Élicitation incrémentale des préférences pour l'optimisation multi-objectifs : modèles non-linéaires, domaines combinatoires et approches tolérantes aux erreurs

Nadjet Bourdache

Thèse de doctorat en informatique

Élicitation incrémentale des préférences pour l'optimisation multi-objectifs :
modèles non-linéaires, domaines combinatoires et approches tolérantes aux erreurs

Présentée et soutenue publiquement le 16 décembre 2020 par

Nadjet Bourdache

Jury :

Directeurs de thèse :

Patrice Perny	Professeur, Sorbonne Université
Olivier Spanjaard	Maître de conférences, Sorbonne Université

Rapporteurs :

Sébastien Destercke	Chargé de recherche CNRS, Université de Technologie de Compiègne
Vincent Mousseau	Professeur, CentraleSupélec

Examinateurs :

Christophe Gonzales	Professeur, Aix Marseille Université
Christophe Marsala	Professeur, Sorbonne Université
Meltem Öztürk	Maître de conférences, Université Paris Dauphine

Remerciements

Ces trois années (et quelques mois) ont été pour moi une expérience très enrichissante tant au plan professionnel que personnel. C'est grâce à l'environnement de travail offert par le LiP6 mais également et principalement grâce à toutes les personnes qui ont participé à cette expérience de près ou de loin. Je tiens donc à remercier toutes ces personnes, certaines (mais pas toutes, ne m'en voulez pas) sont citées ci-dessous.

Mes premiers remerciements s'adressent à Olivier Spanjaard (Maître de conférences HDR à Sorbonne Université) pour avoir accepté de diriger cette thèse aux côtés de Patrice Perny (Professeur à Sorbonne Université) qui a non seulement accepté de diriger cette thèse mais aussi mon stage de M2. Je les remercie de m'avoir fait découvrir le monde de la recherche et le métier d'enseignant-chercheur avec autant de disponibilité, d'investissement, de patience, de bienveillance, de gentillesse et d'humour, ce qui a rendu cette thèse très agréable et en a fait une expérience que je ne regretterai jamais. J'ai pu profiter de leurs connaissances, leur expérience, leurs conseils, et leurs encouragements. J'ai toujours pu compter sur leur aide et leurs conseils que ce soit pour mon travail de recherche, pour mes missions d'enseignement, ou même pour des démarches administratives.

Je tiens également à remercier Sébastien Destercke (Chargé de Recherche CNRS HDR à l'Université de Technologie de Compiègne) et Vincent Mousseau (Professeur à Centrale-Supélec) pour l'intérêt qu'ils ont porté à mes travaux de recherche en me faisant l'honneur de rapporter ma thèse en cette période difficile et chargée pour tous. Je remercie également Christophe Marsala (Professeur à Sorbonne Université), Christophe Gonzales (Professeur à l'Université Aix-Marseille) et Meltem Öztürk (Maître de conférence HDR à l'Université Paris Dauphine) pour avoir accepté d'évaluer ma thèse en tant qu'examinateurs. Je remercie aussi Bruno Escoffier (Professeur à Sorbonne Université) pour avoir accepté de faire partie de mon comité de suivi aux côtés de Sébastien Destercke.

Pour finir, je remercie Pierre Fouilhoux (Professeur à l'université Sorbonne Paris Nord) qui m'a encadré en M1 et Mohamed Yagouni (enseignant-chercheur à l'USTHB, Alger) qui m'a encadré en L3, ce fut mon premier contact avec la recherche.

Je remercie également tous les membres des équipes Décision et RO qui m'ont très bien accueillie dans le couloir et sans qui mes années passées au LiP6 auraient été beaucoup moins agréables. Je remercie en particulier mes co-bureau : Hugo (Gilbert) qui m'a accueillie dans le bureau avec une grande sympathie et avec beaucoup de bienveillance, grâce à qui j'ai pu m'intégrer facilement à l'équipe et qui m'a souvent été d'une aide précieuse ; Parham, qui a occupé une place très particulière pour moi dans ce bureau, qui m'a encouragée, aidée et soutenue (et supportée) depuis son arrivée dans le bureau ; Hugo (Martin) avec qui j'ai pu échanger grandes discussions, blagues, conseils et encouragements ; et pour finir Alexandre qui m'a bien fait rire tout au long de cette thèse.

Leur présence a été source de joie et de bonne humeur, sans eux les journées passées au bureau auraient cruellement manqué de joie, de bonne humeur, de fous rires, et de blagues parfois très pourries mais toujours très drôles (ou presque). Je remercie également Anne-Élisabeth que j'ai eu le plaisir de côtoyer depuis le M2, et aussi plusieurs membres (et anciens membres) du couloir ainsi que du couloir voisin : Gaspard, Marvin, Nawal, Santiago, Siao-Leu, Thibaut, Nicolas, Bruno, Fanny, Safia, Pierre, Pierre-Henri, et pour finir Patrice et Olivier (encore oui ! car ce ne sont pas que des directeurs de thèse) avec qui j'ai partagé repas à la cantine, pauses café, goûters et discussions. Je suis très contente d'avoir eu la chance de rencontrer des personnes aussi sympathiques et bienveillantes qui ont fait de cette thèse une superbe expérience.

Je remercie également mes amis et plus particulièrement Lilia et Rania qui ont toujours été là pour moi, et ce depuis l'enfance.

Je réserve ce dernier passage aux personnes qui comptent le plus pour moi, je remercie très chaleureusement mes parents, Ahmed et Djamila, qui m'ont toujours encouragée et soutenue, qui m'ont appris la valeur du travail et l'importance des études et qui m'ont donné cette envie de toujours avancer et progresser. Je remercie également mes soeurs, Nassiba et Yasmine, mon beau-frère, Simon, et, bien évidemment, merthi à ma nièce Lilia qui m'ont également encouragée, aidée, soutenue et apporté du courage et de l'énergie au quotidien. Sans ma famille je n'en serais certainement pas à écrire ces remerciements aujourd'hui.

Table des matières

Liste des notations

Notations du chapitre 1 :

$\mathcal{X}$	ensemble des solutions réalisables du problème,
p	dimension de $\mathcal{X}$,
n	nombre d'objectifs considérés (critères, agents ou scénarios),
$u(x)$	vecteur de performances de la solution x,
N	ensemble des indices des objectifs défini par $N = \{1, \ldots, n\}$,
$\mathcal{Y}$	espace des objectifs défini par $\{u(x) \in \mathbb{R}^n, \forall x \in \mathcal{X}\}$,
$\succsim_P$	relation de dominance de Pareto,
$\succsim_L$	relation de dominance de Lorenz,
$\succsim_{\text{SSD}}$	relation de dominance SSD (*Second Stochastic Order*),
WS_w	somme pondérée de paramètre w,
OWA_w	somme pondérée ordonnée de paramètre w,
W	ensemble des jeux de poids possibles d'une somme pondérée ou d'un OWA,
WOWA_φ	somme doublement pondérée ordonnée de paramètre φ,
C_v	intégrale de Choquet de capacité v,
f_ω	fonction d'agrégation quelconque de paramètre ω,
$\mathcal{W}$	ensemble des paramètres possibles de f_ω.

Notations du chapitre 2 :

$\mathcal{X}$	ensemble des solutions réalisables du problème,
p	dimension de $\mathcal{X}$,
n	nombre d'objectifs considérés (critères, agents ou scénarios),
$u(x)$	vecteur de performances de la solution x,
N	ensemble des indices des objectifs défini par $N = \{1, \ldots, n\}$,
$\mathcal{Y}$	espace des objectifs défini par $\{u(x) \in \mathbb{R}^n, \forall x \in \mathcal{X}\}$,
OWA_w	somme pondérée ordonnée de paramètre w,
W	ensemble des jeux de poids possibles d'une somme pondérée ou d'un OWA,
$PO_W(\mathcal{X})$	ensemble de solutions potentiellement OWA-optimales dans $\mathcal{X}$ lorsque l'espace des paramètres est donné par W,
WOWA_φ	somme doublement pondérée ordonnée de paramètre φ,
Φ	ensemble des paramètres possibles de l'opérateur WOWA,
$PO_\Phi(\mathcal{X})$	ensemble de solutions potentiellement WOWA-optimales dans $\mathcal{X}$ lorsque l'espace des paramètres est donné par Φ,

$\mathcal{A}$	ensemble des paramètres possibles de l'opérateur WOWA dans sa formulation par des fonctions splines,
$PO_{\mathcal{A}}(\mathcal{X})$	ensemble de solutions potentiellement WOWA-optimales dans $\mathcal{X}$ lorsque l'espace des paramètres est donné par $\mathcal{A}$ (formulation spline),
f_ω	fonction d'agrégation quelconque de paramètre ω,
$\mathcal{W}$	ensemble des paramètres possibles de f_ω,
$NO_{\mathcal{W}}(\mathcal{X})$	ensemble de solutions nécessairement f_ω-optimales dans $\mathcal{X}$ lorsque l'espace des paramètres est donné par $\mathcal{W}$,
$\text{PMR}(x, y, \mathcal{W})$	regret maximum de la paire (x, y) dans $\mathcal{W}$ (*Pairwise Max Regret*),
$\text{MR}(x, \mathcal{X}, \mathcal{W})$	regret maximum de x dans $\mathcal{X}$ (*Maximum Regret*),
$\text{MMR}(\mathcal{X}, \mathcal{W})$	plus petit regret maximum (*MiniMax Regret*).

Notations du chapitre 3 :

$\mathcal{X}$	ensemble des solutions réalisables du problème,
p	dimension de $\mathcal{X}$,
n	nombre d'objectifs considérés (critères, agents ou scénarios),
$u(x)$	vecteur de performances de la solution x,
N	ensemble des indices des objectifs défini par $N = \{1, \ldots, n\}$,
$\mathcal{Y}$	espace des objectifs défini par $\{u(x) \in \mathbb{R}^n, \forall x \in \mathcal{X}\}$,
OWA_w	somme pondérée ordonnée de paramètre w,
W	ensemble des jeux de poids possibles d'une somme pondérée ou d'un OWA,
$\text{PMR}(x, y, \mathcal{W})$	regret maximum de la paire (x, y) dans $\mathcal{W}$ (*Pairwise Max Regret*),
$\text{MR}(x, \mathcal{X}, \mathcal{W})$	regret maximum de x dans $\mathcal{X}$ (*Maximum Regret*),
$\text{MMR}(\mathcal{X}, \mathcal{W})$	plus petit regret maximum (*MiniMax Regret*).

Notations du chapitre 4 :

$\mathcal{X}$	ensemble des solutions réalisables du problème,
p	dimension de $\mathcal{X}$,
n	nombre d'objectifs considérés (critères, agents ou scénarios),
$u(x)$	vecteur de performances de la solution x,
N	ensemble des indices des objectifs défini par $N = \{1, \ldots, n\}$,
WS_w	somme pondérée de paramètre w,
OWA_w	somme pondérée ordonnée de paramètre w,
W	ensemble des jeux de poids possibles d'une somme pondérée ou d'un OWA,
C_v	intégrale de Choquet de capacité v,
f_ω	fonction d'agrégation quelconque de paramètre ω,
$\mathcal{W}$	ensemble des paramètres possibles de f_ω,
$p(\omega)$	densité de probabilité associée à $\mathcal{W}$,
$p(\omega \mid r)$	densité de probabilité mise à jour à l'aide de la réponse r,
S	échantillon de paramètres tiré selon $p(\omega)$,
$\text{PER}(x, y, \mathcal{W})$	regret espéré de la paire (x, y) dans $\mathcal{W}$ (*Pairwise expected Regret*),
$\text{MER}(x, \mathcal{X}, \mathcal{W})$	regret espéré maximum de x dans $\mathcal{X}$ (*Maximum Expected Regret*),
$\text{MMER}(\mathcal{X}, \mathcal{W})$	plus petit regret espéré maximum (*MiniMax expected Regret*).

