illustré par la figure 4.10. Soient r^a, r^b et r^c trois chemins. Selon la norme de Tchebychev utilisée, le chemin r^a avec un délai de 10000μs et une bande passante de 12Mb/s est préféré au chemin r^b avec un délai de 3000μs et une bande passante de 11.5Mb/s et au chemin r^c avec un délai de 10500μs et une bande passante de 22Mb/s. Pourtant le chemin r^b semble plus intéressant que le chemin r^a, en effet une amélioration de la bande passante de 0.5Mb/s peut difficilement compenser une baisse du délai de $7000\mu s$. Pour des raisons identiques, le chemin r^c semble plus intéressant que le chemin r^a.

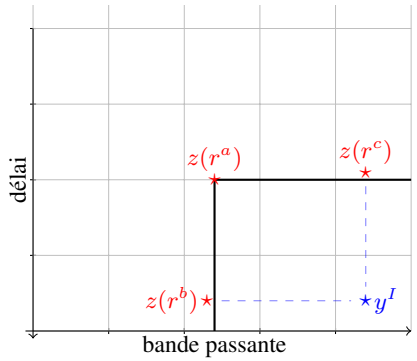

Figure 4.10 – Trois points rouges dans l'espace des objectifs représentant trois chemins r^a, r^b, r^c. Le point bleu est le point idéal y^I. Le trait en gras représente la courbe de niveau de la norme de Tchebychev par rapport au point idéal. Les chemin optimal selon la norme de Tchebychev est r^a.

Si il est intéressant de modéliser une complémentarité entre les objectifs il est aussi intéressant de modéliser une compensation (i.e. pondération) entre ces objectifs. Pour cela l'intégrale de Choquet permet de modéliser à la fois une complémentarité et une compensation entre les objectifs. Notons que l'algorithme de Jaffe [Jaf84] optimise une somme pondérée, cette somme pondérée permet une compensation totale entre les objectifs. L'intégrale de Choquet permet de modéliser des complémentarités non seulement entre l'ensemble des objectifs mais aussi entre certains sous-ensembles d'objectifs. Prenons un objectif de délai et un objectif de variation de délai. On peut considérer que ces deux objectifs sont complémentaires tel que l'on va préférer un chemin avec une variation de délai moyennement satisfaisante et un délai moyennement satisfaisant plutôt qu'un chemin avec un délai peu satisfaisant mais une variation de délai satisfaisante. L'intégrale de Choquet permet aussi de modéliser des substituabilités entre les objectifs (cf. §1.2.4, page 30). Prenons un objectif de délai et un objectif de probabilité de perte des paquets. On peut considérer que ces deux objectifs sont en partie substituables. Un chemin avec un délai satisfaisant mais une probabilité de perte peu satisfaisante peut être préféré à un chemin avec un délai moyennement satisfaisant et une probabilité de perte moyennement satisfaisante. En effet un délai faible permet d'atténuer une probabilité de perte élevée car les paquets perdus peuvent être renvoyé plus rapidement.

Pour toutes les raisons énoncées ci dessus, une fonction d'utilité Ψ^u_μ composée d'une intégrale de Choquet C_μ et de fonctions d'utilité partielles linéaires par morceaux $u_1, \ldots, u_p$ est plus capable de modéliser les préférences d'un administrateur d'un réseau dans le cadre de la prise en compte de plusieurs objectifs de qualité de service :

$$\Psi^u_\mu(z(r)) = C_\mu(u_1(z_1(r)), \ldots, u_p(z_p(r))),\ \forall r \in \mathbb{R}^{\bullet,\bullet}$$

avec $u_k, k \in P$ des fonctions d'utilité partielle monotones linéaires par morceaux et C_μ l'intégrale de Choquet.

Cette fonction d'utilité offre beaucoup plus de possibilités de modélisations des préférences de l'administrateur que des contraintes sur les objectifs, une fonction d'agrégation maximale pondérée, une somme pondérée ou une norme de Tchebychev. Ainsi l'intégrale de Choquet permet de favoriser les solutions de compromis tout en permettant une compensation entre les critères. Les figures 4.10 et 4.11 illustrent les avantages de l'intégrale de Choquet par rapport à une norme de Tchebychev ou une fonction d'agrégation $\max$. On sait que l'optimisation d'une fonction d'agrégation maximale pondérée ou d'une

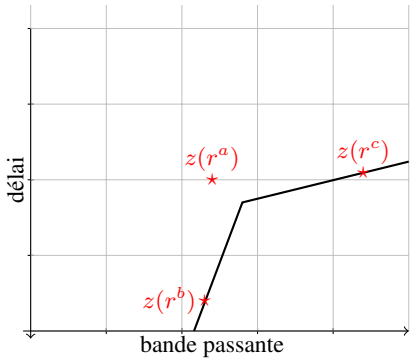

Figure 4.11 – Trois points rouges dans l'espace des objectifs représentant trois chemins r^a, r^b, r^c. Le trait en gras représentent la courbe de niveau d'une intégrale de Choquet. Les chemins optimaux selon cette intégrale de Choquet sont r^b et r^c. Cette intégrale de Choquet favorise les chemins de compromis (comme r^a) mais permet une compensation entre l'objectif de délai et celui de bande passante. L'excellent délai du chemin r^b lui permet de compenser sa bande passante légèrement moins bonne que celle de r^a.

norme de Tchebychev permet de résoudre le problème de routage MCP. L'optimisation d'une fonction d'utilité Ψ_μ^u permet de résoudre non seulement un problème MCP mais aussi un problème MCOP (cf. page 122). Les relations entre les problèmes et méthodes dans le cadre du routage multi-objectif sont représentées par la figure 4.12.

Dans l'exemple 4.5, un problème de routage multi-objectif est défini. Et différentes approches pour résoudre ce problème sont présentées.

Exemple 4.5. Routage multi-objectif.
On utilise dans cet exemple le réseaux EUROPE2 *composé de* 16 *routeurs et de* 20 *liens (cf. Figure 4.13, de la présente page). Trois objectifs sont considérés, un objectif additif à minimiser modélisant le délai de transmission (z_1), un objectif bottleneck modélisant la bande passante (z_2), un objectif additif à minimiser modélisant le nombre de sauts (z_3). Dans le cadre du routage multi-objectif, on cherche à construire un chemin pour un paquet du routeur/nœud* 7 *au routeur/nœud* 11. *Nous considérons dans cet exemple une approche a posteriori (calcul de l'ensemble des solutions efficaces puis sélection de l'une d'entre elles) afin d'illustrer la modélisation des préférences dans le cadre du routage multi-objectif. Comme expliqué dans la section 1.3, dans le cadre du routage dans les réseaux l'approche a priori (ou décision automatique) semble être l'approche la plus évidente à mettre en œuvre contrairement à l'approche a posteriori. Les chemins efficaces entre ces deux nœuds sont illustrés dans la figure 4.14 et dans le tableau 4.6.*

Afin de sélectionner un chemin du nœud 7 *vers le nœud* 11 *pour un flux de paquets, l'administrateur peut définir des contraintes sur les objectifs. L'administrateur estime qu'il est préférable que le délai soit inférieur à* $15000\mu s$ *:* $z_1(r) \leq 15000$ *μs, la bande passante soit supérieure à* 300*Mb/s :* $z_2(r) \geq 300$ *Mb/s, le nombre de sauts inférieur à* 10 *:* $z_3(r) \leq 10$. *Seuls les chemins r^b et r^c vérifient ces trois contraintes. Ces contraintes sont équivalentes aux fonctions d'utilité indicatrices u'_1, u'_2, u'_3 définies sur les figures 4.15, 4.16 et 4.17. Dans le cadre de l'algorithme* RMC *le chemin r^b est sélectionné à l'aide d'une norme de Tchebytchev Υ_α. Dans le cadre des algorithmes* SAMCRA *et* TAMCRA *le chemin r^b est sélectionné à l'aide de la fonction maximale pondérée. Dans le cadre de l'algorithme de Jaffe c'est aussi le chemin r^b qui est sélectionné à l'aide de la somme pondérée.*

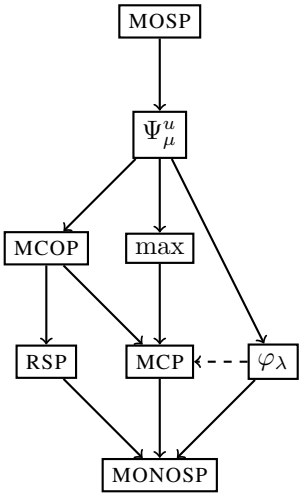

Figure 4.12 – Relations entre différent problèmes de plus court chemins. Le problème de plus court chemin multi-objectif (MOSP) implique le calcul de l'ensemble des solutions efficaces (algorithme RMC). Les problèmes RSP,MCP,MCOP,MONOSP impliquent le calcul d'une solution admissible selon les contraintes et optimale selon l'objectif considéré. Les problèmes notés par une fonction d'agrégation φ_λ, $\max$ ou Ψ_μ^u impliquent le calcul d'une solution minimale selon cette fonction d'utilité. Ces fonctions d'utilité sont une somme pondérée φ_λ, une fonction maximale pondérée notée $\max$ (ou une norme de Tchebychev) ou une fonction d'utilité composée d'une intégrale de Choquet et de fonctions d'utilité partielles Ψ_μ^u. La flèche pleine entre $\max$ et MCP signifie que la minimisation d'une fonction maximale pondérée permet de résoudre le problème MCP exactement. La flèche en pointillé entre φ_λ et MCP signifie qu'un algorithme optimisant la fonction φ_λ permet de résoudre le problème MCP de manière approximative (voir algorithme de Jaffe [Jaf84]).

chemins		z_1	z_2	z_3
r^a	$\langle 7,8,9,15,10,11\rangle$	5750	226621	5
r^b	$\langle 7,6,5,3,4,16,15,10,11\rangle$	6250	372461	8
r^c	$\langle 7,6,5,3,4,16,15,9,14,10,11\rangle$	12000	420644	10
r^d	$\langle 7,8,1,2,3,4,16,15,9,14,12,11\rangle$	21250	443005	11
r^e	$\langle 7,8,1,2,3,4,16,15,9,14,10,11\rangle$	17250	429359	11

Table 4.5 – Chemins efficaces de 7 à 11 sur le réseau EUROPE2 et leurs performances sur les trois objectifs. Le premier objectif (z_1) modélise le délai. Le deuxième objectif (z_2) modélise la bande passante. Le troisième objectif (z_3) modélise le nombre de sauts.