Notations du chapitre 5 :

$\mathcal{X}$	ensemble des solutions réalisables du problème,
p	dimension de $\mathcal{X}$,
n	nombre d'objectifs considérés (critères, agents ou scénarios),
$u(x)$	vecteur de performances de la solution x,
N	ensemble des indices des objectifs défini par $N = \{1, \ldots, n\}$,
WS_w	somme pondérée de paramètre w,
OWA_w	somme pondérée ordonnée de paramètre w,
W	ensemble des jeux de poids possibles d'une somme pondérée ou d'un OWA,
f_ω	fonction d'agrégation quelconque de paramètre ω,
$\mathcal{W}$	ensemble des paramètres possibles de f_ω,
W_x	polyèdre d'optimalité associé à la solution $x \in \mathcal{X}$,
$P(x)$	probabilité que la solution x soit f_ω-optimale,
$P(x\|r)$	probabilité que la solution x soit f_ω optimale mise à jour à l'aide de la réponse r,
$\text{PER}(x, y, \mathcal{W}, P)$	regret espéré de la paire (x, y) dans $\mathcal{W}$ (*Pairwise expected Regret*),
$\text{MER}(x, \mathcal{X}, \mathcal{W}, P)$	regret espéré maximum de x dans $\mathcal{X}$ (*Maximum Expected Regret*),
$\text{MMER}(\mathcal{X}, \mathcal{W}, P)$	plus petit regret espéré maximum (*MiniMax expected Regret*).

Introduction

La théorie de la décision algorithmique est une thématique de recherche récente qui se situe au carrefour de deux disciplines : la théorie de la décision et l'algorithmique. La *théorie de la décision* s'intéresse principalement aux aspects de *modélisation* et d'*agrégation* des préférences dans une situation de prise de décision dans un environnement risqué ou incertain, ou lors de la prise en compte de plusieurs critères ou des points de vue de plusieurs agents. Il s'agit de concevoir des outils formels permettant la mise en œuvre de principes normatifs (comportements décisionnels rationnels en théorie) et la description de préférences propres à un individu. La théorie de la décision *algorithmique* quant à elle s'intéresse *également* aux aspects *computationnels* et vise à concevoir des méthodes d'aide à la décision dans le risque, dans l'incertain, en présence de critères multiples, ou de plusieurs agents (aux préférences potentiellement conflictuelles), notamment sur domaine *combinatoire* ou *continu*, menant ainsi à plusieurs domaines de recherche : l'optimisation multi-objectifs, le choix social computationnel, l'optimisation robuste et dans l'incertain. Dans cette thèse, on s'intéresse à la thématique de l'optimisation multicritère, multi-agents et de décision dans le risque en présence de préférences partiellement connues.

Concevoir des méthodes de résolution prenant en compte différents points de vue constitue un challenge d'un point de vue algorithmique, car la nature (très) souvent conflictuelle des objectifs rend la notion d'optimalité subjective, et donc très difficile à définir de manière générique, ce qui constitue déjà une difficulté importante dans la détermination d'une solution préférée parmi un ensemble *explicite* de solutions. Cette difficulté est naturellement accentuée lorsque le problème étudié est défini sur domaine combinatoire. En effet, ces problèmes peuvent être NP-difficiles même dans le cas mono-objectif "classique", ou le deviennent lorsque l'on passe au cas multi-objectifs. L'intérêt porté à l'aide à la décision multi-objectifs provient de ses larges applications pratiques. Il s'agit souvent d'utiliser les méthodes développées dans le domaine de la théorie de la décision pour concevoir des outils de recommandation et d'aide à la décision dans différents contextes, par exemple : la composition de comités, la gestion de budgets, de production ou de stocks, l'attribution de tâches, de ressources ou de logements sociaux, ou encore la recommandation de films, séries, vidéos, livres, etc.

Lors de la conception de systèmes d'aide à la décision dans un contexte multi-objectifs, une modélisation des préférences du décideur est nécessaire afin de pouvoir expliciter formellement un modèle représentant son système de valeurs, et de garantir la cohérence interne de ses préférences (aspect normatif). Une approche de modélisation courante en théorie de la décision est d'utiliser des opérateurs d'agrégation permettant de traduire des préférences parfois très complexes en une formulation mono-objectif plus simple. Faire une recommandation personnalisée au décideur revient alors à chercher une solution optimale au sens de la fonction d'agrégation considérée. Les modèles de la littérature sont très

nombreux et permettent de modéliser un très large panel de comportements décisionnels différents. On peut citer, par exemple, l'opérateur (bien connu) de la somme pondérée (WS pour *Weighted Sum)*, la somme pondérée ordonnée (OWA pour *Ordered Weighted Average*) [Yager, 1998], la somme doublement pondérée ordonnée (WOWA pour *Weighted Oredered Weighted Average*) [Torra, 1997] ou encore l'intégrale de Choquet, e.g., [Grabisch et Labreuche, 2010]. Même si l'on se ramène à un problème mono-objectif, l'optimisation de ces opérateurs sur domaine combinatoire n'est pas toujours simple car certains d'entre eux ne sont pas linéaires. Cette non-linéarité se justifie par des besoins descriptifs ou des principes normatifs, mais implique une difficulté de résolution supplémentaire. Il existe néanmoins dans la littérature des méthodes de résolution permettant d'optimiser la valeur d'un opérateur non-linéaire, par exemple, pour OWA on peut citer [Ogryczak et Śliwiński, 2003; Galand et Spanjaard, 2012], pour WOWA [Ogryczak et Śliwiński, 2007], et pour l'intégrale de Choquet [Lesca et Perny, 2010; Martin et Perny, 2020].

La richesse du pouvoir descriptif de ces opérateurs provient du fait qu'ils sont tous caractérisés par un paramètre qui permet d'adapter le modèle décisionnel aux préférences particulières de chaque décideur. Les méthodes de résolution citées plus haut peuvent alors être très efficaces lorsque les préférences du décideur sont précisément connues et que le paramètre du modèle est fixé. Elles deviennent néanmoins inapplicables dans le cas inverse. Or, dans des situations réelles, il arrive très souvent que les préférences du décideur ne soient pas connues de manière précise. Déterminer un paramètre modélisant ces préférences semble alors être un problème d'une importance critique pour la détermination d'une bonne recommandation. Lorsque les paramètres sont imprécisément connus, il est toutefois possible de déterminer des sous-ensembles de solutions, appelées des solutions *potentiellement optimales*, car elles sont susceptibles d'être optimales si l'on considère certains jeux de paramètres admissibles (voir [Konczak et Lang, 2005; Greco *et al.*, 2009; Benabbou et Perny, 2015b]). Plus les informations préférentielles dont on dispose sont riches, plus la connaissance des valeurs de paramètres sera précise, et plus l'ensemble des solutions potentiellement optimales sera restreint. De ce fait, une élicitation des préférences du décideur peut permettre de suffisamment réduire l'ensemble de solutions potentiellement optimales pour pouvoir faire apparaître une solution *nécessairement* optimale, c'est-à-dire, une solution qui reste optimale pour tous les jeux de paramètres restants.

L'élicitation des préférences sur domaine combinatoire est une tâche difficile à accomplir dans la prise de décision individuelle et collective. Il s'agit de mettre au point des procédures d'interactions avec le décideur permettant de récolter des informations sur ses préférences, et de réduire suffisamment l'incertitude concernant les valeurs des paramètres les représentant pour pouvoir déterminer une solution optimale. La difficulté de la conception de telles méthodes réside dans la définition exacte de ces interactions. En effet, un compromis est généralement à faire entre difficulté des questions posées et pouvoir informatif des réponses apportées par le décideur. Dans bien des cas, une élicitation précise des préférences n'est pas envisageable à cause du nombre (potentiellement) exponentiel de solutions à comparer qui représenterait une charge d'élicitation trop importante pour le décideur. On tentera alors d'obtenir uniquement les informations préférentielles nécessaires à la détermination d'une recommandation optimale, en supposant une certaine régularité dans les préférences justifiant l'utilisation d'un modèle décisionnel. Une approche efficace pour l'élicitation est d'intégrer des interactions avec le décideur directe-

ment dans l'algorithme de résolution. De cette manière, on adapte le questionnaire et la recherche d'une solution optimale aux préférences spécifiques du décideur et à la nature des solutions disponibles.

Ainsi, l'objectif de cette élicitation est de pouvoir mieux discriminer entre les solutions possibles du problème et non d'apprendre le modèle de manière exacte. Cette approche, dite d'*élicitation incrémentale*, a fait l'objet de nombreux travaux récents. Ces travaux portent essentiellement sur la résolution de problèmes définis sur ensembles *explicites* de solutions (avec des modèles linéaires ou non) [White III *et al.*, 1984; Salo et Hamalainen, 2001; Wang et Boutilier, 2003; Braziunas et Boutilier, 2007; Perny *et al.*, 2016; Benabbou et Perny, 2017], ou sur domaine combinatoire mais en utilisant principalement des modèles linéaires [Boutilier *et al.*, 2006a; Gelain *et al.*, 2010; Benabbou et Perny, 2015b,c; Brero *et al.*, 2018]. L'élicitation d'un opérateur non-linéaire pour la résolution d'un problème d'optimisation défini sur domaine combinatoire est par contre beaucoup moins étudiée. Il existe quelques travaux abordant cette problématique, mais ces méthodes sont généralement soit spécifiques au problème traité [Benabbou et Perny, 2015a], soit approchées [Benabbou *et al.*, 2020], soit consacrées à l'élicitation des valeurs d'utilité plutôt que des paramètres préférentiels [Vendrov *et al.*, 2020].

Par ailleurs, notons qu'un grand nombre des travaux concernant l'élicitation des préférences sont fondées sur l'hypothèse que le décideur répond toujours correctement aux questions préférentielles posées. Or, dans des situations réelles, il peut être considéré comme utopique de présumer que le décideur soit capable de répondre à toutes les questions posées de manière cohérente. C'est pourquoi différentes méthodes ont été proposées, par exemple par Chajewska *et al.* [2000]; Guo et Sanner [2010]; Sauré et Vielma [2019]; Vendrov *et al.* [2020], mais ces méthodes traitent essentiellement le cas de l'élicitation d'un modèle de décision linéaire pour la résolution de problèmes définis sur domaine explicite.

Positionnement de la thèse. Dans cette thèse, nous nous intéressons à la conception de méthodes d'élicitation incrémentale des préférences dans une situation de prise de décision multicritère, multi-agents ou de décision dans le risque. Nous nous intéressons plus particulièrement aux compromis que fait le décideur sur les différents objectifs. Pour cela, les méthodes que nous introduisons sont fondées sur l'utilisation d'un modèle décisionnel dont les paramètres permettent de modéliser les préférences particulières du décideur. Ainsi, l'élicitation de ses préférences revient à éliciter les paramètres du modèle décisionnel utilisé. L'originalité des travaux de cette thèse provient de deux aspects en particulier :

1. L'élicitation des paramètres de modèles *non-linéaires* pour la détermination d'une solution préférée parmi un ensemble de solutions défini sur *domaine combinatoire* et contenant un nombre de solutions potentiellement exponentiel. Ce type de problèmes n'est pas facile à résoudre mais représente un réel intérêt pratique compte tenu du pouvoir descriptif des modèles non-linéaires.
2. L'élicitation des paramètres de modèles (linéaires ou non) en prenant en compte la présence de *possibles incohérences* dans les réponses du décideur, pour la détermination d'une solution préférée (parmi un ensemble de solutions défini de manière explicite ou implicite). De telles approches permettent de considérer des situations "réalistes" où le décideur est un être humain ne pouvant avoir un modèle précis en tête lui permettant de répondre à toute question de manière précise et cohérente avec les autres réponses données.

Organisation du document. Ce document se décompose en cinq chapitres. Le premier présente le cadre général de cette thèse. On y trouve la définition de la problématique du choix multi-objectifs dans un cadre où les préférences sont partiellement connues, puis un état de l'art concernant la modélisation et l'élicitation des préférences dans ce cadre. Le chapitre 2 présente nos premières contributions concernant l'élicitation des paramètres d'un modèle non-linéaire pour la résolution de problèmes définis sur domaine combinatoire. On y présente différentes méthodes fondées sur une énumération implicite et méthodique des solutions réalisables du problème traité. Le chapitre 3 introduit une approche différente pour résoudre le même type de problèmes que dans le chapitre précédent ; cette approche consiste à guider l'exploration de l'espace des solutions par un travail d'exploration et de mise à jour de l'espace des paramètres. Le chapitre 4 s'intéresse à la prise en compte de possibles erreurs (liées à la difficulté des questions ou à l'évolution des préférences) ou incohérences dans les réponses du décideur par la gestion d'une distribution de probabilité continue sur l'espace des paramètres. On s'intéresse alors, dans un premier temps, à la résolution de problèmes définis sur domaines explicites pour l'élicitation de modèles non-linéaires, puis on étend l'approche à la résolution de problèmes définis sur domaine combinatoire, mais cette fois principalement en utilisant un modèle linéaire. Le chapitre 5 présente enfin une méthode fondée sur la gestion d'une distribution de probabilité discrète permettant de réduire la charge de calcul lors de l'élicitation (par rapport à l'utilisation d'une distribution continue). Un chapitre de conclusion synthétise ensuite les contributions de cette thèse et présente quelques pistes de recherches futures.

Chapitre 1

Modélisation et élicitation de préférences en décision multi-objectifs

Résumé du chapitre

Ce chapitre vise à présenter le cadre général de cette thèse. Nous présentons tout d'abord la problématique de choix lorsque plusieurs points de vue sont considérés dans la prise de décision (critères, agents ou scénarios), puis nous réalisons un bref état de l'art concernant la modélisation et l'élicitation des préférences dans un tel cadre. Dans un premier temps, nous présentons différents modèles de décision permettant de représenter des comportements décisionnels de natures et de complexités variées. Les modèles auxquels nous nous intéressons sont des opérateurs d'agrégation caractérisés par des paramètres (vecteurs ou fonctions de pondération) permettant d'adapter le comportement décisionnel qu'ils modélisent aux préférences particulières du décideur. Ces préférences n'étant pas précisément connues, il n'est initialement pas possible de déterminer la valeur exacte des paramètres sans interactions avec le décideur. Une méthode d'élicitation est alors nécessaire afin de préciser suffisamment la valeur de ces paramètres pour pouvoir déterminer une recommandation qui puisse satisfaire le décideur. La seconde partie de ce chapitre est consacrée à un état de l'art concernant les méthodes d'élicitation existantes. Pour finir, une conclusion résume les principaux éléments introduits dans ce chapitre et rappelle le positionnement de cette thèse par rapport à la littérature concernant l'élicitation des préférences.

1.1 Introduction

L'intérêt grandissant pour l'étude et la conception de systèmes de recommandation et d'aide à la décision personnalisés provient de l'omniprésence de la prise de décision dans le quotidien de chacun, tant pour les aspects personnels de nos vies que pour les aspects professionnels. La prise de décision se manifeste dans des situations banales du quotidien (par exemple, le choix d'un film, d'un plat au restaurant ou d'un cadeau pour un proche), dans des situations moins fréquentes et plus importantes (par exemple, l'achat d'un nouvel ordinateur, le choix d'une personne à embaucher), tout comme elle se manifeste dans des situations bien plus critiques où les choix ont des conséquences durables (par exemple, le choix d'une maison ou d'un appartement, le choix d'une stratégie d'entreprise, ou un choix d'investissement). Dans de telles situations, le besoin d'avoir recours à un système d'aide à la décision dépend de la difficulté de la décision à prendre. Cette difficulté est souvent liée : à la nature de la décision (par exemple, classer n candidats est plus difficile que de choisir un candidat préféré parmi les n candidats), au nombre d'alternatives à comparer, à l'abondance de la (ou des) ressource(s) nécessaire(s) (par exemple, le budget accordé à un achat), et à l'importance des enjeux qu'elle peut impliquer. Une autre source de difficulté très importante est la présence de plusieurs objectifs à optimiser. Cette difficulté provient de la nature souvent conflictuelle des objectifs qui nécessite la détermination de compromis (lorsque ce n'est pas le cas, on peut généralement se ramener au cas d'un problème mono-objectif plus simple à traiter). Ce type de situations survient dans trois contextes différents menant à trois thématiques de recherche liées à la théorie de la décision algorithmique :

- **Décision multicritère :** plusieurs critères d'évaluation, très souvent conflictuels, peuvent être considérés pour estimer la qualité d'une alternative (qualité, prix, ...). Chaque objectif représente alors une performance selon un critère donné.
- **Décision multi-agents :** plusieurs agents, avec des opinions souvent divergentes, peuvent être impliqués dans la prise de décision. Chaque objectif représente alors la satisfaction d'un agent (ou le prix qu'il paie). Lorsque l'on souhaite déterminer une solution qui répartit la satisfaction des agents de manière équitable, on parle alors d'optimisation *équitable*.
- **Décision dans le risque :** les conséquences d'une décision peuvent ne pas être prédictibles de manière certaine car elles dépendent d'éléments extérieurs incertains (les conditions météorologiques par exemple). Plusieurs scénarios peuvent alors être à envisager, et chaque objectif représente l'évaluation globale d'une solution dans un scénario donné. Lorsque les probabilités des différents scénarios sont connues, il s'agit bien de *décision dans le risque* ; si par contre ces probabilités ne sont pas connues, alors on parle de *décision dans l'incertain*. Notons que l'on parle souvent d'*optimisation robuste* pour désigner la résolution de problèmes de décision dans le risque ou dans l'incertain sur domaine combinatoire.

Dès lors que plusieurs objectifs sont à optimiser, la notion d'optimalité devient subjective et dépend des préférences spécifiques à chaque individu. Il existe en effet différents comportements décisionnels face à ce type de problèmes de décision. Par exemple, un individu A peut accorder plus ou moins d'importance à un objectif par rapport aux autres, un individu B considérer que la satisfaction globale des objectifs constitue un bon critère d'optimalité, alors qu'un individu C voudrait que les objectifs soient satisfaits de la ma-

nière la plus équilibrée possible. Finalement, il existe une infinité de nuances entre ces trois comportements. Afin de guider ou d'automatiser une prise de décision, on doit alors disposer d'un certain nombre d'informations concernant les préférences de la personne responsable de la décision, appelée *le décideur*. Or, il est en pratique rarement facile pour lui d'exprimer de manière précise ce qu'il recherche. En effet, même si le décideur sait qu'il privilégie un critère (ou un ensemble de critères) par rapport à un (ou plusieurs) autre(s) critère(s), il peut ne pas être capable de préciser l'importance exacte de chaque critère. Ainsi, en amont de la résolution du problème, il faut *éliciter* et *modéliser* les préférences du décideur afin de pouvoir déterminer une bonne recommandation à lui présenter. La prise de décision se fait donc en interaction avec le décideur et la pertinence du système d'aide à la décision dépend de la manière dont ces interactions sont faites, et de la manière dont les informations issues de ces interactions sont traitées. Un aspect important peut alors être à considérer dans la conception d'un système d'aide à la décision : l'exactitude et la cohérence des informations fournies par le décideur lors de ses interactions avec le système. Une *méthode d'élicitation robuste* doit pouvoir permettre de déterminer une bonne recommandation malgré la présence d'informations préférentielles inexactes, incohérentes, ou variables au cours du temps. Or, il n'est pas toujours évident de traiter de telles informations, car cela vient souvent au détriment de l'efficacité de la procédure d'élicitation (en termes de nombre d'interactions avec le décideur et de temps passé entre deux interactions). Un équilibre est alors à trouver entre pertinence de la recommandation et efficacité de la procédure d'élicitation.

Ce chapitre présente brièvement l'optimisation multi-objectifs et la littérature concernant l'utilisation d'opérateurs d'agrégation paramétrés pour modéliser et éliciter efficacement les préférences du décideur. Il est organisé de la manière suivante : on définit tout d'abord dans la section 1.2 la problématique du choix multi-objectifs, la modélisation des préférences et l'optimisation multi-objectifs fondées sur des modèles de décision. On présente notamment les modèles décisionnels auxquels on s'intéresse dans cette thèse : somme pondérée, somme pondérée ordonnée, somme doublement pondérée ordonnée et intégrale de Choquet. On introduit également la notion d'optimisation multi-objectifs avec connaissance partielle des paramètres préférentiels. On présente ensuite dans la section 1.3 différents travaux de la littérature qui se sont intéressés à l'élicitation des préférences dans des contextes proches du nôtre.

1.2 Optimisation multi-objectifs fondée sur des modèles de décision

Un problème de décision multi-objectifs est caractérisé par la prise en compte explicite de plusieurs objectifs à optimiser. On le définit de la manière suivante : on dispose d'un ensemble de solutions $\mathcal{X}$ défini explicitement (par exemple par un ensemble d'actions, d'objets ou de candidats) ou implicitement (par exemple, à l'aide d'un ensemble de contraintes linéaires sur des variables de décision). Dans le second cas, $\mathcal{X}$ contient un nombre exponentiel de solutions réalisables. L'ensemble $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_p$ est appelé *espace des solutions*, et sa dimension p dépend du problème traité. Généralement, $\mathcal{X}_i \subseteq \mathbb{R}^c, \forall i \in \{1, \ldots, p\}$, avec $c \in \mathbb{N} \setminus \{0\}$. Toute solution $x \in \mathcal{X}$ s'écrit alors $x = (x_1, \ldots, x_p)$ et son évaluation selon les n objectifs considérés est donnée par son vec-

teur de performances noté $u(x) = (u_1(x), \ldots, u_n(x))$. Notons $N = \{1, \ldots, n\}$ l'ensemble des objectifs. Selon le contexte, la composante $u_i(x), i \in N$, représente :

- la performance de la solution x sur le critère i (décision multicritère),
- la performance de la solution x pour l'agent i (décision multi-agents),
- la performance de la solution x dans le scénario i (décision dans le risque ou dans l'incertain).

Dans tous les cas, on définit l'*image de* $\mathcal{X}$ dans l'*espace des objectifs* par $\mathcal{Y} = \{u(x) \in \mathbb{R}^n, \forall x \in \mathcal{X}\}$. La formulation précise du problème de décision multi-objectifs dépend ensuite de la problématique étudiée. Les trois problématiques principalement étudiées en aide à la décision sont [Roy, 1985] :

- le *choix*, ou la détermination d'une solution préférée,
- le *tri*, ou la répartition des solutions réalisables selon différentes catégories ou classes,
- le *rangement*, ou le fait d'ordonner toutes les solutions selon un ordre de préférences.

Dans cette thèse, nous nous focalisons sur la problématique du choix qui consiste à déterminer une solution préférée parmi l'ensemble des solutions de $\mathcal{X}$. Dans ce cas, le problème de décision s'écrit :

$$\begin{cases} opt_1 \ u_1(x) \\ \quad \vdots \\ opt_n \ u_n(x) \\ x \in \mathcal{X} \ (\text{ou } u(x) \in \mathcal{Y}) \end{cases} \tag{1.1}$$

où opt_i désigne la maximisation ou la minimisation de $u_i(x)$ dans $\mathcal{X}$. Dans le reste du document (sauf mention contraire), nous supposerons que les objectifs sont tous à maximiser, i.e., $opt_i = \max, \forall i \in N$. Cette hypothèse n'est pas restrictive puisque toute opération de minimisation peut être ramenée à une opération de maximisation par un simple changement de signe. L'objectif est donc de trouver une solution x qui maximise au mieux chacun des n objectifs. Du fait de la nature souvent conflictuelle des objectifs, il arrive très souvent qu'améliorer la performance d'une solution sur un objectif dégrade sa performance sur un autre objectif (par exemple, améliorer la qualité d'un produit implique très souvent une augmentation de son prix). En d'autres termes, une solution optimale pour un objectif donné peut être largement battue sur un autre objectif. Il est donc très rare en pratique qu'il existe une solution qui optimise simultanément les n objectifs. L'exemple suivant illustre ce propos :

Exemple 1.1. *Supposons que l'on dispose d'un ensemble de solutions dont les vecteurs de performances sont représentés par des ronds dans l'espace bi-objectifs de la figure 1.1. On peut facilement voir sur cette figure que les deux solutions dont les vecteurs de performances sont représentés par des ronds blancs optimisent chacune l'un des deux objectifs mais offrent une performance médiocre pour l'autre objectif. D'autre part, il est impossible de dire objectivement que l'une des solutions est meilleure que l'autre. Définir une préférence entre ces deux solutions dépend entièrement des préférences du décideur.*

Sans informations préférentielles et sans outils supplémentaires, il est donc assez rare de pouvoir déterminer une solution optimale pour le décideur en utilisant uniquement les données du problème présentes dans la formulation (1.1). On peut tout de même

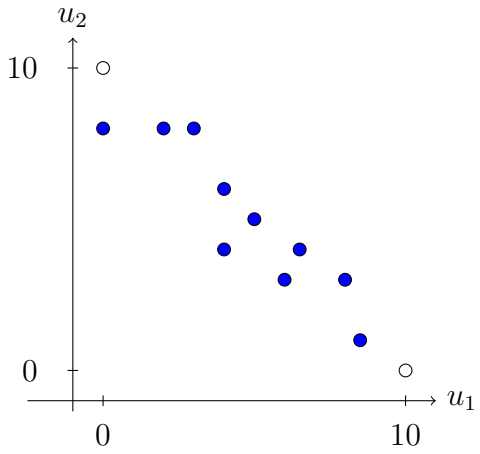


FIGURE 1.1 – Exemple d'un ensemble de vecteurs de performances dans l'espace bi-objectifs.

identifier des solutions potentiellement intéressantes pour lui à l'aide de quelques *règles de dominance*.