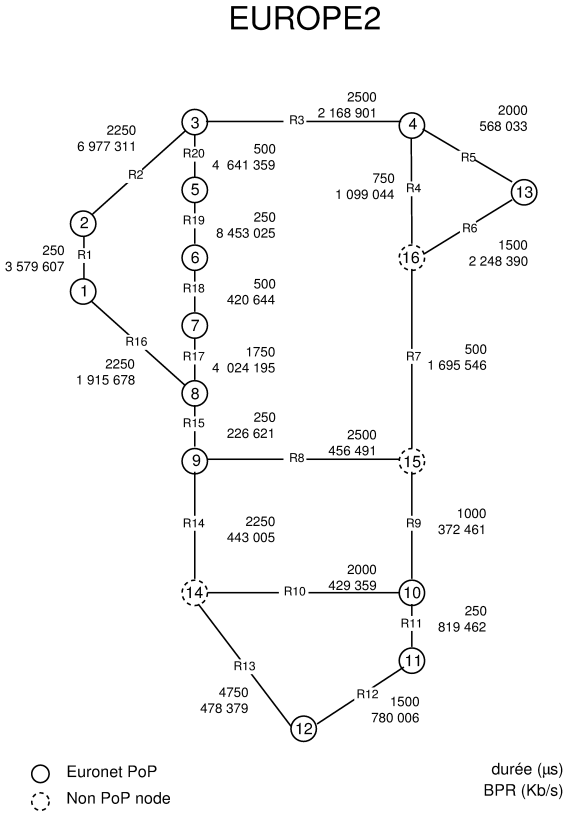

Figure 4.13 – Graphe du réseau EUROPE2.

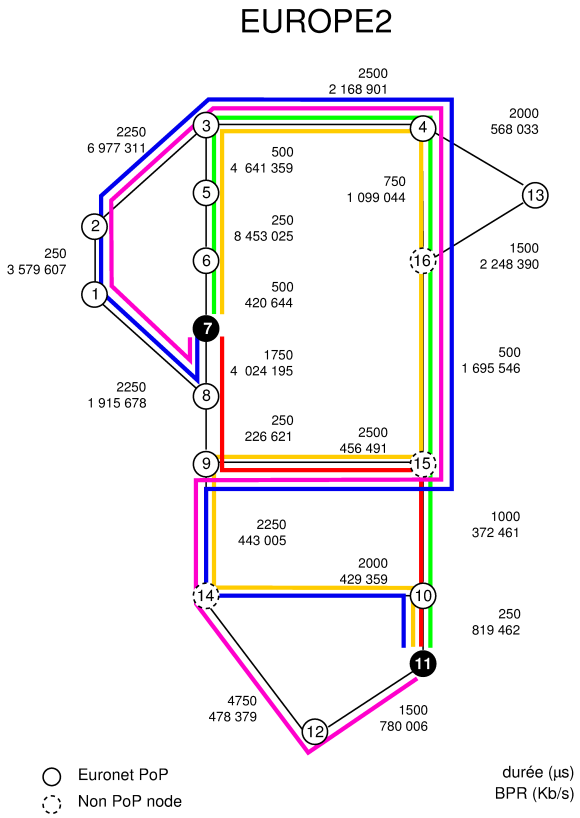

Figure 4.14 – Graphe du réseau EUROPE2, les chemins efficaces de 7 vers 11 sont indiqués par des couleurs.

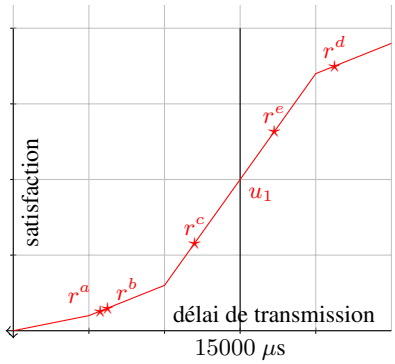

Figure 4.15 – Fonction d'utilité partielle linéaire par morceaux croissante u_1 associée avec l'objectif de délai. Les performances des chemins efficaces r^a, r^b, r^c, r^d et r^e sur cet objectif sont indiquées.

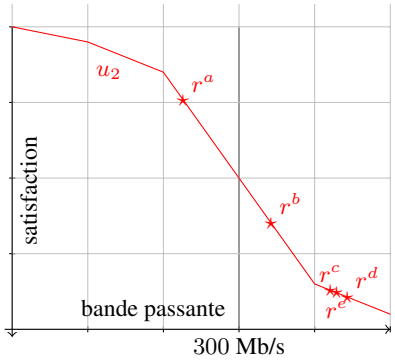

Figure 4.16 – Fonction d'utilité partielle linéaire par morceaux décroissante u_2 associée avec l'objectif de bande passante. Les performances des chemins efficaces r^a, r^b, r^c, r^d et r^e sur cet objectif sont indiquées.

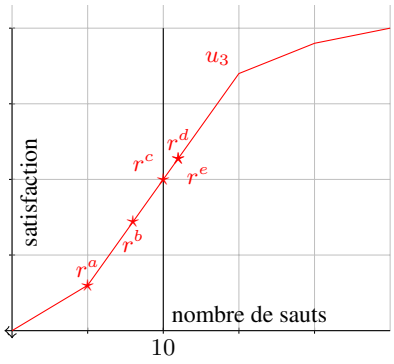

Figure 4.17 – Fonction d'utilité partielle linéaire par morceaux croissante u_3 associée avec l'objectif du nombre de sauts. Les performances des chemins efficaces r^a, r^b, r^c, r^d et r^e sur cet objectif sont indiquées.

Nous proposons à l'administrateur de définir une fonction d'utilité Ψ^u_μ composée d'une intégrale de Choquet C_μ et de fonctions d'utilité partielles $u_1, \ldots, u_p$. Des fonctions d'utilité partielles linéaires par morceaux sont utilisées pour modéliser les préférences de l'administrateur sur chaque objectif. Sur chaque objectif, l'administrateur souhaite définir une fonction d'utilité défavorisant les chemins ne respectant la contrainte sur cet objectif définie précédement. Ces fonctions d'utilité partielles linéaires par morceaux u_1, u_2, u_2 sont définies sur les figures 4.15, 4.16 et 4.17. L'agrégation des objectifs est réalisée avec une intégrale de Choquet. L'administrateur estime que l'objectif de bande passante et l'objectif délai sont les deux objectifs les plus importants. De plus l'administrateur préfère une solution de compromis : il préfère une solution satisfaisante sur l'ensemble des objectifs plutôt qu'une solution très satisfaisante sur certains et peu satisfaisante sur d'autres. Il considère aussi que les objectifs de délai et de nombre de sauts sont complémentaires, car le nombre de sauts peut influer sur le délai négativement surtout si le réseau est surchargé. L'intégrale de Choquet C^l_μ est définie par la fonction de capacité μ^l telle que : $\mu^l(\emptyset) = 0.0$, $\mu^l(\{1\}) = 0.5$, $\mu^l(\{2\}) = 0.5$, $\mu^l(\{3\}) = 0.5$, $\mu^l(\{1,2\}) = 0.9$, $\mu^l(\{1,3\}) = 0.6$, $\mu^l(\{2,3\}) = 0.9$ et $\mu^l(\{1,2,3\}) = 1$. ξ est l'échelle de satisfaction sur laquelle est évaluée la satisfaction des chemins sur chaque objectif et sur l'ensemble des objectifs pour le décideur. ξ est l'ensemble des réels entre 0 et 20 : $\xi = [0, 20]$. Les utilités de chaque chemin définies sur l'échelle de satisfaction ξ selon chaque fonction d'utilité $u_1, \ldots, u_3$ et selon la fonction d'utilité $\Psi^u_{\mu^l}$ sont données dans la table 4.6. Selon la fonction d'utilité $\Psi^u_{\mu^l}$ le chemin r^c est préféré.

Des contraintes sur les objectifs peuvent (aisément) être déterminées par l'administrateur du réseau dans le cadre d'un *service level agreement* avec un client. Une fonction d'utilité Ψ^u_μ est un modèle (de préférence) beaucoup plus complexe que des contraintes sur les objectifs. Ceci constitue la principale difficulté pour la construction de cette fonction d'utilité. Si on considère que l'administrateur est capable de donner des informations sur ces préférences, concernant les chemins empruntés par les paquets, alors

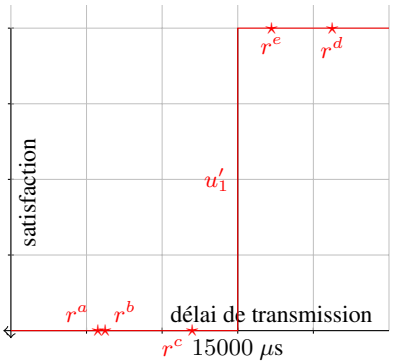

Figure 4.18 – Fonction d'utilité partielle indicatrice u'_1 associée avec l'objectif de délai. Les performances des chemins efficaces r^a, r^b, r^c, r^d et r^e sur cet objectif sont indiquées.

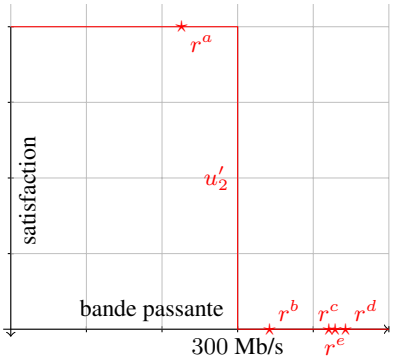

Figure 4.19 – Fonction d'utilité partielle indicatrice u'_2 associée avec l'objectif de la bande passante. Les performances des chemins efficaces r^a, r^b, r^c, r^d et r^e sur cet objectif sont indiquées.

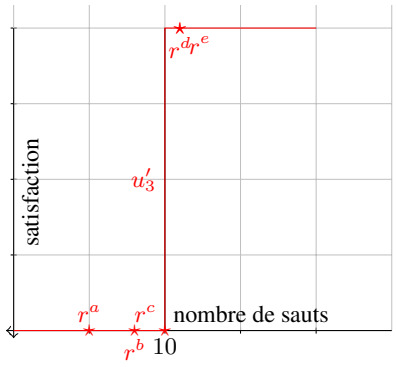

Figure 4.20 – Fonction d'utilité partielle indicatrice u'_3 associée avec l'objectif du nombre de sauts. Les performances des chemins efficaces r^a, r^b, r^c, r^d et r^e sur cet objectif sont indiquées.

chemins	$u_1(z_1)$	$u_2(z_2)$	$u_3(z_3)$	$\Psi^u_{\mu^l}(z)$
r^a	1.3	15.138	3	8.899
r^b	2.5	7.032	7.2	6.663
r^c	5.8	2.588	10	6.615
r^d	17.5	2.140	11.4	10.746
r^e	13.15	2.416	11.4	8.681

Table 4.6 – Utilités des chemins efficaces de 7 à 11 du réseau EUROPE2 sur chaque objectif ($u_1(z_1)$, $u_2(z_2)$ et $u_3(z_3)$) et globalement ($\Psi^u_\mu(z)$).

il est possible d'utiliser les méthodes existantes [LL05, GKM08, GL10], proposées dans le cadre de l'aide à la décision multi-critère, pour construire cette fonction d'utilité. Ces méthodes on été présentées dans la section 1.2.