1.2.1 Règles de dominance

Bien que la notion d'optimalité dans un contexte de décision multi-objectifs soit subjective, et qu'il soit impossible de la définir de manière générique et indépendante des préférences spécifiques du décideur, il existe différentes règles de dominance, plus ou moins discriminantes, permettant de déterminer un ensemble de solutions pouvant susciter de l'intérêt pour le décideur. On présente ici trois règles de dominance importantes qui modélisent des préférences de natures différentes. La première que nous présentons constitue une règle incontournable lorsque l'on souhaite modéliser des préférences rationnelles.

a. La dominance de Pareto

Cette relation de dominance consiste à comparer les solutions sur chaque objectif séparément et à écarter toute solution dominée sur tous les objectifs *simultanément*. Une telle solution est nécessairement inintéressante puisqu'il existe une ou plusieurs solutions pouvant l'améliorer sur un ou plusieurs objectifs sans en dégrader d'autres. Plus formellement, on définit cette relation par :

Définition 1.1. *Soient x et y deux solutions de $\mathcal{X}$, on a :*

$$x \succsim_P y \Leftrightarrow \forall i \in N, u_i(x) \geq u_i(y)$$

on dit que x domine y au sens de Pareto, ou encore que y est Pareto-dominée par x.

La partie asymétrique de cette relation de dominance se définit par :

$$x \succ_P y \Leftrightarrow (x \succsim_P y \text{ et } non(y \succsim_P x))$$

Notons que, par abus de notation, il peut arriver que l'on écrive $u(x) \succ_P u(y)$ (ou $u(x) \succsim_P u(y)$). En utilisant la relation de Pareto-dominance, on peut définir l'ensemble des solutions potentiellement intéressantes pour le décideur comme l'ensemble des solutions non-dominées au sens de Pareto :

Définition 1.2. *L'ensemble* $\mathrm{ND}_P(\mathcal{X})$ *défini par*

$$\mathrm{ND}_P(\mathcal{X}) = \{x \in \mathcal{X} | \forall y \in \mathcal{X}, non(y \succ_P x)\}$$

donne l'ensemble des solutions Pareto-optimales*, aussi appelé* l'ensemble de Pareto*.*

À partir des définitions 1.1 et 1.2, on peut observer que l'ensemble $\mathrm{ND}_P(\mathcal{X})$ est non-vide car la relation de Pareto-dominance de la définition 1.1 est transitive. De plus, on peut observer que toute solution de $\mathrm{ND}_P(\mathcal{X})$ (solution Pareto-optimale) est telle qu'il n'est pas possible d'améliorer sa performance pour un objectif donné sans dégrader la performance d'au moins un autre objectif (sinon elle ne pourrait pas être dans $\mathrm{ND}_P(\mathcal{X})$). Les solutions Pareto-optimales représentent tous les compromis potentiellement intéressants dans le cadre d'un problème de choix.

Exemple 1.1 (suite). *Supposons que l'on dispose du même ensemble de solutions que dans l'exemple précédent. Sur la figure 1.2, les solutions de l'ensemble de Pareto sont représentées par des ronds blancs :*

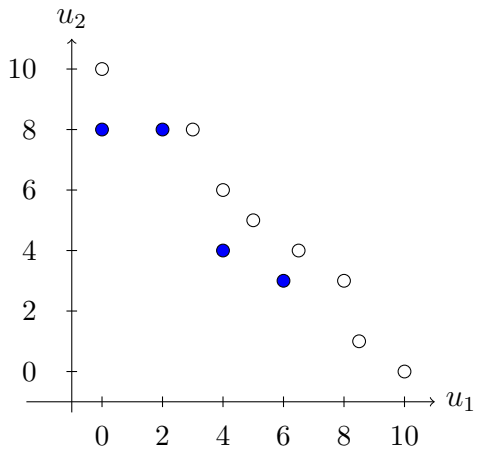


FIGURE 1.2 – Exemple d'ensemble de Pareto (ronds blancs).

On peut facilement voir sur cette figure que, par exemple, le vecteur de performances $(0, 8)$ *est Pareto-dominé à la fois par* $(3, 8)$*, qui est meilleur pour le premier objectif, et par* $(0, 10)$ *qui est meilleur pour le second objectif. On peut également donner l'exemple du vecteur* $(6, 3)$*, qui est strictement Pareto-dominé par* $(6.5, 4)$ *(meilleur sur les 2 critères à la fois). On peut également voir que pour toute solution Pareto-optimale, une amélioration sur un objectif se fait toujours au détriment de la performance sur l'autre objectif. Par exemple, on peut facilement trouver une solution dont le vecteur de performances est meilleur que* $(0, 10)$ *sur le premier objectif, mais cette solution sera toujours moins performante sur le second objectif.*

Même si la dominance de Pareto décrit un comportement rationnel, et qu'elle permet d'écarter des solutions dominées et donc inintéressantes pour le décideur, elle ne constitue pas une relation de préférence très riche. D'une part, elle ne définit pas un ordre total sur les solutions de $\mathcal{X}$. D'autre part, elle ne possède pas une capacité descriptive très importante puisqu'elle ne traduit pas de comportements décisionnels autres que la rationalité des préférences. L'ensemble de Pareto peut contenir un très grand nombre de

solutions dont beaucoup peuvent ne présenter aucun intérêt réel pour le décideur. On peut par exemple constater sur la figure 1.2 que l'ensemble de Pareto contient des solutions très différentes ne pouvant pas convenir à n'importe quel type de préférences : certaines privilégient (plus ou moins) le premier objectif alors que d'autres privilégient le second, certaines ont un vecteur de performances très déséquilibré alors que d'autres ont un vecteur de performances parfaitement équilibré. Nous présentons maintenant une relation de dominance un peu plus riche permettant de privilégier les solutions ayant un vecteur de performances équilibré lorsque l'on sait que le décideur a une préférence pour de telles solutions : la dominance de Lorenz.

b. La dominance de Lorenz

Il existe différents contextes où le décideur privilégie les solutions ayant un vecteur de performances équilibré. En décision multi-agents, il semble assez naturel qu'un décideur supervisant une décision collective soit à la recherche d'une solution qui permette une répartition *équitable* de la satisfaction des différents agents. En décision multicritère, il arrive très souvent, lorsque les critères ont tous la même importance, de vouloir trouver une solution où les performances selon les différents critères sont *équilibrées*. Et finalement, en décision dans le risque, lorsque les scénarios ont la même probabilité, on cherche souvent à déterminer une solution *robuste* qui permet d'assurer une bonne performance quel que soit le scénario qui finit par se produire. Dans ces différents contextes, la dominante de Pareto ne suffit pas à traduire la notion d'équité/équilibre/robustesse.

La dominance de Lorenz caractérise une notion d'équilibre décrite par un axiome connu sous le nom de *principe de transfert de Pigou-Dalton*, e.g., [Shorrocks, 1983; Moulin, 1988]. Cet axiome formalise la notion d'équilibre par une idée très simple : le transfert d'une quantité $0 < \varepsilon \leq u_i(x) - u_j(x)$ de la performance $u_i(x)$ vers la performance $u_j(x)$ offre une meilleure répartition des valeurs de performances. Notons que cette règle de dominance vient de la mesure des inégalités concernant des distributions de revenus. Plus formellement :

Définition 1.3. ***(Principe de transfert)*** *Soit une solution $x \in \mathcal{X}$ dont le vecteur de performances $u(x) \in \mathcal{Y}$ est tel qu'il existe deux indices $i \in N, j \in N$ pour lesquels $u_i(x) > u_j(x)$. Pour tout ε tel que $0 < \varepsilon \leq u_i(x) - u_j(x)$ on a :*

$$(u_1(x), \ldots, u_i(x) - \varepsilon, \ldots, u_j(x) + \varepsilon, \ldots, u_n(x)) \succ u(x)$$

Ainsi, si $u_i(x) > u_j(x)$ l'amélioration de la performance $u_j(x)$ (d'une quantité ε) au détriment de la performance $u_i(x)$ donne une solution au vecteur de performances plus équilibré si la quantité ε ne dépasse pas l'écart $u_i(x) - u_j(x)$ (sinon le déséquilibre est accentué). Autrement dit, un tel transfert mène à une solution plus intéressante pour un décideur ayant une préférence pour des performances équilibrées.

Exemple 1.2. *Soient x et y deux solutions associées aux vecteurs de performances $u(x) = (5, 5)$ et $u(y) = (0, 10)$. La solution x domine la solution y au sens de la règle de dominance induite par le principe de transfert de Pigou-Dalton. En effet, il existe un transfert de valeur $\varepsilon = 5$ menant de y à x : $u(x) = (u_1(y) + 5, u_2(y) - 5) = (0 + 5, 10 - 5) = (5, 5)$.*

Par définition de ce principe de transfert, la règle de dominance qu'il induit ne peut s'appliquer qu'à la comparaison de vecteurs de même moyenne. En pratique, un grand nombre de solutions reste donc impossible à comparer. Pour remédier à cela, on peut faire appel à une règle plus générale qui consiste à combiner le principe de transfert avec la dominance de Pareto. L'exemple suivant illustre ce propos assez simplement :

Exemple 1.3. *Soient x et y deux solutions associées aux vecteurs de performances $u(x) = (5, 5)$ et $u(y) = (3, 6)$. Ces deux solutions ont une moyenne différente et, de ce fait, aucune ne peut être obtenue à partir de l'autre en effectuant un transfert de Pigou-Dalton. Par contre, on peut facilement voir que $u(x)$ Pareto-domine $(4, 5)$ qui à son tour domine $u(y)$ au sens du principe de transfert car $(4, 5) = (3+1, 6-1)$. On obtient alors par transitivité que $u(x) \succ u(y)$.*

On peut donc déterminer l'existence ou non d'une relation de dominance entre deux solutions n'ayant pas nécessairement la même moyenne en déterminant l'existence d'une séquence de dominances de Pareto et de transferts de Pigou-Dalton. Néanmoins, on pourrait craindre que le fait de déterminer l'existence d'une telle séquence soit une opération très fastidieuse. Toutefois il n'en est rien : il existe une caractérisation simple des vecteurs pouvant être comparés de cette manière. Cette caractérisation utilise la notion de dominance de Lorenz généralisée [Marshall *et al.*, 1979; Shorrocks, 1983] :

Définition 1.4. *Soit une solution $x \in \mathcal{X}$ associée au vecteur de performances $u(x) \in \mathcal{Y}$. Le vecteur de Lorenz associé à la solution x est défini par*

$$L(x) = (u_{(1)}(x), u_{(1)}(x) + u_{(2)}(x), \ldots, u_{(1)}(x) + \cdots + u_{(n)}(x))$$

où $u_{(i)}(x)$ est la i^e plus petite composante du vecteur $u(x)$.

La *dominance de Lorenz généralisée* est définie à partir des vecteurs de Lorenz :

Définition 1.5. *Soient x et y deux solutions de $\mathcal{X}$. On a :*

$$x \succsim_L y \Leftrightarrow L(x) \succsim_P L(y) \Leftrightarrow \forall i \in N, \sum_{j=1}^{i} u_{(j)}(x) \geq \sum_{j=1}^{i} u_{(j)}(y)$$

on dit alors que x domine y au sens de Lorenz généralisé.