4.7.2 Résolution du problème de plus court chemin

On considère que les préférences de l'administrateur, pour des paquets associés à un même SLA, soient modélisées a priori par une fonction d'utilité Ψ_μ^u composée d'une intégrale de Choquet et de fonctions d'utilité partielles linéaires par morceaux. L'algorithme de routage doit permettre à chaque routeur de calculer des chemins optimaux selon la fonction d'utilité Ψ_u^μ pour ces paquets.

Une première méthode consiste à construire l'ensemble de chemins efficaces puis de sélectionner un chemin optimal selon la fonction d'utilité Ψ_μ^u. Cette méthode est semblable à celle utilisée par l'algorithme RMC. Elle ne permet pas de prendre en compte la fonction d'utilité durant la résolution du problème. Notons que, quand au moins un objectif bottleneck est pris en compte, l'ensemble maximal complet des solutions efficaces peut être important [GBR06, IMP10].

Nous avons montré dans cette thèse différentes méthodes pour prendre en compte une fonction d'utilité durant la résolution du problème de plus court chemin multi-objectif, voir les chapitres 3 et 4. Dans le cadre du routage réactif, les routeurs doivent résoudre un problème de plus court chemin d'un nœud à un autre nœud ($R_{s,t}$), les travaux du chapitre 3 peuvent s'appliquer. Néanmoins si un objectif bottleneck (comme la bande passante) ou multiplicatif est pris en compte alors les bornes inférieures 3.3, 3.8 et 3.9 ne peuvent s'appliquer, la borne inférieure 3.3 peut s'appliquer mais difficilement. Notons que les travaux du chapitre 4 peuvent aussi être appliqués dans le cadre du routage réactif et permettent de prendre en compte des objectifs bottleneck et multiplicatifs.
Dans le cadre du routage proactif, les routeurs doivent résoudre un problème de plus court chemin d'un nœud à tous les nœuds ($R_{s,\bullet}$), les travaux du chapitre 4 peuvent s'appliquer. On considère dans la suite le cadre du routage proactif.

Dans le chapitre précédent 4 nous avons proposé d'utiliser une relation de dominance prenant en compte la fonction d'utilité à la place de la dominance de Pareto (ou de la dominance GBR, voir la définition 2.6 page 54) dans un algorithme d'étiquetage. Cette dominance, notée Ψ_μ^u-dominance dans notre cas, permet d'éviter de construire l'ensemble des solutions efficaces. Uniquement un sous-ensemble de solutions efficaces préférées au regard de la fonction d'utilité est construit. Comme nous l'avons montré la Ψ_μ^u-dominance est difficile à évaluer, pour cette raison nous avons proposé une condition suffisante de la Ψ_μ^u-dominance (cf. Théorème 4.2, page 107). Cette condition suffisante est utilisée à la place de la Ψ_μ^u-dominance dans un algorithme d'étiquetage et permet de supprimer plus de labels que la dominance de Pareto ou la dominance GBR.

Dans la table 4.7 on évalue l'efficacité de la Ψ_μ^u-dominance dans un algorithme d'étiquetage par rapport à la dominance GBR (cf. Définition 2.6, page 54) utilisée dans l'algorithme RMC pour des instances de plus court chemin mutli-objectif avec un objectif bottleneck. Pour cela on utilise les fonctions de capacité définies précédemment dans le tableau 4.1 page 130. Trois objectifs sont pris en compte, deux objectifs additifs et un objectif bottleneck. Les graphes sont générés avec le programme GGEN (cf. §4.4, page 110). Les temps de calcul de deux algorithmes de label setting avec la fonction lexicographique comme fonction de sélection sont comparés dans la table 4.7. Le premier algorithme d'étiquetage utilise la Ψ_μ^u-dominance (cf. §4.4.1, page 111), le deuxième algorithme est l'algorithme RMC (cf. §4.6.3.2, page 122) qui utilise la dominance GBR. Ces deux algorithmes diffèrent uniquement par la dominance utilisée.

fonction utilité	temps de calcul (sec)						déviation (%)	
	$m = 5 \times n$			$m = 50 \times n$		$m = 500 \times n$		
	1000	6000	10000	1000	6000	1000	max.	moy.
$\Psi^u_{\mu^a}$	0.005	0.048	0.094	0.124	1.021	2.083	18.20	6.75
$\Psi^u_{\mu^b}$	0.016	0.144	0.277	0.549	5.247	5.148	44.98	20.84
$\Psi^u_{\mu^c}$	0.007	0.065	0.126	0.152	1.297	2.054	17.947	7.94
$\Psi^u_{\mu^d}$	0.009	0.08	0.154	0.206	1.854	2.901	25.348	10.62
$\Psi^u_{\mu^e}$	0.179	1.998	3.906	5.976	69.676	20.879	248.61	194.75
$\Psi^u_{\mu^f}$	0.16	1.811	3.564	5.921	67.802	20.921	222.22	184.08
$\Psi^u_{\mu^g}$	0.015	0.147	0.287	0.633	6.429	6.543	57.17	23.76
$\Psi^u_{\mu^h}$	0.018	0.179	0.333	0.94	9.724	8.237	71.97	30.96
$\Psi^u_{\mu^l}$	0.017	0.166	0.303	0.833	8.455	7.872	64.57	27.84
RMC	0.072	1.134	2.31	2.7324	40.1676	11.4444	100	100

Table 4.7 – Expérimentations sur un problème de plus courts chemins d'un nœud à tous les nœuds ($R_{s,\bullet}$), n est le nombre de nœuds et m est le nombre d'arcs, avec 3 objectifs : 2 objectifs additifs et 1 objectif bottleneck. Comparaison des temps de calcul entre un algorithme d'étiquetage 2.1 utilisant la Ψ^u_μ-dominance et l'algorithme RMC qui utilise la dominance GBR. Les fonctions de capacité μ^a,...,μ^h sont définies dans la table 4.1 page 130, la fonction de capacité μ^l est définie dans l'exemple 4.5 précédent. Le vecteur de fonctions d'utilité partielles u est un vecteur de fonctions d'identité. La déviation (exprimée en pourcentage) est la moyenne des temps de calcul ou le temps de calcul maximum par rapport au temps de calcul de l'algorithme RMC.

La table 4.7 montre que l'utilisation de la Ψ_μ^u-dominance permet pour la plupart des fonctions d'utilité proposées de réduire les temps de calcul. Si l'on compare ces résultats à ceux du chapitre 4 précédent on remarque que : pour l'algorithme d'étiquetage utilisant la Ψ_μ^u-dominance, on obtient des temps de calcul plus longs sur des instances avec un objectif bottleneck (cf. Table 4.7) que sur des instances identiques sans objectif bottleneck (cf. Table 4.2, page 113). Cette différence des temps de calcul est normale puisque la condition suffisante définie par le théorème 4.2 est moins souvent vérifiée quand un objectif bottleneck est pris en compte. De même quand un objectif bottleneck est pris en compte, l'algorithme RMC est plus lent que un algorithme équivalent utilisant la dominance de Pareto (l'algorithme de Martins), car la dominance GBR permet de supprimer moins de labels que la dominance de Pareto. Notons que l'utilisation de la dominance de Pareto avec un objectif bottleneck ne permet pas de calculer l'ensemble maximal complet des solutions efficaces [GBR06]. Ce qui implique que l'algorithme de Martins ne permet pas de construire certains chemins optimaux selon la fonction d'utilité Ψ_μ^u contrairement à l'algorithme RMC ou à l'algorithme de label setting utilisant la Ψ_μ^u-dominance.

Pour les fonctions de capacité μ^e et μ^f modélisant des très fortes interactions entre les objectifs l'utilisation de la Ψ_μ^u-dominance implique des temps de calcul supplémentaires par rapport à l'utilisation de la dominance GBR. Ce problème est dû à l'évaluation de la condition suffisante de la Ψ_μ^u-dominance qui nécessite plus de temps que l'évaluation de la dominance GBR. Ce temps de calcul supplémentaire peut être réduit voire annulé en forçant l'algorithme d'étiquetage à ne pas évaluer cette condition suffisante quand la fonction de capacité modélise des interactions trop fortes. Pour cela nous avons proposé d'utiliser un seuil de différence (cf. §4.4.1, page 111). Sur ce point des travaux supplémentaires sont nécessaires.

4.8 Conclusion

Dans ce chapitre nous avons proposé d'utiliser une fonction d'utilité composée de fonctions d'utilité partielles linéaires par morceaux et d'une intégrale de Choquet afin de modéliser les préférences de l'administrateur dans le cadre du routage multi-objectif prenant en compte la qualité de service. Cette méthodologie peut être vue comme une généralisation du routage par contraintes.

Pour construire des chemins optimaux selon cette fonction d'utilité, un algorithme d'étiquetage utilisant la dominance selon cette fonction d'utilité (cf. Chapitre 4, page 99) peut être utilisé afin d'éviter de calculer l'ensemble des solutions efficaces. Dans le cadre du routage multi-objectif il est parfois utile de prendre en compte un objectif bottleneck modélisant la bande passante. Les résultats numériques montrent que l'utilisation de la dominance selon cette fonction d'utilité est efficace pour des problèmes multi-objectifs avec un objectif bottleneck.

De nombreuses perspectives de recherche existent suite à ce travail. Les perspectives concernant l'amélioration la méthode évaluant la dominance selon une fonction d'utilité ont été présentées dans la conclusion du chapitre précédent. La prise en compte de plusieurs objectifs (et de plusieurs classes de qualité de service) pour le routage des paquets de données permet de mieux repartir la répartition de la charge globale d'un réseau. Une perspective de ces travaux consiste à évaluer l'impact de la méthodologie proposée dans ce chapitre sur la répartition de la charge globale du réseau.

Conclusions et perspectives

Dans ce mémoire de thèse nous nous somme intéressés à la résolution de problèmes de plus courts chemins multi-objectifs en optimisant une fonction d'utilité. Ce chapitre rappel les différentes contributions et perspectives de cette thèse.