De la même manière que pour la dominance de Pareto, on définit la partie asymétrique de cette relation de dominance par :

$$x \succ_L y \Leftrightarrow (x \succsim_L y \text{ et } non(y \succsim_L x))$$

Le théorème suivant établit le lien entre la dominance de Lorenz généralisée et le transfert de Pigou-Dalton [Chong, 1976] :

Théorème 1.1. *Pour toute paire de solutions $(x, y) \in \mathcal{X}^2$ associées aux vecteurs de performances $u(x)$ et $u(y)$, si $x \succ_P y$ ou si $u(x)$ se déduit de $u(y)$ par un transfert de Pigou-Dalton alors $x \succ_L y$. Inversement si $x \succ_L y$ alors il existe une séquence de transferts de Pigou-Dalton et/ou d'améliorations au sens de Pareto qui permettent de transformer $u(y)$ en $u(x)$.*

Exemple 1.3 (suite). *Reprenons l'exemple précédent avec les deux solutions x et y associées aux vecteurs de performances $u(x) = (5, 5)$ et $u(y) = (3, 6)$. Les vecteurs de Lorenz sont $L(x) = (5, 10)$ et $L(y) = (3, 9)$. On peut facilement voir que $L(x) \succ_P L(y)$; on en déduit alors que $x \succ_L y$ sans avoir à déterminer la séquence de relations de dominance de Pareto et de transferts de Pigou-Dalton.*

Ainsi, la dominance de Lorenz généralisée nous permet d'identifier des solutions susceptibles d'intéresser un décideur recherchant des performances équilibrées. Notons que la relation de dominance de Lorenz généralisée respecte la dominance de Pareto puisque $x \succ_P y \Rightarrow x \succ_L y$ (conséquence du théorème 1.1). Cette relation de dominance est donc, en général, plus discriminante que la dominance de Pareto. Par conséquent, l'ensemble de solutions Lorenz-optimales, défini ci-dessous, est un sous-ensemble de l'ensemble de Pareto.

Définition 1.6. *L'ensemble des solutions* Lorenz-optimales $\mathrm{ND}_L(\mathcal{X})$ *est défini par :*

$$\mathrm{ND}_L(\mathcal{X}) = \{x \in \mathcal{X} | \forall y \in \mathcal{X}, non(y \succ_L x)\}$$

Exemple 1.1 (suite). *Reprenons l'exemple 1.1. Sur la figure 1.3 les solutions Lorenz-optimales sont représentées par des ronds blancs :*

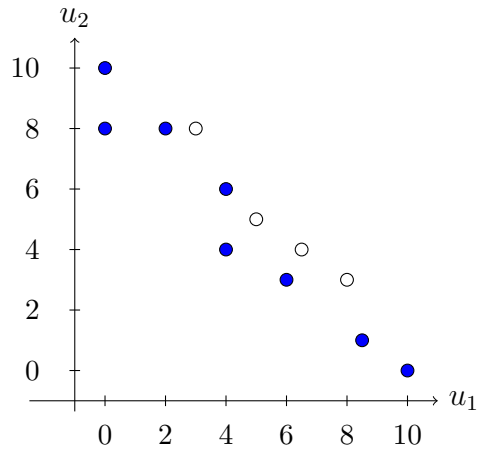


FIGURE 1.3 – Exemple de solutions Lorenz-optimales (carrés blancs).

Comme la dominance de Lorenz est un raffinement de la dominance de Pareto, on peut chercher les solutions Lorenz-optimales parmi celles qui sont Pareto-optimales dans $\mathcal{X}$ (voir la figure 1.2). On peut voir que les vecteurs $(0, 10)$ et $(10, 0)$ sont dominés au sens de Lorenz par $(5, 5)$ car $(0, 0+10) = (0, 10) \succ_P (5, 5+5) = (5, 10)$. Il en est de même pour le vecteur $(8.5, 1)$, qui est également dominé au sens de Lorenz par $(5, 5)$ car $(1, 9.5) \succ_P (5, 10)$.

Notons que même si la dominance de Lorenz est plus discriminante que la dominance de Pareto, elle ne permet tout de même pas de définir un ordre total sur les solutions de $\mathcal{X}$. En effet, cette relation de dominance ne décrit qu'un principe général d'équilibre mais ne permet pas une modélisation personnalisée des préférences du décideur. L'ensemble des solutions Lorenz-optimales peut contenir un très grand nombre de solutions incomparables

au sens de cette relation. La dominance stochastique du second ordre, que nous présentons dans la suite peut être vue comme une extension pondérée de cette règle permettant de caractériser une relation de dominance un peu plus riche. Cette relation offre la possibilité d'accorder plus ou moins d'importance aux différents objectifs en plus de privilégier les solutions ayant un vecteur de performances équilibré.

c. La dominance stochastique du second ordre (Lorenz pondérée)

La dominance de Lorenz généralisée traite symétriquement les composantes d'un vecteur de performances : une relation de préférences de la forme $x \succ_L y$ entre deux solutions x et y n'est pas impactée par une opération de permutation des composantes de $u(x)$ ou de $u(y)$ puisque les éléments de ces vecteurs sont triés (afin de déterminer les vecteurs de Lorenz) avant d'être comparés. Cette symétrie semble naturelle lorsque l'on souhaite accorder la même importance à chaque objectif, mais il peut arriver que le problème traité nécessite d'accorder plus ou moins d'importance à un ou plusieurs objectif(s). Par exemple, en décision dans le risque ou (dans l'incertain), on peut avoir la certitude (ou la conviction) qu'un scénario est bien plus probable qu'un autre. On voudrait alors prendre en compte cette information dans la modélisation des préférences ainsi que dans la résolution du problème en attribuant une probabilité élevée au scénario en question. Une manière d'imposer une telle notion est d'utiliser une extension pondérée de la dominance de Lorenz que l'on définit ici.

Notons tout d'abord $v : 2^N \to [0, 1]$ une fonction d'ensemble qui associe un poids à chaque sous-ensemble d'objectifs. En décision multicritère ou multi-agents, ces poids représentent l'importance accordée à chaque critère/agent ainsi qu'à chaque coalition de critères/agents. Dans le cas d'un problème de décision dans le risque, ce poids représente la probabilité qu'un état (événement) ou qu'un sous-ensemble d'états se produise. On associe ensuite à chaque solution $x \in \mathcal{X}$ une fonction cumulative $F_x(z)$ indiquant le poids de la coalition formée par l'ensemble des objectifs pour lesquels la performance de x ne dépasse pas la valeur z :

$$F_x(z) = v(\{i \in N \mid u_i(x) \leq z\})$$

On définit alors la dominance stochastique du second ordre SSD (pour *Second order Stochastic Dominance*) par :

Définition 1.7. *Soient x et y deux solutions de $\mathcal{X}$. On a :*

$$x \succsim_{\text{SSD}} y \Leftrightarrow \forall z \in \mathbb{R}, F_x^2(z) \leq F_y^2(z)$$

où $F_x^2(z) = \int_{-\infty}^{z} F_x(t)dt$. On dit alors que x domine y au sens de la dominance SSD.

La partie asymétrique de cette relation de dominance se définit par :

$$x \succ_{\text{SSD}} y \Leftrightarrow (x \succsim_{\text{SSD}} y \text{ et } non(y \succsim_{\text{SSD}} x))$$

Notons que l'on parle généralement de dominance *stochastique* lorsque la fonction v représente une distribution de probabilité. Par abus de langage, on utilisera le terme

dominance stochastique qu'il s'agisse de décision multicritère, multi-agents ou de décision dans le risque.

Cette règle de dominance est considérée comme étant une généralisation de la dominance de Lorenz car lorsque v est définie de manière à ce que les objectifs aient tous la même importance, vérifier $x \succ_{\text{SSD}} y$ revient à vérifier la relation de dominance $x \succ_L y$. On peut définir l'ensemble des solutions non-dominées au sens de la règle de dominance SSD :

Définition 1.8. *L'ensemble de solutions* SSD-optimales $\text{ND}_{\text{SSD}}(\mathcal{X})$ *est défini par :*

$$\text{ND}_{\text{SSD}}(\mathcal{X}) = \{x \in \mathcal{X} | \forall y \in \mathcal{X}, non(y \succ_{\text{SSD}} x)\}$$

Exemple 1.1 (suite). *Reprenons toujours le même exemple bi-objectifs. Supposons que l'on accorde plus d'importance au premier objectif et que la fonction v soit définie par $v(\emptyset) = 0, v(\{1\}) = 0.6, v(\{2\}) = 0.4$ et $v(\{1,2\}) = 1$. A titre d'exemple, on donne sur la figure 1.4 les courbes $F_{u^i}(z)$ associées à 4 vecteurs Pareto optimaux : $u^1 = (3,8), u^2 = (5,5), u^3 = (6.5,4)$ et $u^4 = (8,3)$.*

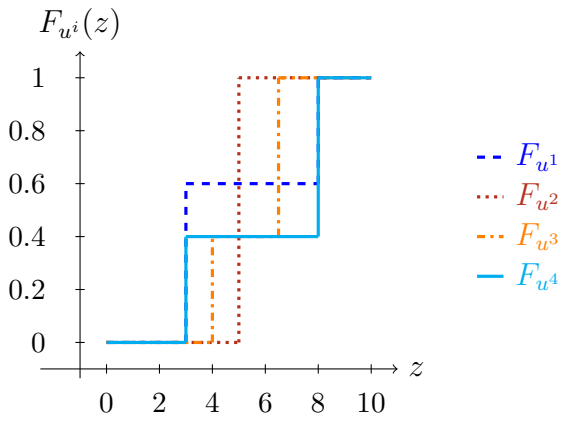


FIGURE 1.4 – Cumulatives $F_{u^i}(z)$.

La relation $x \succ_{\text{SSD}} y$ est vérifiée si l'intégrale de F_x de 0 à z (surface sous la courbe) est inférieure à l'intégrale de F_y de 0 à z (surface sous la courbe) pour toute valeur de z. On observe alors que $(8,3) \succ_{\text{SSD}} (3,8)$ car la courbe $F_{u_1}(z)$ est au-dessus de la courbe $F_{u_4}(z)$ pour toute valeur $z \in \mathbb{R}$. Notons que les vecteurs u_1 et u_4 ont le même vecteur trié ; néanmoins le fait d'accorder plus d'importance au premier objectif donne l'avantage au vecteur $(8,3)$ sur le vecteur $(3,8)$.

La figure 1.5 représente l'ensemble des solutions SSD-optimales par des ronds blancs :

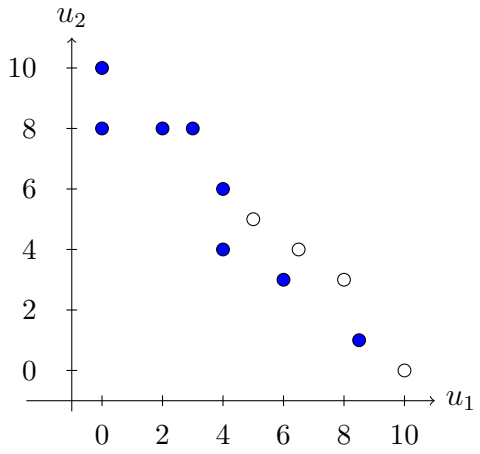


FIGURE 1.5 – Exemple de solutions SSD-optimales pour $v(\{1\}) = 0.6$ et $v(\{2\}) = 0.4$ (ronds blancs).

Notons que la définition de v ne dépend pas toujours des préférences du décideur. Par exemple, lors de la prise de décision dans le risque, plusieurs scénarios sont pris en compte : la fonction v donne alors la probabilité objective que chaque scénario (ou sous-ensemble de scénarios) se produise. Elle peut donc, selon le contexte, être déterminée indépendamment des préférences particulières du décideur.

Nous venons de présenter trois règles de dominance très utilisées lors de la comparaison de solutions évaluées selon plusieurs objectifs. Elles définissent chacune un principe général de rationalité des préférences et/ou d'exigence d'équilibre dans les performances sur les différents objectifs, en incluant ou non une pondération sur ces objectifs. Néanmoins, comme nous avons pu le voir, ces règles sont souvent trop génériques et n'offrent pas la possibilité de personnaliser le comportement décisionnel. Elles induisent un ordre partiel sur les solutions réalisables et, par conséquent, l'ensemble des solutions non-dominées peut être très vaste et contenir un nombre trop important de solutions à comparer pour le décideur. Afin de résoudre ce problème, l'approche la plus communément utilisée en théorie de la décision consiste à raffiner ces relations de dominance en utilisant une fonction d'agrégation f permettant à la fois d'enrichir la modélisation des préférences et de simplifier la formulation du problème de décision.

1.2.2 Agrégation et modélisation des préférences

L'agrégation des préférences est une opération de synthèse de l'information contenue dans les vecteurs de performances des solutions de $\mathcal{X}$, les rendant plus faciles à comparer. Lorsque l'opérateur d'agrégation est bien choisi, il permet de modéliser fidèlement les préférences du décideur et de définir une règle de dominance personnalisée, plus discriminante que la relation de dominance de Pareto, de Lorenz ou de Lorenz pondérée. L'intérêt d'utiliser une telle fonction est de simplifier la formulation et la résolution du problème de décision tout en ayant une bonne structuration des préférences du décideur.

On définit généralement une règle d'agrégation à l'aide d'une fonction $h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ qui associe une valeur réelle à chaque couple (x, y) de solutions possibles. Cette valeur permet de déterminer un ordre de préférence entre x et y sur la base de l'agrégation et

de la comparaison de leur vecteur de performances respectifs. On définit alors une règle de dominance sur la base de cette fonction d'agrégation :

$$x \succsim_h y \Leftrightarrow h(x, y) \geq 0$$

Il existe dans la littérature deux approches distinctes pour définir une règle de dominance en utilisant une fonction d'agrégation :

- **La comparaison par valeurs :** cette approche, également connue sous le nom de l'approche *agréger puis comparer* (*AC*) [Perny, 2000], consiste à associer une valeur scalaire $f(x)$ à chaque solution $x \in \mathcal{X}$ afin d'évaluer sa qualité. La comparaison de deux solutions $x, y \in \mathcal{X}$ revient alors à comparer les valeurs $f(x)$ et $f(y)$. On définit la fonction h par :

  $$h(x, y) = \varphi(f(x), f(y)) \geq 0$$

  où $\varphi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ définit la manière dont on compare $f(x)$ et $f(y)$. Généralement, on définit cette fonction par $\varphi(a, b) = a - b, \forall a, b \in \mathbb{R}$.
  Exemple 1.4. *La règle de dominance suivante entre dans le cadre de l'approche AC :*

  $$x \succsim y \Leftrightarrow \sum_{i=1}^{n} u_i(x) \geq \sum_{i=1}^{n} u_i(y)$$

  Ici $f(x) = \sum_{i=1}^{n} u_i(x), \forall x \in \mathcal{X},$ *et* $\varphi(a, b) = a - b, \forall a, b \in \mathbb{R}$.

  Notons que l'approche AC consiste à combiner les performances d'une solution selon différentes fonctions objectifs. Afin que cette opération ait du sens, elle nécessite la commensurabilité des valeurs de performances. Or, cette hypothèse n'est pas toujours initialement respectée, c'est notamment le cas lorsque les différents objectifs représentent des critères d'évaluation de natures différentes : par exemple, certains critères peuvent être quantitatifs alors que d'autres sont qualitatifs. Dans ce cas, obtenir une nouvelle mesure des performances qui respecte l'hypothèse de commensurabilité peut poser un problème pratique. En effet, l'intérêt que le décideur porte aux différentes alternatives, et selon chaque objectif, n'est pas toujours linéaire. Ainsi, l'interclassement des importances accordées à chaque niveau de performances selon les différents objectifs peut différer d'un décideur à l'autre. Un processus d'interaction entre le décideur et le système de décision doit alors être mis en place afin de ramener les évaluations selon les différents objectifs sur une même échelle, ce qui peut souvent être long et cognitivement éprouvant pour le décideur. Dans cette thèse, on supposera que les valeurs de performances sont commensurables (par définition du problème, ou en supposant que le travail de changement d'échelle a été effectué en amont). Hormis cet inconvénient, l'approche AC offre de nombreux avantages, parmi lesquels :
  - la relation $\succsim_h$ est transitive, ce qui traduit une forme de rationalité dans les préférences que l'on modélise,
  - la relation $\succsim_h$ définit un ordre total sur les solutions de $\mathcal{X}$ puisque f associe à chaque solution $x \in \mathcal{X}$ une valeur dans $\mathbb{R}$ évaluant sa qualité.

  Une telle relation permet donc de reformuler le problème de choix multi-objectifs comme un problème d'optimisation mono-objectif.

- **La comparaison par vecteurs :** cette approche, également appelée approche *comparer puis agréger* (*CA*) [Perny, 2000], consiste à d'abord comparer les vecteurs de performances de x et y (ou une transformation de ces vecteurs) composante par composante, puis à agréger les résultats de ces comparaisons. On écrit :

$$h(x,y) = f(\varphi_1(g_1(x), g_1(y)), \ldots, \varphi_n(g_n(x), g_n(y))) \geq 0$$

où $\varphi_i : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ désigne le i^e indice de préférence entre x et y, et où $g_i(x)$ désigne une performance agrégée partielle calculée à partir de $u(x)$. La fonction $g(x) = (g_1(x), \ldots, g_n(x))$ peut être la fonction identité, une opération de permutation ou toute autre opération fondée sur un calcul impliquant uniquement les composantes de $u(x)$ (voir les deux exemples suivants).
Ici, l'idée n'est pas (toujours) de comparer l'intensité de la différence entre les performances individuelles de x et de y mais (souvent) de déterminer l'ordre de préférence entre ces performances, comme le montre l'exemple 1.5 ci-dessous. Ainsi, l'information nécessaire sur le système de valeurs du décideur est beaucoup moins précise que dans l'approche AC, ce qui a pour avantage d'imposer au décideur un effort cognitif moins important lors de l'élicitation du modèle de préférence.

Exemple 1.5. *La règle de dominance suivante entre dans le cadre de l'approche CA :*

$$x \succsim y \Leftrightarrow \sum_{i=1}^{n} \varphi_i(u_i(x), u_i(y)) \geq 0$$

avec pour tout $i \in N$ et pour tout $a, b \in \mathbb{R}$:

$$\varphi_i(a,b) = \begin{cases} 1 & si\ a > b \\ 0 & si\ a = b \\ -1 & sinon \end{cases}$$

Ici, les fonctions $g_i, i \in N$ et f sont définies par :

$$g_i(x) = u_i(x), \qquad et \qquad \forall c \in \mathbb{R}^n, f(c_1, \ldots, c_n) = \sum_{i=1}^{n} c_i$$

Exemple 1.6. *Les dominances de Pareto, de Lorenz et de Lorenz pondérée entrent dans le cadre de l'approche CA. Par exemple, la dominance de Pareto peut être définie de manière similaire à la règle de l'exemple 1.5 en remplaçant la définition de f par :*

$$f(c_1, \ldots, c_n) = \sum_{i=1}^{n} c_i - n$$

Il en est de même pour la dominance de Lorenz en utilisant la même fonction f et en définissant g_i, pour tout $i \in \{1, \ldots, n\}$, par :

$$g_i(x) = \sum_{j=1}^{i} u_{(j)}(x)$$

Souvent, la fonction φ_i est utilisée pour comparer les performances de deux solutions sur un même critère (comparer $u_i(x)$ et $u_i(y)$). Dans ce cas, cette approche

ne nécessite pas que les valeurs des performances selon les différentes fonctions objectifs soient commensurables. On peut alors l'utiliser directement lorsque l'évaluation des solutions porte, par exemple, sur des critères qui ne sont pas de même nature (certains quantitatifs et d'autres qualitatifs). Il arrive tout de même, dans certains cas, que la fonction φ_i soit utilisée pour comparer les performances de x et de y sur des objectifs différents, comme dans le cas de la dominance de Lorenz par exemple (cf. exemple 1.6 ci-dessus). Dans ce cas, la commensurabilité des valeurs de performances est évidemment nécessaire.

Dans tous les cas, une telle fonction ne garantit pas toujours la transitivité de la relation de préférences $\succsim$ (par exemple, la relation de l'exemple 1.5 n'est pas transitive), ce qui peut poser problème lorsque l'on souhaite déterminer une solution préférée. De plus, cette approche ne définit pas toujours une règle d'agrégation induisant un ordre total sur les solutions de $\mathcal{X}$, ce qui peut également poser problème lorsque l'on tente de résoudre une problématique de choix.

Le choix entre AC et CA dépend principalement du type de comportement décisionnel que l'on souhaite modéliser, des informations préférentielles dont on dispose, mais également du type de problème que l'on souhaite résoudre. Pour résumer, comme nous avons pu le voir plus haut, AC force la commensurabilité des valeurs de performances, la complétude et la transitivité des préférences, contrairement à CA qui ne force aucune des trois propriétés. Notons que ces propriétés nécessitent souvent la présence d'une information très riche sur le système de valeurs du décideur. En particulier, les valeurs $u_i(x), \forall i \in N$. Celles-ci doivent être déterminées de manière précise pour AC, alors que l'approche CA nécessite souvent uniquement une information ordinale entre deux valeurs $u_i(x), u_j(x), i \neq j$.

Dans le cadre de ce travail de thèse, on se place dans une problématique de choix. Il est donc nécessaire de disposer d'une règle d'agrégation transitive qui définit un ordre total sur les solutions de $\mathcal{X}$ afin de garantir l'existence d'une solution optimale. Nous nous focaliserons donc sur l'approche AC. Le problème de décision défini par 1.1 revient à un problème d'optimisation de la forme :

$$\begin{cases} \max f(x) \\ x \in \mathcal{X} \end{cases} \tag{1.2}$$

Dans la suite du document, nous supposons que l'on dispose des valeurs précises des performances $u_i(x), \forall i \in N$ et que les valeurs $u_i(x), u_j(x), \forall i \neq j$, sont commensurables (voir le paragraphe de la section 1.3 concernant l'élicitation des valeurs u_i).

On se pose maintenant la question de la définition formelle de la fonction f. Celle-ci dépend entièrement de la nature et de la complexité du comportement décisionnel que l'on souhaite modéliser. De nombreux modèles décisionnels plus ou moins sophistiqués ont été proposés dans la littérature. Ces modèles vérifient des propriétés axiomatiquement garantissant une forme de rationalité et une bonne modélisation des préférences. On mentionne ici certaines propriétés intéressantes que l'on souhaite souvent retrouver dans un modèle décisionnel :

- **idempotence** : une fonction $f : \mathbb{R}^p \to \mathbb{R}$ est idempotente si est seulement si pour tout $x \in \mathbb{R}^p$ on a :

$$\forall i \in \{1, \ldots, n\}, u_i(x) = r \ \Rightarrow f(x) = r$$

Cette propriété traduit une idée assez intuitive qui consiste à dire que lorsque les performances d'une solution sont les mêmes pour tous les objectifs (performances équivalentes sur tous les critères, unanimité des agents, ou performance inchangée quel que soit le scénario considéré) alors la valeur agrégée du vecteur de performances doit être égale à la performance sur chaque objectif.

- **monotonie** : une fonction $f : \mathbb{R}^p \to \mathbb{R}$ est monotone par rapport à une règle de dominance R si et seulement si pour tout couple $x, y \in \mathbb{R}^p$:

$$x \succ_R y \Rightarrow f(x) > f(y)$$

Cette propriété signifie que les préférences représentées par f doivent refléter la notion de préférence définie par R. Par exemple, on souhaite que le modèle utilisé respecte le principe de rationalité défini par la règle de dominance de Pareto. On veut alors que la fonction f soit monotone par rapport à la dominance de Pareto. Si l'on souhaite modéliser une préférence pour les solutions ayant un vecteur de performances équilibré, on peut imposer que f soit monotone par rapport à la dominance de Lorenz.

- **symétrie** : une fonction $f : \mathbb{R}^p \to \mathbb{R}$ est symétrique si et seulement si pour tout couple $x, y \in \mathbb{R}^p$:

$$u(y) = \sigma(u(x)) \ \Rightarrow f(x) = f(y)$$

où $\sigma(u(x))$ est une permutation des composantes du vecteur de performances $u(x)$. Cette propriété traduit le fait que le décideur accorde la même importance aux différents objectifs considérés. Elle est pertinente lorsque l'on résout un problème d'optimisation multi-agents ou d'optimisation dans l'incertain lorsque les agents/scénarios ont tous la même importance.