Contributions concernant le calcul d'une somme pondérée bornant la fonction d'utilité

Les premières contributions concernent le problème de plus court chemin d'un nœud source à un nœud puits avec des objectifs additifs. Ce problème est résolu à l'aide d'un algorithme d'étiquetage augmenté d'une règle de coupe. La règle de coupe permet, grâce à une somme pondérée bornant inférieurement la fonction d'utilité, de supprimer des labels durant l'exécution de l'algorithme d'étiquetage [GP07] et ainsi réduire les temps de calcul. Le calcul d'une somme pondérée bornant la fonction d'utilité représente une des principales difficultés pour exploiter cette règle de coupe. Dans la littérature plusieurs méthodes ont été proposées pour calculer une somme pondérée bornant une fonction d'utilité composé d'un norme euclidienne [PMRS03], d'une fonction OWA [GS07] ou d'une intégrale de Choquet convexe [GP07]. De telles fonctions d'utilité sont trop restrictives pour la modélisation des préférences dans le cadre de l'aide à la décision multi-critère [FGE05]. Nous avons travaillé sur des méthodes pour calculer une somme pondérée bornant inférieurement une fonction d'utilité composée d'une intégrale de Choquet (sans hypothèse restrictive) et de fonctions d'utilité partielles linéaires par morceaux. Les premières méthodes proposées dans cette thèse permettent de calculer une somme pondérée bornant une intégrale de Choquet. Nous avons ensuite proposé des méthodes pour calculer des fonctions linéaires (ou affines) bornant les fonctions d'utilité partielles.

Pour calculer une somme pondérée bornant une intégrale de Choquet quelconque nous proposons de résoudre un programme linéaire dont les contraintes linéaires sont définies par la fonction de capacité de l'intégrale de Choquet. Différentes fonctions objectif pour ce programme linéaire sont proposées et comparées. L'utilisation de ces sommes pondérées dans la règle de coupe permet diminuer les temps de calcul d'un algorithme d'étiquetage de manière significative. Selon la fonction objectif utilisée dans le programme linéaire calculant la somme pondérée, les temps de calcul de l'algorithme d'étiquetage peuvent varier considérablement. Les expérimentations ont permis d'identifier une fonction objectif qui définit une borne efficace dans l'ensemble des cas de l'intégrale de Choquet.

Les temps de calcul de l'algorithme d'étiquetage montrent que l'utilisation de cette méthode n'est pas toujours efficace quand l'intégrale de Choquet n'est pas convexe. Pour cette raison nous proposons une deuxième méthode pour calculer une somme pondérée bornant inférieurement une intégrale de Choquet. Elle repose sur la résolution d'un programme linéaire dont les contraintes linéaires sont définies par la fonction de capacité, une borne supérieure et une borne inférieure sur les objectifs. Les fonctions objectif du programme linéaire précédent peuvent aussi être utilisées pour ce programme linéaire. Cette deuxième méthode permet de diminuer significativement les temps de calcul de l'algorithme d'étiquetage.

Nous avons de plus proposé deux méthodes pour prendre en compte des fonctions d'utilité partielles dans la somme pondérée bornant la fonction d'utilité. À notre connaissance, aucune méthode n'était disponible dans la littérature pour résoudre ce problème. Nous proposons de construire pour chaque objectif une fonction linéaire (ou affine) bornant inférieurement la fonction d'utilité partielle associée à cet objectif. Ces fonctions linéaires (ou affines) sont ensuite intégrées à la somme pondérée bornant l'intégrale de Choquet pour donner une nouvelle somme pondérée (augmentée d'une valeur fixe) bornant inférieurement la fonction d'utilité.

Perspectives

Suite à ces travaux on présente deux perspectives. Avec deux objectifs, Sauvanet et Néron [SN10] proposent de calculer et d'utiliser dans la règle de coupe deux fonctions d'évaluation (à la place d'une seule), ces dernières définissent alors un ensemble bornant les coûts de l'ensemble des chemins. Il est alors sans doute possible de généraliser ces travaux pour calculer et utiliser un ensemble bornant (cf. Définition 3.4, page 78) défini par deux fonctions d'évaluation ou plus avec plus de deux objectifs. On note que Ehrgott et Gandibleux [EG01] proposent une méthode pour construire un ensemble similaire. L'utilisation de plusieurs sommes pondérées différentes bornant la fonction d'utilité, constitue une autre perspective. Dans ce cas, il faudrait étudier combien de sommes pondérées à utiliser simultanément et comment déterminer ces sommes pondérées différentes.

Contributions concernant la dominance selon la fonction d'utilité

Dans un deuxième temps, nous avons proposé des contributions concernant le problème de plus courts chemins d'un nœud source à tous les nœuds du graphe avec des objectifs additifs, bottleneck ou multiplicatifs. Pour résoudre ce problème, seule l'utilisation d'une règle de dominance basée sur la dominance de Pareto était proposée dans la littérature pour supprimer des labels dans un algorithme d'étiquetage. Nous avons travaillé sur la redéfinition de la dominance de Pareto dans le but de prendre en compte une fonction d'utilité. Nous proposons le concept de dominance selon une fonction d'utilité. Dans le cadre de l'optimisation avec des objectifs additifs, nous avons défini une condition suffisante de la dominance selon une fonction d'utilité basée sur une intégrale de Choquet. Cette condition suffisante peut être utilisée de la même manière que la dominance de Pareto dans une règle de dominance pour supprimer des labels dans un algorithme d'étiquetage. On montre que l'utilisation de cette condition suffisante permet de diminuer les temps de calcul significativement par rapport à l'utilisation de la dominance de Pareto.

Dans le cadre du problème plus général de l'optimisation avec des objectifs additifs, bottleneck ou multiplicatifs, nous avons défini une condition suffisante de la dominance selon une fonction d'utilité composée d'une intégrale de Choquet et de fonctions d'utilité partielles linéaires par morceaux. Cette condition suffisante, qui est une généralisation de la précédente, permet de prendre en compte efficacement des objectifs bottleneck.

Perspectives

Plusieurs perspectives à ces contributions concernant la dominance selon une fonction d'utilité existent. En particuliers les deux conditions suffisantes proposées pour vérifier la dominance selon une fonction d'utilité ne sont pas suffisantes et nécessaires pour vérifier cette dominance. Un point important

concerne la prise en compte de fonctions d'utilité partielles, élément clé du modèle multi-critère, où l'approche gagnerait à être approfondie. Une autre perspective concerne l'utilisation de la dominance selon une fonction d'utilité pour résoudre d'autres problèmes d'optimisation combinatoire multi-objectif. Des problèmes multi-objectif, autres que celui de plus court chemin, peuvent être résolus avec des algorithmes d'étiquetage (ou des algorithmes de programmation dynamique), notamment le problème de sac à dos multi-objectif [CCF+03, Jor10]. Dans ce cas il est sans doute possible d'utiliser la dominance selon une fonction d'utilité, reste à savoir si son utilisation est intéressante pour diminuer les temps de calcul. Pour d'autres types de méthodes exactes, telles que les algorithmes de séparation et évaluation (B&B), où la dominance de Pareto est utilisée pour éviter l'explosion combinatoire, la dominance selon une fonction d'utilité pourrait y être utilisée à la place de la dominance de Pareto. La dominance de Pareto est aussi utilisée dans des métaheuristiques. L'utilisation de la dominance selon une fonction d'utilité, à la place de la dominance de Pareto, pourrait permettre d'orienter l'algorithme vers les solutions optimales selon cette fonction d'utilité.

Contribution concernant le routage multi-objectif

Dans le cadre du routage prenant en compte plusieurs objectifs de qualité de service, de nombreuses méthodes ont été développées pour prendre en compte des contraintes sur les objectifs dans des algorithmes d'étiquetage [KMKK02]. La plupart de ces méthodes [Jaf84, NM99] proposent d'optimiser une fonction d'agrégation afin de construire une solution satisfaisant l'ensemble des contraintes. On peut considérer que les contraintes sur les objectifs correspondent à un modèle (simpliste) des préférences de l'administrateur. Plutôt que de définir des contraintes, nous proposons de définir une fonction d'utilité composée d'une intégrale de Choquet et de fonctions d'utilité partielles linéaires par morceaux pour modéliser les préférences de l'administrateur. Nous montrons que cette fonction d'utilité permet de modéliser des contraintes sur certains objectifs. Elle permet aussi de modéliser une pondération, une complémentarité et une substituabilité entre les objectifs [GL10], ainsi cette fonction d'utilité généralise [GP99] les fonctions d'agrégation habituellement utilisées dans le cadre du routage par contraintes. Afin optimiser cette fonction d'utilité, nous proposons d'utiliser un algorithme d'étiquetage avec la dominance selon cette fonction d'utilité. Dans le cadre de routage multi-objectif, il est parfois nécessaire de modéliser la bande passante des liens du réseau avec un objectif bottleneck. Nous montrons que la méthode proposée est efficace pour prendre en compte un objectif bottleneck.

Perspectives

Suite à ce travail de nombreuses perspectives existent. Dans une optique de *traffic engineering*, une perspective consiste à étudier l'impact de la prise en compte de plusieurs objectifs (pour router les paquets) sur la répartition de la charge du réseau sur les liens, notamment quand les objectifs sont agrégés à l'aide d'une fonction d'utilité. Une autre perspective serait de proposer une méthode pour la modélisation des préférences de l'administrateur, sous la forme d'une fonction d'utilité, dans le cadre du routage prenant en compte plusieurs objectifs de qualité de service. Pour ce travail il est possible de s'appuyer sur les méthodes existantes d'aide à la décision multi-critère [GKM08]. Deux problèmes liés à la prise en compte de plusieurs objectifs dans le cadre du routage distribué ont été identifiés [Beu06, MNK01] : le bouclage de paquets dans le réseau et l'utilisation de chemins non optimaux par les paquets. Une preuve identique à celle proposée par Van Mieghem et al [MNK01] peut être faite pour démontrer qu'il ne peut y avoir de bouclage de paquets si tous les routeurs utilisent exactement la même fonction d'utilité (contrairement à la méthode RMC). Concernant l'utilisation de chemins non optimaux (ou ne vérifiant pas les

contraintes) il n'existe pas de solution simple à ce problème si la fonction d'utilité n'est pas monotone (cf. §3.1.1, page 75), ce qui n'est pas le cas pour tous les algorithmes présentés et proposées à l'exception de l'algorithme de Jaffe [Jaf84]. Une perspective de ce travail serait de proposer dans le cadre du routage avec une fonction d'utilité des méthodes permettant d'éviter que les paquets empruntent des chemins non optimaux. Différentes pistes à ce sujet existent dans la littérature [Beu06, MNK01].

Bibliographie

[AM91] J. Azevedo and E. Martins. An algorithm for the multiobjective shortest path problem on acyclic networks. *Investigacao Operacional*, 11 :52–69, 1991.

[AN79] Y.P. Aneja and K.P.K. Nair. Bicriteria transportation problem. *Management Science*, 25 :73–78, 1979.

[ASG07] I. Alaya, C. Solnon, and K. Ghédira. Ant colony optimization for multi-objective optimization problems. In *Proceedings of ICTAI'07, International Conference on Tools with Artificial Intelligence*, volume 1, pages 450–457. IEEE Computer Society Press, 2007.

[BD04] J. Branke and K. Deb. Integrating user preferences into evolutionary multi-objective optimization. In *Knowledge Incorporation in Evolutionary Computation*, pages 461–478. Springer, 2004.

[BDMS08] J. Branke, K. Deb, K. Miettinen, and R. Slowinski. *Multiobjective Optimization : Interactive and Evolutionary Approaches*, volume 5252 of *Lecture Notes in Computer Science*. Springer, 2008. 470 p.

[BdMTL71] R. Benayoun, J. de Montgolfier, J. Tergny, and O.I. Larichev. Linear programming with multiple objective functions : Step method (STEM). *Mathematical Programming*, 1(3) :366–375, 1971.

[BEH90] O. Berman, D. Einav, and G. Handler. The constrained bottleneck problem in networks. *Operations Research*, 38 :178–181, 1990.

[Bel57] R.E Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[Bel58] R.E. Bellman. On a routing problem. *Quarterly Applied Mathematics*, 16 :87–90, 1958.

[Ber93] D.P. Bertsekas. A simple and fast label correcting algorithm for shortest paths. *Networks*, 23(8) :703–709, 1993.

[Beu06] F. Beugnies. Rapport final RMC2. Technical report, LAMIH Université de Valenciennes, France, février 2006.

[BGSZ09] J. Branke, S. Greco, R. Slowinski, and P. Zielniewicz. Interactive evolutionary multiobjective optimization using robust ordinal regression. In M. Ehrgott, C.M. Fonseca, X. Gandibleux, J.K. Hao, and M. Sevaux, editors, *Proceedings of 5th International Conference on Evolutionary Multi-Criterion Optimization*, volume 5467 of *Lecture Notes in Computer Science*, pages 554–568. Springer, 2009.

[BP05] D. Bouyssou and M. Pirlot. Conjoint measurement tools for MCDM. In J.R. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis : State of the Art Surveys*, pages 73–130. Springer, 2005.

[BSS89] J. Brumbaugh-Smith and D. Shier. An empirical investigation of some bicriterion shortest path algorithms. *European Journal of Operational Research*, 43(2) :216–224, 1989.

[BT91] D.P. Bertsekas and J.N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16 :580–595, 1991.

[BV85] J.P. Brans and P. Vincke. A preference ranking organisation method : The PROMETHEE method for MCDM. *Management Science*, 31(6) :647–656, 1985.

[CAA93] J.N. Clímaco, C.H. Antunes, and M.J. Alves. Interactive decision support for multiobjective transportation problems. *European Journal of Operational Research*, 65(1) :58–67, 1993.

[CCF+03] M.E. Captivo, J. Clímaco, J. Figueira, E. Martins, and J.L. Santos. Solving bicriteria 0-1 knapsack problems using a labeling algorithm. *Computers & Operations Research*, 30(12) :1865–1886, 2003.

[CCP03] J.C.N. Climaco, J.M.F. Craveirinha, and M. Pascoal. A bicriterion approach for routing problems in multimedia networks. *Networks*, 41(4) :206–220, 2003.

[CCP06] J. Clímaco, J. Craveirinha, and M. Pascoal. An automated reference point-like approach for multicriteria shortest path problems. *Journal of Systems Science and Systems Engineering*, 15(3) :314–329, 2006.

[Cho53] G. Choquet. Theory of capacities. *Annales de l'Institut Fourier*, 5 :131–295, 1953.

[CM81] J.C.N. Climaco and E.Q.V. Martins. On the determination of the nondominated paths in multi-objective network problem. *Methods in Operations Research*, 40 :255–258, 1981.

[CM82] J.C.N. Climaco and E.Q.V. Martins. A bicriterion shortest path problem. *European Journal of Operational Research*, 11 :399–404, 1982.

[CM85] H. Corley and I. Moon. Shortest paths in networks with vector weights. *Journal of Optimization Theory and Application*, 46 :79–86, 1985.

[CM86] J. Current and H.K. Min. Multiobjective design of transportation networks : Taxonomy and annotation. *European Journal of Operational Research*, 26(2) :187 – 201, 1986.

[CM93] J. Current and M. Marsh. Multiobjective transportation network design and routing problems : Taxonomy and annotation. *European Journal of Operational Research*, 65(1) :4–19, 1993.

[CMM90] R.L. Carraway, T.L. Morin, and H. Moskowitz. Generalized dynamic programming for multicriteria optimization. *European Journal of Operational Research*, 44 :95–104, 1990.

[CMT11] K. Chiba, Y. Makino, and T. Takatoya. Design-informatics approach applicable to real-world problem. In *Proceedings of 2011 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM)*, pages 167–174, Paris, France, april 2011.

[CP10] J.C.N. Climaco and M. Pascoal. Multicriteria path and tree problems–discussion on exact algorithms. In *Proceedings of ALIO-INFORMS International Meeting*, Buenos Aires, may 2010.

[CRC90] J.R. Current, C.S. Revelle, and J.L. Cohon. An interactive approach to identify the best compromise solution for two objective shortest path problems. *Computers & Operations Research*, 17(2) :187–198, 1990.

[DCD95] P. Dasgupta, P.P. Chakrabarti, and S.C. DeSarkar. Utility of pathmax in partial order heuristic search. *Information Processing Letters*, 55(6) :317–322, 1995.

[DD80] H.G. Daellenbach and C.A. DeKluyver. Note on multiple objective dynamic programming. *Journal of the Operational Research Society*, 31 :591—594, 1980.

[Deb01] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. J. Wiley, 2001. 518 p.

[DG02] F. Degoutin and X. Gandibleux. Un retour d'expérience sur la résolution de problèmes combinatoires bi-objectifs. In *5e journée du groupe de travail Programmation Mathéma-tique MultiObjectif (PM2O)*. Angers, France, May 2002.

[DGN+09] J. Durillo, J. Garca, A. Nebro, C.A. Coello-Coello, F. Luna, and E. Alba. Multi-objective particle swarm optimizers : An experimental comparison. In *Proceedings of 5th International Conference on Evolutionary MultiCriterion Optimization*, page 495–509, 2009.

[Dij59] E.W Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[DP84] N. Deo and C.Y. Pang. Shortest-path algorithms : Taxonomy and annotation. *Networks*, 4(2) :275–323, 1984.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm : NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197, 4 2002.

[DS10] C. Delort and O. Spanjaard. Using bound sets in multiobjective optimization : Application to the biobjective binary knapsack problem. In *Proceedings of 9th International Symposium on Experimental Algorithms (SEA'10)*, volume 6049 of *Lecture Notes in Computer Science*, pages 253–265, 2010.

[DW09] D. Delling and D. Wagner. Pareto paths with SHARC. In J. Vahrenhold, editor, *Experimental Algorithms*, volume 5526 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 2009.

[Dye05] J.S. Dyer. MAUT - multiattribute utility theory. In J.R. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis : State of the Art Surveys*, pages 265–295. Springer, 2005.

[eCCV05] C.A. Bana e Costa, J.M. De Corte, and J.C. Vansnick. On the mathematical foundation of macbeth. In J.R. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis : State of the Art Surveys*, pages 387–500. Springer, 2005.

[eCV94] C.A. Bana e Costa and J.C. Vansnick. A theoretical framework for measuring attractiveness by a categorical based evaluation technique (MACBETH). In *Proceedings of XIth International Conference on MCDM*, pages 15–24, Coimbra, Portugal, 1994.

[EG00] M. Ehrgott and X. Gandibleux. A survey and annoted bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22 :425–460, 2000.

[EG01] M. Ehrgott and X. Gandibleux. Bounds and bound sets for biobjective combinatorial optimization problems. In M. Koksalan and S. Ziont, editors, *Multiple Criteria Decision Making in the New Millennium*, volume 507 of *Lectures Notes in Economics and Mathematical Systems*, pages 241–253. Springer, 2001.

[EG02] M. Ehrgott and X. Gandibleux. *Multiple Criteria Optimization. State of the art annotated bibliographic surveys*, volume 52. Kluwer Academic, 2002. 520 p.

[EG04] M. Ehrgott and X. Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *TOP (Spanish journal of operations research)*, 12(1) :1–63, May 2004.

[EG07] M. Ehrgott and X. Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Computers & Operations Research*, 34(9) :2674–2694, 2007.

[Ehr00a] M. Ehrgott. Approximation algorithms for combinatorial multicriteria optimization problems. *International Transactions in Operational Research*, 7(1) :5–31, 2000.

[Ehr00b] M. Ehrgott. *Multicriteria optimization*, volume 491 of *Lecture Notes in Economics and Mathematical Systems*. Springer, 2000. 323 p.

[Epp94] D. Eppstein. Finding the k shortest paths. In *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pages 154–165. 1994.

[ETP03] M. Ehrgott and D. Tenfelde-Podehl. Computation of ideal and nadir values and implications for their use in MCDM methods. *European Journal of Operational Research*, 151(1) :119–139, 2003.

[Eul36] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Mémoires de l'Académie des sciences de Berlin*, 1736.

[Eva84] G. Evans. An overview of techniques for solving multiobjective mathematical problems. *Management Science*, 30(11) :1268–1282, 1984.

[FGE05] J.R. Figueira, S. Greco, and M. Ehrgott. *Multiple Criteria Decision Analysis : State of the Art Surveys*. Springer, 2005. 1045 p.

[FGL11a] H. Fouchal, X. Gandibleux, and F. Lehuédé. A Lower Bound of the Choquet Integral Integrated Within Martins' Algorithm. In Y. Shi, S. Wang, G. Kou, and J. Wallenius, editors, *New State of MCDM in the 21st Century, Selected Papers of the 20th International Conference on Multiple Criteria Decision Making 2009*, volume 648 of *Lecture Notes in Economics and Mathematical Systems*, pages 79–90. 2011.

[FGL11b] H. Fouchal, X. Gandibleux, and F. Lehuédé. Preferred solutions computed with a label setting algorithm based on Choquet integral for multi-objective shortest paths. In *Proceedings of 2011 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM)*, pages 143–150, Paris, France, april 2011.