Le choix de la fonction d'agrégation à utiliser pour modéliser les préférences du décideur dépendra donc des propriétés souhaitées en fonction du comportement décisionnel du décideur. De manière générale, la monotonicité par rapport à Pareto est toujours exigée. On donne maintenant quelques exemples de modèles décisionnels issus de la littérature.

a. Quelques exemples de fonctions d'agrégation

Une approche simple et fréquemment employée en recherche opérationnelle et en intelligence artificielle consiste à optimiser la performance moyenne, i.e., à poser

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} u_i(x)$$

C'est ce que l'on appelle en théorie de la décision le critère *utilitariste*, car il favorise la performance globale des objectifs. Ce critère ne convient cependant pas à tout décideur et à tout problème de décision. En effet, une solution moyenne-optimale peut s'avérer être de très mauvaise qualité dans certains cas. Par exemple, elle ne conviendra pas lorsque le décideur privilégie les solutions ayant un vecteur de performances équilibré, à cause de l'effet de compensation entre les différentes performances. Dans ce cas particulier, une option plus adaptée consiste à choisir la solution maximisant la composante minimum du vecteur de performances :

$$f(x) = \min_{i \in N} u_i(x)$$

Ce critère, que l'on appelle le critère *égalitariste*, traduit la notion d'égalité comme le fait de maximiser la pire performance. Même si elle peut sembler être relativement intuitive, cette approche est considérée comme étant particulièrement pessimiste car elle réduit tout vecteur de performances à sa plus mauvaise composante, ce qui ne reflète pas toujours un comportement décisionnel réel. De plus, elle est peu discriminante puisqu'elle ne permet pas de tenir compte des performances autres que la performance minimum, et ne permet pas de séparer d'éventuels ex-æquo (sur leur pire performance). Ainsi, deux solutions considérées comme équivalentes au sens de ce critère peuvent être telles que l'une Pareto-domine l'autre, ce qui peut être contre-intuitif. On voudrait alors pouvoir disposer d'un opérateur d'agrégation plus discriminant que la moyenne et le min et reflétant des comportements décisionnels réels. De très nombreux modèles plus ou moins complexes ont été proposés pour modéliser différents types de comportements : priorités sur les objectifs, exigence d'équilibre dans le vecteur de performances, interactions entre objectifs, et une toute une gamme de nuances entre ces comportements. Pour permettre une telle flexibilité, ces modèles sont souvent paramétrés par des vecteurs ou des fonctions de pondération permettant de modéliser un large éventail de comportements décisionnels. Nous décrivons ci-dessous quatre modèles largement étudiés en théorie de la décision : la somme pondérée, la somme pondérée ordonnée, la somme doublement pondérée ordonnée, et enfin l'intégrale de Choquet. Ces fonctions d'agrégation sont toutes entièrement caractérisées par leurs paramètres qui, en variant, permettent d'aboutir à des solutions Pareto-optimales différentes.

Somme pondérée (WS). La somme pondérée (WS pour *Weighted Sum*) est une fonction d'agrégation très simple. Elle est paramétrée par un vecteur de pondération w défini dans $W = \{w \in \mathbb{R}_+^n | \sum_{i=1}^n w_i = 1\}$, qui permet d'associer une importance particulière à chaque objectif. Elle se définit par :

Définition 1.9. *Étant donné un vecteur de pondération $w \in W$, pour toute solution $x \in \mathcal{X}$ de vecteur de performances $u(x) \in \mathcal{Y}$, on a :*

$$\mathrm{WS}_w(x) = \sum_{i=1}^{n} w_i u_i(x)$$

Il est facile de voir que la somme pondérée est un opérateur *idempotent* et *monotone* par rapport à la dominance de Pareto lorsque tous les poids $w_i, i \in N$, sont strictement positifs.

La relation de préférence induite par la somme pondérée est donnée par :

$$x \succsim y \Leftrightarrow \mathrm{WS}_w(x) \geq \mathrm{WS}_w(y)$$

On peut donc utiliser cet opérateur lorsque l'on sait que le décideur accorde une importance différente à chaque objectif. En faisant varier le vecteur w dans W, on peut privilégier différents objectifs avec une grande variété d'intensités différentes. Le problème de décision initial est alors ramené à un problème d'optimisation de la forme du programme donné par le système (1.2) en remplaçant $f(x)$ par $\mathrm{WS}_w(x)$.

Cet opérateur est utilisé dans diverses situations du fait de sa simplicité pratique. Par exemple, lors du calcul de moyennes de notes avec des coefficients attribués à chaque note,

ou encore le calcul d'espérances mathématiques en théorie des probabilités. Néanmoins, son expressivité est relativement limitée, par exemple, on ne peut pas modéliser une préférence pour l'équité avec une somme pondérée. Ceci est dû au fait que, selon la forme de l'espace des solutions, toute solution de Pareto ne peut pas toujours être atteinte en optimisant une somme pondérée, et ce, quel que soit le vecteur de pondération considéré. C'est pourquoi on peut s'intéresser à des agrégateurs plus sophistiqués étant capables de refléter des comportements décisionnels plus complexes.

Somme pondérée ordonnée (OWA). La somme pondérée ordonnée ou (OWA pour *Ordered Weighted Average*), introduite par Yager [1998], est une fonction d'agrégation dépendante du rang : elle est paramétrée par un vecteur de pondération $w \in W$ qui permet d'associer une importance particulière à chaque composante du vecteur de performances en fonction de son rang. Cet agrégateur peut être vu comme une somme pondérée où le poids w_i n'est pas toujours associé à la composante u_i du vecteur de performances mais plutôt à la composante $u_{(i)}$ du vecteur de performances *trié* dans l'ordre croissant [1]. Plus formellement, on définit cet agrégateur par :

Définition 1.10. *Étant donné un vecteur de pondération $w \in W$, pour toute solution $x \in \mathcal{X}$ de vecteur de performances $u(x) \in \mathcal{Y}$, on a :*

$$\mathrm{OWA}_w(x) = \sum_{i=1}^{n} w_i u_{(i)}(x)$$

où $u_{(i)}(x)$ est la i^e plus petite composante du vecteur $u(x)$.

La relation de préférence induite par OWA est donnée par :

$$x \succsim y \Leftrightarrow \mathrm{OWA}_w(x) \geq \mathrm{OWA}_w(y)$$

Exemple 1.7. *Supposons que l'on souhaite comparer deux solutions x et y d'utilités respectives $u(x) = (10, 12, 10)$ et $u(y) = (12, 7, 19)$. Supposons que l'on évalue ces solutions avec un* OWA *de paramètre $w = (1/2, 1/6, 1/3)$. On a :*

$$\mathrm{OWA}_w(x) = \frac{1}{2} \times 10 + \frac{1}{6} \times 10 + \frac{1}{3} \times 12 = 32/3$$
$$\mathrm{OWA}_w(y) = \frac{1}{2} \times 7 + \frac{1}{6} \times 12 + \frac{1}{3} \times 19 = 71/6$$

on en déduit que $y \succsim x$.

Le problème de décision initial se reformule alors comme le problème d'optimisation suivant :

$$(P_{\mathrm{OWA}}) : \begin{cases} \max \ \mathrm{OWA}_w(x) \\ x \in \mathcal{X} \end{cases} \tag{1.3}$$

Le vecteur de pondération w permet de contrôler le type de comportement décisionnel que l'on modélise et donc de contrôler le type de solutions optimales que l'on obtient.

1. Ce tri correspond en fait au tri des composantes du vecteur de performances par qualité croissante : de la moins bonne à la meilleure performance. Dans un problème de minimisation, on triera le vecteur de performances dans l'ordre décroissant.

En accordant plus ou moins d'importance aux bonnes et aux mauvaises performances, il traduit en effet différents comportements décisionnels plus ou moins optimistes. Il permet alors de définir toute une classe d'agrégateurs contenant, entre autres, les opérateurs classiques tels que le *min* avec $w = (1, 0, \ldots, 0)$, le *max* avec $w = (0, \ldots, 0, 1)$, la *médiane* avec $w = (\underbrace{0, \ldots, 0}_{\lfloor \frac{n}{2} \rfloor}, 1, \underbrace{0, \ldots, 0}_{\lfloor \frac{n}{2} \rfloor}$ pour n impair et $w = (\underbrace{0, \ldots, 0}_{\frac{n}{2}-1}, 1, \underbrace{0, \ldots, 0}_{\frac{n}{2}-1}$ pour n pair, et la *moyenne* avec $w = (\frac{1}{n}, \ldots, \frac{1}{n})$. Avec un choix de paramètres adapté, on peut également traduire des degrés, plus ou moins importants, d'exigence d'équilibre dans le vecteur de performances de la solution optimale : pour privilégier une solution équilibrée, on choisit un vecteur poids à composantes décroissantes de manière à surpondérer les mauvaises performances. On peut effectivement voir qu'avec un tel vecteur de pondération, $\mathrm{OWA}_w(x)$ n'est autre qu'une combinaison linéaire des composantes de Lorenz. En effet, on a $u_{(1)}(x) = L_1(x)$ et $u_{(i)}(x) = L_i(x) - L_{i-1}(x)$ pour tout $i \in \{2, \ldots, n\}$; on peut alors réécrire $\mathrm{OWA}_w(x)$ de la manière suivante :

$$\mathrm{OWA}_w(x) = \sum_{i=1}^{n} w'_i L_i(x) \tag{1.4}$$

avec $w'_i = w_i - w_{i+1}, i \in \{2, \ldots, n\}$ et $w'_n = w_n$. L'opérateur OWA_w est donc dans ce cas monotone par rapport à la dominance de Lorenz. En faisant varier w dans $W = \{w \in \mathbb{R}^n_+ | \sum_{i=1}^n w_i = 1, w_1 \geq \cdots \geq w_n\}$, on peut définir une infinité de relations de préférences, compatibles avec la dominance de Lorenz, allant du critère égalitariste avec $w = (1, 0, \ldots, 0)$ au critère utilitariste avec $w = (\frac{1}{n}, \ldots, \frac{1}{n})$.

Exemple 1.7 (suite) *Reprenons l'exemple précédent en ajoutant la solution z d'utilité $u(z) = (12, 9, 16)$. Supposons dans un premier temps que l'on évalue les solutions x, y et z par le critère égalitariste. On obtient alors :*

$$\begin{aligned}
\mathrm{OWA}_w(x) &= 1 \times 10 + 0 \times 10 + 0 \times 12 = 10 \\
\mathrm{OWA}_w(y) &= 1 \times 7 + 0 \times 12 + 0 \times 19 = 7 \\
\mathrm{OWA}_w(z) &= 1 \times 9 + 0 \times 12 + 0 \times 16 = 9
\end{aligned}$$

autrement dit $x \succsim z \succsim y$. On peut voir que la préférence entre x et y est inversée par rapport à l'exemple précédent où w n'est pas à composantes décroissantes. Remarquons en effet que $u(x)$ est bien plus équilibré que $u(y)$.

Évaluons-les maintenant à l'aide du critère utilitariste :

$$\begin{aligned}
\mathrm{OWA}_w(x) &= \frac{1}{3}(10 + 10 + 12) = 32/3 \\
\mathrm{OWA}_w(y) &= \frac{1}{3}(7 + 12 + 19) = 38/3 \\
\mathrm{OWA}_w(z) &= \frac{1}{3}(9 + 12 + 16) = 37/3
\end{aligned}$$

on obtient cette fois $y \succsim z \succsim x$ ce qui est assez naturel puisque z est très performante globalement et que l'équilibre de $u(x)$ est assuré au détriment de la performance globale.

Pour modéliser un compromis entre ces deux critères, on peut par exemple utiliser le

vecteur de pondération $w = (1/2, 1/3, 1/6)$. *On a alors :*

$$\begin{aligned}
\mathrm{OWA}_w(x) &= \frac{1}{2} \times 10 + \frac{1}{3} \times 10 + \frac{1}{6} \times 12 = 31/3 \approx 10.33 \\
\mathrm{OWA}_w(y) &= \frac{1}{2} \times 7 + \frac{1}{3} \times 12 + \frac{1}{6} \times 19 = 32/3 \approx 10.67 \\
\mathrm{OWA}_w(z) &= \frac{1}{2} \times 9 + \frac{1}{3} \times 12 + \frac{1}{6} \times 16 = 67/6 \approx 11.17
\end{aligned}$$

on obtient maintenant $z \succsim y \succsim x$. *On peut effectivement voir que* $u(z)$ *offre un bon compromis entre équilibre des performances et performance moyenne.*

Cet exemple illustre bien le fait que, non seulement l'opérateur OWA permet de privilégier les solutions aux vecteurs de performances équilibrées, mais également de modéliser différents comportements offrant ainsi différents compromis entre le souci d'équité et celui d'efficience globale. On utilise donc souvent l'opérateur OWA pour la modélisation de comportements privilégiant, avec plus ou moins de force, un équilibre des performances sur les différents objectifs.