[FGMS08] J.R. Figueira, S. Greco, V. Mousseau, and R. Slowinski. Interactive multiobjective optimization using a set of additive value functions. In J. Branke, K. Deb, K. Miettinen, and R. Slowinski, editors, *Multiobjective Optimization : Interactive and Evolutionary Approaches*, volume 5252 of *Lecture Notes in Computer Science*, pages 99–122. Springer, 2008.

[FGRS10] J.R. Figueira, S. Greco, B. Roy, and R. Slowinski. Electre methods : Main features and recent developments. In P. Pardalos, D. Hearn, and C. Zopounidis, editors, *Handbook of Multicriteria Analysis*, volume 103 of *Applied Optimization*, pages 51–89. Springer, 2010.

[FGS09] J.R. Figueira, S. Greco, and R. Slowinski. Building a set of additive value functions representing a reference preorder and intensities of preference : GRIP method. *European Journal of Operational Research*, 195(2) :460–486, 2009.

[FLB+10] E. Fernandez, E. Lopez, S. Bernal, C.A. Coello-Coello, and J. Navarro. Evolutionary multiobjective optimization using an outranking-based dominance generalization. *Computers & Operations Research*, 37(2) :390–395, 2010.

[FMR05] J.R. Figueira, V. Mousseau, and B. Roy. ELECTRE methods. In J.R. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis : State of the Art Surveys*, pages 133–162. Springer, 2005.

[FP00] M. Futtersack and P. Perny. BCA*, une généralisation d'A* pour la recherche de solutions de compromis dans des problèmes de recherche multiobjectifs. In *Compte rendu de la 12ème conference Reconnaissance des Formes et Intelligence Artificielle*, volume 3, pages 377–386, 2000.

[FRT02] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 40 :118–124, 2002.

[FT02] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4) :756–767, 2002.

[GBR06] X. Gandibleux, F. Beugnies, and S. Randriamasy. Martins' algorithm revisited for multi-objective shortest path problems with a maxmin cost function. *4OR : A Quarterly Journal of Operations Research*, 4(1) :47–59, March 2006.

[Geo67] A.M. Geoffrion. Solving bicriterion mathematical programs. *Operations Research*, 15(1) :39–54, 1967.

[Geo68] A.M. Geoffrion. Proper efficiency and the theory of vector optimisation. *Journal of Mathematical Analysis and Application*, 22 :618–630, 1968.

[GG03] J. Granat and F. Guerriero. The interactive analysis of the multicriteria shortest path problem by the reference point method. *European Journal of Operational Research*, 151(1) :103–118, 2003.

[GKM08] M. Grabisch, I. Kojadinovic, and P. Meyer. A review of capacity identification methods for Choquet integral based multi-attribute utility theory — applications of the kappalab R package. *European Journal of Operational Research*, 186(2) :766–785, 2008.

[GL02] M. Grabisch and C. Labreuche. Bi-capacities for decision making on bipolar scales. In *Proceedings of the EUROFUSE 02 Workshop on Information Systems*, pages 185–190, 2002.

[GL05] M. Grabisch and C. Labreuche. Fuzzy measures and integrals in MCDA. In J.R. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis : State of the Art Surveys*, pages 563–608. Springer, 2005.

[GL10] M. Grabisch and C. Labreuche. A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 175(1) :247–286, 2010.

[GLP11] L. Galand, J. Lesca, and P. Perny. Multiobjective dynamic programming versus linear programming for compromise search with choquet integral. In *International Conference on Multiple Criteria Decision Making*, page 175, Jyvaskyla, Finland, June 2011.

[GM01] F. Guerriero and R. Musmanno. Label correcting methods to solve multicriteria shortest path problems. *Journal of Optimization Theory and Applications*, 111(3) :589–613, 2001.

[GM08] M. Gondran and M. Minoux. *Graphs, Dioids and Semirings : New Models and Algorithms*. Operations Research/Computer Science Interfaces Series. Springer, 2008. 381 p.

[GMF97] X. Gandibleux, N. Mezdaoui, and A. Freville. A tabu search procedure to solve multiobjective combinatorial optimization problems. In R. Caballero, F. Ruiz, and R. Steuer, editors, *Advances in Multiple Objective and Goal Programming*, volume 455 of *Lecture Notes in Economics and Mathematical Systems*, pages 291—300. 1997.

[GMSK00] M. Grabisch, T. Murofushi, M. Sugeno, and J. Kacprzyk. *Fuzzy Measures and Integrals : Theory and Applications*. Physica Verlag, 2000.

[GN10] K. Ghoseiri and B. Nadjari. An ant colony optimization algorithm for the bi-objective shortest path problem. *Applied Soft Computing*, 10(4) :1237–1246, 2010.

[GP86] G. Gallo and S. Pallottino. Shortest path methods : A unifying approach. In G. Gallo and C. Sandi, editors, *Netflow at Pisa*, volume 26 of *Mathematical Programming Studies*, pages 38–64. Springer, 1986.

[GP99] M. Grabisch and P. Perny. Agrégation multicritère. In *Utilisations de la logique floue*, pages 81–120. Hermès, 1999.

[GP06a] L. Galand and P. Perny. Interactive search for compromise solutions in multicriteria graph problems. In *Proceedings of 9th IFAC Symposium on Automated Systems Based on Human Skill And Knowledge*, Nancy, may 2006. Elsevier.

[GP06b] L. Galand and P. Perny. Search for compromise solutions in multiobjective state space graphs. In *Proceedings of 17th European Conference on Artificial Intelligence*, pages 93–97, 2006.

[GP07] L. Galand and P. Perny. Search for Choquet-optimal paths under uncertainty. In *Proceedings of 23rd conference on Uncertainty in Artificial Intelligence*, pages 125–132, Vancouver, July 2007. AAAI Press.

[GP10] B. Golden and P. Perny. Infinite order lorenz dominance for fair multiagent optimization. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, volume 1 of *AAMAS '10*, pages 383–390, 2010.

[GPS09] L. Galand, P. Perny, and O. Spanjaard. A branch and bound algorithm for choquet optimization in multicriteria problems. In M. Ehrgott, B. Naujoks, T.J. Stewart, and J. Wallenius, editors, *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, volume 634 of *Lecture Notes in Economics and Mathematical Systems*, pages 355–365. Springer, 2009.

[GPS10] L. Galand, P. Perny, and O. Spanjaard. Choquet-based optimisation in multiobjective shortest path and spanning tree problems. *European Journal of Operational Research*, 204(2) :303–315, 2010.

[GR00] M. Grabisch and M. Roubens. Application of the Choquet integral in multicriteria decision making. In M. Grabisch, T. Murofushi, and M. Sugeno, editors, *Fuzzy Measures and Integrals - Theory and Applications*, pages 348–374. Physica Verlag, 2000.

[Gra95] M. Grabisch. Fuzzy integral in multicriteria decision making. *Fuzzy Sets and Systems*, 69 :279–298, 1995.

[Gra96] M. Grabisch. The application of fuzzy integrals in multicriteria decision making. *European Journal of Operational Research*, 89 :445–456, 1996.

[Gra97a] M. Grabisch. Alternative representations of discrete fuzzy measures for decision making. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 5(5) :587–607, 1997.

[Gra97b] M. Grabisch. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems*, 92(2) :167–189, 1997.

[Gra10] T. Grabener. *Calcul d'itinéraire multimodal et multiobjectif en milieu urbain*. PhD thesis, Université de Toulouse, 2010.

[GS07] L. Galand and O. Spanjaard. OWA-based search in state space graphs with multiple cost functions. In *Proceedings of 20th International Florida Artificial Intelligence Research Society Conference*, pages 86–91, Key West, may 2007. AAAI Press.

[GSSD08] R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction hierarchies : Faster and simpler hierarchical routing in road networks. In C. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 319–333. Springer, 2008.

[GV02] V. Gabrel and D. Vanderpooten. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite. *European Journal of Operational Research*, 139(3) :533—-542, 2002.

[GVT98] X. Gandibleux, D. Vancoppenolle, and D. Tuyttens. A first making use of GRASP for solving MOCO problems. Technical report, University of Valenciennes, France, 1998. presented at MCDM 14, Charlottesville, VA.

[Han80] P. Hansen. Bicriterion path problems. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making : Theory and Application*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 109–127. Springer, 1980.

[Hen86] M. Henig. The shortest path problem with two objective functions. *European Journal of Operational Research*, 25(2) :281–291, 1986.

[HHW01] C. Hallam, K.J. Harrison, and J.A. Ward. A multiobjective optimal path algorithm. *Digital Signal Processing*, 11(2) :133–143, 2001.

[HK96] S. Harikumar and S. Kumar. Iterative Deepening Multiobjective A*. *Information Processing Letters*, 58(1) :11–15, 1996.

[HLW71] Y. Haimes, L. Lasdon, and D. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cyberbetics*, 1 :296—297, 1971.

[HNR68] P.E. Hart, N.J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2) :100–107, 1968.

[HPS96] F. Huarng, S. Pulat, and L.H. Shih. A computational comparison of some bicriterion shortest path algorithms. *Journal of the Chinese Institute of Industrial Engineers*, 13(2) :121–125, 1996.

[HQF07] F.G. He, H. Qi, and Q. Fan. An evolutionary algorithm for the multi-objective shortest path problem. In *Proceedings of ISKE'07, International Conference on Intelligent Systems and Knowledge Engineering*, October 2007.

[Ign76] J.P. Ignizio. *Goal Programming and Its Extensions*. D.C. Heath, Lexington, MA, 1976.

[IMP10] M. Iori, S. Martello, and D. Pretolani. An aggregate label setting policy for the multi-objective shortest path problem. *European Journal of Operational Research*, 207(3) :1489–1496, 2010.

[Jaf84] J.M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14(1) :95–116, 1984.

[JLMS87] E. Jacquet-Lagrèze, R. Meziani, and R. Slowinski. MOLP with an interactive assessment of a piecewise-linear utility function. *European Journal of Operational Research*, 31(3) :350–357, 1987.

[JLS82] E. Jacquet-Lagrèze and Y. Siskos. Assessing a set of additive utility functions for multicriteria decision making : the UTA method. *European Journal of Operational Research*, 10 :151–164, 1982.

[JMT02] D. F. Jones, S. K. Mirrazavi, and M. Tamiz. Multi-objective meta-heuristics : An overview of the current state-of-the-art. *European Journal of Operational Research*, 137(1) :1–9, 2002.

[Jor10] J. Jorge. *Nouvelles propositions pour la résolution exacte du sac à dos multi-objectif unidimensionnel en variables binaires*. Thèse de doctorat, Université de Nantes, 2010.

[Jun04] U. Junker. Preference-based search and multi-criteria optimization. *Annals of Operations Research*, 130(1) :75–115, 2004.

[Kee96] R.L. Keeney. *Value-focused thinking : A path to creative decisionmaking*. Harvard university press, 1996. 432 p.

[KKKM04] F. Kuipers, T. Korkmaz, M. Krunz, and P. Van Mieghem. Performance evaluation of constraint-based path selection algorithms. *IEEE Network*, 18 :16–23, 2004.

[KMKK02] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz. An overview of constraint-based path selection algorithms for QoS routing. *IEEE Communications Magazine*, 40(12) :50–55, 2002.

[KR76] R.L. Keeney and H. Raiffa. *Decisions with multiple objectives : Preferences and value tradeoffs*. J. Wiley, New York, 1976. 592 p.

[KY97] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*, volume 14 of *Nonconvex Optimization and its Applications*. Kluwer Academic, 1997. 356 p.

[Leh03] F. Lehuédé. *Intégration d'un modèle d'aide à la décision multicritère en programmation par contraintes*. Thèse de doctorat, Université Paris VI, 2003.

[LG03] C. Labreuche and M. Grabisch. The Choquet integral for the aggregation of interval scales in multicriteria decision making. *Fuzzy Sets and Systems*, 137(1) :11–26, July 2003.

[LGLS06a] F. Lehuédé, M. Grabisch, C. Labreuche, and P. Savéant. Integration and propagation of a multi-criteria decision making model in constraint programming. *Journal of Heuristics*, 12(4-5) :329–346, September 2006.

[LGLS06b] F. Lehuédé, M. Grabisch, C. Labreuche, and P. Savéant. MCS - a new algorithm for multicriteria optimisation in constraint programming. *Annals of Operations Research*, 147(1) :143–174, October 2006.

[LL05] C. Labreuche and F. Lehuédé. MYRIAD : a tool suite for MCDA. In *Proceedings of EUSFLAT'05*, pages 204–209, Barcelona, september 2005.

[LL06] F. Lehuédé and C. Labreuche. Inconsistencies in the determination of a capacity. In *Proceedings of ECAI 2006 Multidisciplinary Workshop on Advances in Preference Handling*, Riva del Garda, Italy, 2006.

[LR01] G. Liu and K.G. Ramakrishnan. A* Prune : an algorithm for finding k shortest paths subject to multiple constraints. In *Proceedings of INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 743–749. IEEE, 2001.

[Mar84a] E.Q.V. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2) :236–245, 1984.

[Mar84b] E.Q.V. Martins. On a special class of bicriterion path problems. *European Journal of Operational Research*, 17 :85–94, 1984.

[Mar98] J.-L. Marichal. *Aggregation Operators for Multicriteria Decision Aid*. PhD thesis, Institute of Mathematics, University of Liège, Liège, Belgium, 1998.

[MdlC03] L. Mandow and J.L. Pérez de-la Cruz. Multicriteria heuristic search. *European Journal of Operational Research*, 150(2) :253–280, 2003.

[MdlC05a] L. Mandow and J.L Pérez de-la Cruz. Comparison of heuristics in multiobjective A* search. In R. Marín, E. Onaindia, A. Bugarín, and J. Santos, editors, *Proceedings of CAEPIA'05*, volume 4177 of *Lecture Notes in Computer Science*, pages 180–189. Springer, 2005.

[MdlC05b] L. Mandow and J.L Pérez de-la Cruz. A new approach to multiobjective A* search. In *Proceedings of IJCAI'05*, pages 218–223. Professional Book Center, 2005.

[MG99] P. Miranda and M. Grabisch. Optimization issues for fuzzy measures. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 7(6) :545–560, 1999.

[MH92] I. Murthy and S.S. Her. Solving min-max shortest-path problems on a network. *Naval Research Logistics*, 39 :669–683, 1992.

[MHS07] M. Müller-Hannemann and M. Schnee. Finding all attractive train connections by multicriteria pareto search. In F. Geraets, L. Kroon, A. Schoebel, D. Wagner, and C. Zaroliagis, editors, *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*, pages 246–263. Springer, 2007.

[MHW06] M. Müller-Hannemann and K. Weihe. On the cardinality of the pareto set in bicriteria shortest path problems. *Annals of Operations Research*, 147(1) :269–286, 2006.

[MKK+03] P. Van Mieghem, F. Kuipers, T. Korkmaz, M. Krunz, M. Curado, E. Monteiro, X. Masip-Bruin, J. Solé-Pareta, and S. Sánchez-López. Quality of service routing. In *Quality of Future Internet Services*, volume 2856 of *Lecture Notes in Computer Science*, pages 80–117, 2003.

[MMdlCRS10] E. Machuca, L. Mandow, J.L. Pérez de-la Cruz, and A. Ruiz-Sepulveda. An empirical comparison of some multiobjective graph search algorithms. In R. Dillmann, J. Beyerer, U. Hanebeck, and T. Schultz, editors, *KI 2010 : Advances in Artificial Intelligence*, volume 6359 of *Lecture Notes in Computer Science*, pages 238–245. Springer, 2010.

[MMO91] J. Mote, I. Murthy, and D.L. Olson. A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53 :81–92, 1991.

[MNK01] P. Van Mieghem, H. De Neve, and F. Kuipers. Hop-by-hop quality of service routing. *Computer Networks*, 37(3-4) :407–423, 2001.

[MPJ07] J. Maria, A. Pangiliana, and G. Janssens. Evolutionary algorithms for the multiobjective shortest path planning problem. *International journal of computer and information science and engineering*, 1(1) :54—59, 2007.

[MPRS07] E.Q.V. Martins, J.M. Paixao, M.S. Rosa, and J.L. Santos. Ranking multiobjective shortest paths. Technical Report 07-11, Center for Mathematics, University of Coimbra, 2007.

[MPS99] E.Q. Martins, M.M. Pascoal, and J.L. Santos. Deviation algorithms for ranking shortest paths. *International Journal of Foundations of Computer Science*, 10(3) :247–261, 1999.

[MS00] E.Q.V. Martins and J.L. Santos. The labelling algorithm for the multiobjective shortest path problem. Technical report, Center for Mathematics, University of Coimbra, 2000.

[MT90] S. Martello and P. Toth. *Knapsack problems : algorithms and computer implementations*. J. Wiley, 1990.

[Mun04] G. Munda. Social multi-criteria evaluation : Methodological foundations and operational consequences. *European Journal of Operational Research*, 158(3) :662–677, 2004.

[NM99] H. De Neve and P. Van Mieghem. TAMCRA : a tunable accuracy multiple constraints routing algorithm. *Computer Communications*, 23(7) :667–679, 1999.

[NW99] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*. Interscience Series in Discrete Mathematics and Optimization. Wiley, 1999. 763 p.

[OS06] S. Omkarprasad and K. Sushil. Analytic hierarchy process : An overview of applications. *European Journal of Operational Research*, 169(1) :1–29, 2006.

[Par06] V. Pareto. *Manuale di Economia Politica*. Piccola Biblioteca Scientifica, Milan, 1906. Translated into English by Ann S. Schwier (1971), Manual of Political Economy, MacMillan, London.

[PBM09] L.L. Pinto, C.T. Bornstein, and N. Maculan. The tricriterion shortest path problem with at least two bottleneck objective functions. *European Journal of Operational Research*, 198(2) :387–391, 2009.

[PBR00] J.C. Pomerol and S. Barba-Romero. *Multicriterion decision in management : principles and practice*. Kluwer Academic, 2000. 395 p.

[PF98] B. Pelegrin and P. Fernandez. On the sum-max bicriterion path problem. *Computers & Operations Research*, 25(12) :1043–1054, 1998.

[PGE08] A. Przybylski, X. Gandibleux, and M. Ehrgott. Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185(2) :509–533, 2008.

[PGE10a] A. Przybylski, X. Gandibleux, and M. Ehrgott. A Recursive Algorithm for Finding All Nondominated Extreme Points in the Outcome Set of a Multiobjective Integer Programme. *INFORMS Journal on Computing*, 22(3) :371–386, 2010.

[PGE10b] A. Przybylski, X. Gandibleux, and M. Ehrgott. A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3) :149–165, 2010.

[PMRS03] J.M. Paixao, E.Q.V Martins, M.S Rosa, and J.L. Santos. The determination of the path with minimum-cost norm value. *Networks*, 41(4) :184–196, 2003.

[PP10] L.L. Pinto and M.B. Pascoal. On algorithms for the tricriteria shortest path problem with two bottleneck objective functions. *Computers & Operations Research*, 37(10) :1774–1779, 2010.

[PPK03] S. Phelps-Pamuk and M. Koksalan. An interactive evolutionary metaheuristic for multiobjective combinatorial optimization. *Management Science*, 49 :1726–1738, 2003.

[PRS07] J.M. Paixao, M.S. Rosa, and J.L. Santos. Labelling methods for the general case of the multiobjective shortest path problem - a computational study. Technical Report 07-42, Center for Mathematics, University of Coimbra, 2007.

[PS05] P. Perny and O. Spanjaard. A preference-based approach to spanning trees and shortest paths problems. *European Journal of Operational Research*, 162(3) :584–601, 2005.

[PS08] J.M. Paixao and J.L. Santos. A new ranking path algorithm for the multiobjective shortest path problem. Technical Report 08-27, Center for Mathematics, University of Coimbra, 2008.

[PV98] P. Perny and D. Vanderpooten. An interactive multiobjective procedure for selecting medium-term countermeasures after nuclear accidents. *Journal of Multi-Criteria Decision Analysis*, 7(1) :48–60, 1998.

[PY00] C.H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web-sources. In *Proceedings of 41st IEEE Symposium on Foundations of Computer Science*, pages 86–92, novembre 2000.

[Rai10] A. Raith. Speed-up of labelling algorithms for biobjective shortest path problems. In *Proceedings of the 45th Annual Conference of the ORSNZ*, november 2010.

[RB93] B. Roy and D. Bouyssou. *Aide Multicritère à la Décision : Méthodes et Cas*. Economica, Paris, 1993.

[RE09] A. Raith and M. Ehrgott. A comparison of solution strategies for biobjective shortest path problems. *Computers & Operations Research*, 36(4) :1299–1331, 2009.

[RG96] R. Ravi and M. Goemans. The constrained minimum spanning tree problem. In R. Karlsson and A. Lingas, editors, *Algorithm Theory — SWAT'96*, volume 1097 of *Lecture Notes in Computer Science*, pages 66–75. Springer, 1996.

[RGFT04] S. Randriamasy, X. Gandibleux, J. Figueira, and P. Thomin. Device and a method for determining routing paths in a communication network in the presence of selection attributes, 2004. Patent 11/25/04. #20040233850. Washington, DC, USA.

[Rot64] G.C. Rota. Theory of möbius functions. In *On the foundations of combinatorial theory*, volume 2 of *Probability Theory and Related Fields*, pages 340–368. 1964.

[Roy59] B. Roy. Transitivité et connexité. *Comptes Rendus de l'Académie des sciences de Paris*, 249 :216–218, 1959.

[Roy68] B. Roy. Classement et choix en présence de points de vue multiples : La méthode ELECTRE. *Revue Francaise d'Informatique et de Recherche Opérationnelle*, 8 :57–75, 1968.

[Roy76] B. Roy. *From optimization to multicriteria decision aid : Three main operational attitudes*, volume 130 of *Lecture Notes in Computer Science*. Springer, 1976.

[Roy85] B. Roy. *Méthodologie multicritère d'aide à la décision*. Economica, Paris, 1985.

[Roy05] B. Roy. Paradigms and challenges. In J.R. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis : State of the Art Surveys*, pages 3–24. Springer, 2005.

[RP11] L.B. Reinhardt and D. Pisinger. Multi-objective and multi-constrained non-additive shortest path problems. *Computers & Operations Research*, 38(3) :605–616, 2011.

[RV03] I. Refanidis and I. Vlahavas. Multiobjective heuristic state-space planning. *Artificial Intelligence*, 145(1-2) :1–32, 2003.

[RW05] S. Ruzika and M.M. Wiecek. Approximation methods in multiobjective programming. *Journal of Optimization Theory and Applications*, 126 :473–501, 2005.

[SA00] A.J.V. Skriver and K.A. Andersen. A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research*, 27 :507–524, 2000.

[Saa77] T.L. Saaty. A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, 15 :234–281, 1977.

[SC83] R.E. Steuer and E. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26(1) :326–344, 1983.

[SC04] J. Sylva and A. Crema. A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European Journal of Operational Research*, 158 :46–55, 2004.

[Ser86] P. Serafini. Some considerations about computational complexity for multiobjective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent Advances and Historical Development of Vector Optimization*, volume 294 of *Lecture notes in Economics and Mathematical Systems*. Springer, 1986.

[SGM05] Y. Siskos, E. Grigoroudis, and F. Matsatsinis. Uta methods. In J.R. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis : State of the Art Surveys*, pages 297–343. Springer, 2005.

[Sha53] L.S. Shapley. A value for n-person games. In H.W. Kuhn and A.W. Tucker, editors, *Contributions to the Theory of Games, Vol. II*, volume 28 of *Annals of Mathematics Studies*, pages 307–317. Princeton University Press, 1953.

[SHBB08] I. Saad, S. Hammadi, M. Benrejeb, and P. Borne. Choquet integral for criteria aggregation in the flexible job-shop scheduling problems. *Mathematics and Computers in Simulation*, 76(5-6) :447–462, 2008.

[SI91] B.S. Stewart and C.C. White III. Multiobjective A*. *Journal of ACM*, 38(4) :775–814, October 1991.

[Skr00] A. Skriver. A classification of bicriterion shortest path (bsp) algorithms. *Asia-Pacific Journal of Operational Research*, 17 :192–212, 2000.

[SN10] G. Sauvanet and E. Néron. Search for the best compromise solution on multiobjective shortest path problem. In *Proceedings of ISCO 2010 - International Symposium on Combinatorial Optimization*, volume 36, pages 615–622, 2010.

[SNB11] G. Sauvanet, E. Néron, and H. Baptiste. Improvements in labeling methods for bi-objective shortest path problem : application to the distance/safety trade-off. 2011. submitted to European Journal of Operational Research.

[SS08] F. Sourd and O. Spanjaard. A multi-objective branch-and-bound framework. application to the bi-objective spanning tree problem. *INFORMS Journal of Computing*, 20(3) :472–484, 2008.

[STW04] J. Scharnow, K. Tinnefeld, and I. Wegener. The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms*, 3 :349–366, 2004.

[Sug74] M. Sugeno. *Theory of fuzzy integrals and its applications*. PhD thesis, Tokyo Institute of Technology, Tokyo, Japan, 1974.

[Sug77] M. Sugeno. Fuzzy measures and fuzzy integrals : a survey. In M.M. Gupta, G.N. Saridis, and B.R. Gains, editors, *Fuzzy automata and decision processes*, pages 89–102. North Holland, 1977.

[Tar07] Z. Tarapata. Selected multicriteria shortest path problems : An analysis of complexity, models and adaptation of standard algorithms. *International Journal of Applied Mathematics and Computer Science*, 17(2) :269–287, 2007.

[TC88] C.T. Tung and K.L. Chew. A bicriterion pareto-optimal path algorithm. *Asia-Pacific Journal of Operational Research*, 5 :166–172, 1988.

[TC92] C.T. Tung and K.L. Chew. A multicriteria pareto-optimal path algorithm. *European Journal of Operational Research*, 62 :203–209, 1992.

[TMKM09] L. Thiele, K. Miettinen, P.J. Korhonen, and J. Molina. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary Computation*, 17(3) :411–436, 2009.

[TP02] D. Tenfelde-Podehl. *Facilities Layout Problems : Polyhedral Structure, Multiple Objectives and Robustness*. PhD thesis, University of Kaiserslautern. Fachbereich Mathematik, 2002. PhD thesis.

[TP03] D. Tenfelde-Podehl. A recursive algorithm for multiobjective combinatorial optimization problems with Q criteria. Technical report, Institut für Mathematik, Technische Universität Graz, 2003.

[TZ09] G. Tsaggouris and C. Zaroliagis. Multiobjective optimization : Improved FPTAS for shortest paths and non-linear objectives with applications. *Theory of Computing Systems*, 45(1) :162–186, 2009.

[UT95] E.L. Ulungu and J. Teghem. The two phases method : An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20(2) :149–165, 1995.

[Van89] D. Vanderpooten. *L'approche interactive dans l'aide multicritère à la décision : Aspects conceptuels, méthodologiques et informatiques*. Thèse de doctorat, Université Paris-Dauphine, 1989.

[VV89] D. Vanderpooten and P. Vincke. Description and analysis of some representative interactive multicriteria procedures. *Mathematical and Computer Modelling*, 12 :1221–1238, 1989.

[War87] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1) :70–79, 1987.

[WC96] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7) :1228–1234, 1996.

[WDF+08] J. Wallenius, J Dyer, P. Fishburn, R. Steuer, S. Zionts, and K. Deb. Multiple Criteria Decision Making, MultiAttribute Utility Theory : Recent accomplishments and what lies ahead. *Management Science*, 54(7) :1336–1349, July 2008.

[Whi82] D. White. The set of efficient solutions for multiple objective shortest path problems. *Computers & Operations Research*, 9, 1982.

[WSC92] C.C. White, B.S. Stewart, and R.L. Carraway. Multiobjective, preference-based search in acyclic or-graphs. *European Journal of Operational Research*, 56(3) :357–363, 1992.

[XN99] X. Xiao and L.N. Ni. Internet QoS : a big picture. *IEEE Network*, 13(2) :8–18, 1999.

[Yag88] R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1) :183–190, 1988.

[YF03] O. Younis and S. Fahmy. Constraint-based routing in the internet : Basic principles and recent research. *IEEE Communications Surveys & Tutorials*, 5(1) :2–13, 2003.

[ZBT07] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited : On the design of pareto-compliant indicators via weighted integration. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 862–876. Springer, 2007.

[ZLT02] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2 : Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In K.C. Giannakoglou et al, editor, *Proceedings of EUROGEN'01, Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems*, pages 95–100, 2002.

Liste des tableaux

Liste des figures

Liste des exemples

Table des matières

Optimisation de l'intégrale de Choquet pour le calcul de plus courts chemins multi-objectifs préférés

Hugo FOUCHAL

Résumé

Dans cette thèse nous nous intéressons à la prise en compte des préférences du décideur pour le calcul des plus courts chemins multi-objectif. Les préférences du décideur sont modélisées a priori par une fonction d'utilité basée sur une intégrale de Choquet. Plutôt que de calculer l'ensemble des solutions efficaces puis de sélectionner une solution optimale selon cette fonction d'utilité, nous cherchons à directement construire une solution efficace optimale au regard des préférences. Le travail réalisé développe une règle de coupe, propose une relation de dominance et valide ces propositions sur un problème de routage multi-objectif rencontré dans les réseaux informatiques. La règle de coupe permet de réduire l'espace de recherche en calculant une somme pondérée bornant la fonction d'utilité. Plusieurs méthodes pour le calcul d'une telle somme pondérée sont proposées, analysées et comparées. Nous définissons une nouvelle relation de dominance capable de prendre en compte la fonction d'utilité et raffinant la relation de dominance de Pareto. Des conditions suffisantes de cette dominance sont proposées, elles permettent de réduire le nombre de solutions calculées et les temps de calcul par rapport aux méthodes existantes. Nous montrons que les préférences d'un administrateur réseau, dans le cadre du routage multi-objectif au niveau IP, peuvent être modélisées à l'aide d'une fonction d'utilité basée sur l'intégrale de Choquet. Les résultats précédents sont appliqués pour la résolution de ce problème.

Mots-clés : optimisation combinatoire multi-objectif ; plus courts chemins ; préférences ; intégrale de Choquet ; algorithme d'étiquetage.

Abstract

The purpose of this thesis is to handle decision maker preferences for computing multi-objective shortest paths. Decision maker preferences are modeled a priori with a utility function based on a Choquet integral. Instead of computing the set of efficient solutions and choosing afterward an optimal solution according to the utility function, we aim to directly compute an efficient optimal solution satisfying decision maker preferences. In this thesis we develop a pruning rule, propose a dominance relation and valid these propositions in a multi-objective routing problem for computer networks.

The pruning rule allows to reduce the search space by computing a weighted sum that bounds the utility function. Several methods for computing such weighted sum are proposed, analyzed and compared. We define a new dominance relation that considers the utility function and refines Pareto dominance. Sufficient conditions for this dominance are proposed, they allow to reduce the number of solutions computed and computing time compared to existing methods. We highlight that network administrator preferences, for multi-objectif routing in IP network, can be modeled with a utility function based on the Choquet integral. Previous results are applied for solving this problem.

Keywords: multi-objective combinatorial optimisation; shortest paths; preferences; Choquet integral; labeling algorithm.