Pour finir, il est facile de voir que l'opérateur OWA est *idempotent*, *monotone* par rapport à Lorenz lorsque w est à composantes décroissantes, et également *symétrique* puisque l'on applique la pondération au vecteur de performances *trié* et non au vecteur de performances en lui même. Cette dernière propriété est naturelle lorsque la même importance est accordée à chaque objectif. Par exemple, dans certains problèmes de décision multi-agents, il semble naturel de considérer que l'on accorde la même importance au point de vue de chaque agent dans la prise de décision collective. Néanmoins, il peut arriver que l'on se retrouve dans une situation où les objectifs n'ont pas la même importance. Par exemple, en décision dans le risque, lorsque les scénarios que l'on considère ont des probabilités différentes, ou en décision multi-agents lorsque les agents ont des droits exogènes. Dans ce cas, l'opérateur OWA ne convient pas car il ne permet pas de prendre en compte cette information. Il devient alors nécessaire d'utiliser un modèle disposant d'un pouvoir descriptif plus riche permettant de différencier l'importance des différents objectifs. C'est pourquoi nous nous intéressons dans la suite à deux généralisations de cet opérateur : WOWA et l'intégrale de Choquet.

Somme pondérée doublement ordonnée (WOWA). Cette formulation est connue dans la littérature sous le nom de *modèle de Yaari* car elle a été introduite par Yaari [1987] dans un contexte de décision dans le risque. Le modèle s'applique aussi à la décision multi-agents et multicritère. On le définit par :

Définition 1.11. *Étant donnés un vecteur de pondération/probabilités* $(p_1, .., p_n)$ *et une fonction de déformation* $\varphi : [0, 1] \to [0, 1]$ *croissante telle que* $\varphi(0) = 0$ *et* $\varphi(1) = 1$, *pour toute solution* $x \in \mathcal{X}$ *de vecteur de performances* $u(x) \in \mathcal{Y}$, *on a :*

$$\begin{aligned}
\mathrm{WOWA}_\varphi(x) &= \sum_{i=1}^{n} [u_{(i)}(x) - u_{(i-1)}(x)] \varphi(\sum_{k=i}^{n} p_{(k)}) && (1.5) \\
&= \sum_{i=1}^{n} [\varphi(\sum_{k=i}^{n} p_{(k)}) - \varphi(\sum_{k=i+1}^{n} p_{(k)})] u_{(i)}(x) && (1.6)
\end{aligned}$$

où $u_{(0)}(x) = 0$, $u_{(i)}(x)$ est la i^e plus petite composante du vecteur de performances $u(x)$, et $p_{(i)}$ désigne la i^e composante du vecteur $(p_1, .., p_n)$ réordonné selon la permutation qui permet de trier $u(x)$ dans l'ordre croissant des valeurs de performances.

La dénomination de WOWA est due à l'importation de ce modèle en décision multi-critère par Torra [1997]. Il propose une formulation similaire en partant d'un OWA mais en utilisant deux jeux de poids différents : p qui donne l'importance ou la probabilité de chaque objectif, et le w (d'un OWA) permettant de contrôler le poids associé aux bonnes et aux mauvaises performances. Il définit alors la fonction de déformation φ, sur un sous-ensemble de points, à l'aide de w par $\varphi(i/n) = \sum_{k=1}^{i} w_{n-k+1}$ obtenant ainsi un modèle doublement pondéré, d'où le nom de *Weighted* OWA. Notons que les valeurs $\varphi(r)$ avec $r \in [0, 1]$ tel que $r \neq i/n, i \in N$, sont obtenues par interpolation linéaire. Dans un contexte de décision dans le risque, cet opérateur est également connu sous le nom de RDU (Rank Dependent Utility), et peut être vu comme une généralisation du modèle EU (Expected Utility) [Quiggin, 1993]. Alors que le modèle EU consiste à évaluer chaque solution par une utilité espérée qui attribue une probabilité à l'utilité g_i de chaque performance $u_i(x), i \in N$, i.e., $EU(x) = \sum_{i=1}^{n} p_i g_i(u_i(x))$, RDU consiste à définir le coefficient de $g_{(i)}$ en fonction des probabilités cumulées des scénarios j tels que $g_j \geq g_{(i)}$, i.e., $\text{RDU}_\varphi(x) = \sum_{i=1}^{n} [\varphi(\sum_{j=i}^{n} p_{(j)}) - \varphi(\sum_{j=i+1}^{n} p_{(j)})] g_{(i)}(u(x))$.

De même que pour les agrégateurs précédents, cet agrégateur induit une règle de dominance :

$$x \succsim y \Leftrightarrow \text{WOWA}_\varphi(x) \geq \text{WOWA}_\varphi(y)$$

Exemple 1.7 (suite) *Reprenons l'exemple précédent avec $u(x) = (10, 12, 10), u(y) = (12, 7, 19)$ et $u(z) = (12, 9, 16)$. Prenons $p = (1/2, 1/6, 1/3)$ et $\varphi(r) = r^2, \forall r \in [0, 1]$. On a alors :*

$$\text{WOWA}_\varphi(x) = 10 + (10 - 10)\varphi(1/3 + 1/6) + (12 - 10)\varphi(1/6) = \frac{181}{18} \approx 10.05$$
$$\text{WOWA}_\varphi(y) = 7 + (12 - 7)\varphi(1/2 + 1/3) + (19 - 12)\varphi(1/3) = \frac{45}{4} = 11.25$$
$$\text{WOWA}_\varphi(z) = 9 + (12 - 9)\varphi(1/2 + 1/3) + (16 - 12)\varphi(1/3) = \frac{83}{7} \approx 11.53$$

On en déduit la préférence $z \succsim y \succsim x$.

Le problème de décision initial se reformule ici comme le problème d'optimisation suivant :

$$(P_{\text{WOWA}}) : \begin{cases} \max \ \text{WOWA}_w(x) \\ x \in \mathcal{X} \end{cases} \tag{1.7}$$

Notons que lorsque les objectifs ont tous la même importance, c'est-à-dire que p est uniforme ($p_i = 1/n, \forall i$), alors la formulation de l'équation (1.6) devient :

$$\sum_{i=1}^{n} [\varphi(\frac{n-i+1}{n}) - \varphi(\frac{n-i}{n})] u_{(i)}(x)$$

le coefficient de $u_{(i)}(x)$ ne dépend maintenant que de l'indice i et non du vecteur $u(x)$. En posant $w_i = \varphi(\frac{n-i+1}{n}) - \varphi(\frac{n-i}{n})$, on voit que l'on retombe sur la formule de l'OWA. Dans ce cas, le modèle de Yaari peut être vu comme une extension pondérée d'OWA.

La fonction de pondération φ, aussi appelée fonction de déformation, permet de contrôler le type de comportement décisionnel que l'on modélise et donc de contrôler le type de solutions optimales que l'on obtient en résolvant (P_{WOWA}). En déformant plus ou moins les poids (ou probabilités) cumulés, l'opérateur WOWA traduit des comportements décisionnels plus ou moins optimistes. Il contient alors toute une classe d'agrégateurs, dont ceux cités plus haut pour OWA, puisque OWA est un cas particulier de WOWA. Cet opérateur permet de modéliser des préférences plus riches qu'OWA puisqu'elle permet, en choisissant une fonction adaptée, de privilégier plus ou moins l'équilibre dans la répartition des valeurs de performances, tout en permettant de prendre en compte l'importance relative des différents objectifs. En effet, il a été établi par Hong *et al.* [1987], ainsi que par Yaari [1987] que si la fonction φ est convexe alors la relation de préférence induite par WOWA est monotone avec la relation de dominance SSD. En faisant varier φ dans l'ensemble des fonctions positives, strictement croissantes et telles que $\varphi(0) = 0, \varphi(1) = 1$ et $\varphi(r) \leq r, \forall r \in [0,1]$, on peut définir un large éventail de comportements décisionnels compatibles avec la dominance SSD. Notons que lorsque l'égalité est vérifiée, i.e., $\varphi(r) = r, \forall r \in [0,1]$, on ne traduit pas d'exigence d'équilibre dans les valeurs de performances, seule l'information concernant l'importance de chaque objectif est prise en compte. Dans ce cas, il est facile de voir que $\text{WOWA}_\varphi(x) = \text{WS}_p(x)$.

Exemple 1.7 (suite) *Reprenons l'exemple précédent avec* $u(x) = (10, 12, 10), u(y) = (12, 7, 19)$ *et* $u(z) = (12, 9, 16)$. *Prenons* $p = (1/6, 1/3, 1/2)$ *et évaluons les solutions à l'aide de la somme pondérée, autrement dit avec* $\varphi(r) = r$ *:*

$$\text{WOWA}_\varphi(x) = 10 + (10-10)\varphi(1/2+1/3) + (12-10)\varphi(1/3) = \frac{32}{3} \approx 10.67$$
$$\text{WOWA}_\varphi(y) = 7 + (12-7)\varphi(1/6+1/2) + (19-12)\varphi(1/2) = \frac{83}{6} \approx 13.83$$
$$\text{WOWA}_\varphi(z) = 9 + (12-9)\varphi(1/6+1/2) + (16-12)\varphi(1/2) = 13$$

On en déduit la préférence $y \succsim z \succsim x$.

Évaluons maintenant x, y *et* z *avec le critère égalitariste, autrement dit avec* $\varphi(r) = 0, \forall r \in [0,1)$, *et* $\varphi(1) = 1$. *On a alors :*

$$\text{WOWA}_\varphi(x) = 10 + (10-10) \times 0 + (12-10) \times 0 = 10$$
$$\text{WOWA}_\varphi(y) = 7 + (12-7) \times 0 + (19-12) \times 0 = 7$$
$$\text{WOWA}_\varphi(z) = 9 + (12-9) \times 0 + (16-12) \times 0 = 9$$

On en déduit la préférence $x \succsim z \succsim y$. *Ici la pondération donnée par* p *n'est pas prise en compte, seule l'équilibre des performances compte (l'objectif est de maximiser la pire performance).*

On peut maintenant combiner équilibre et prise en compte de la pondération p *en posant par exemple* $\varphi(r) = r^2$ *:*

$$\text{WOWA}_\varphi(x) = 10 + (10-10)\varphi(1/2+1/3) + (12-10)\varphi(1/3) = \frac{92}{9} \approx 10.22$$
$$\text{WOWA}_\varphi(y) = 7 + (12-7)\varphi(1/6+1/2) + (19-12)\varphi(1/2) = \frac{395}{36} \approx 10.97$$
$$\text{WOWA}_\varphi(z) = 9 + (12-9)\varphi(1/6+1/2) + (16-12)\varphi(1/2) = \frac{34}{3} = \approx 11.33$$

On en déduit cette fois la préférence $z \succsim y \succsim x$.

Cet exemple illustre bien le pouvoir descriptif de WOWA : nous avons vu que cet opérateur permet de définir différents degrés de compromis entre l'équilibre dans la répartition des performances et la prise en compte de l'importance accordée à chaque objectif.

Pour finir, l'opérateur WOWA est *idempotent* et *monotone* par rapport à la règle de dominance SSD. Il permet donc de modéliser un grand panel de comportements décisionnels, et possède un pouvoir descriptif très riche. Néanmoins, certains types de préférences restent impossibles à modéliser avec cet agrégateur, comme par exemple la notion de coalition entre objectifs. D'où l'intérêt pour l'opérateur suivant.

Intégrale de Choquet. L'intégrale de Choquet [Choquet, 1954; Schmeidler, 1986; Grabisch *et al.*, 1996] est un opérateur d'agrégation encore plus général que WOWA, et qui permet de modéliser des comportements décisionnels relativement complexes. En plus de permettre de modéliser une préférence pour des solutions ayant un vecteur de performances équilibré (il contient en particulier OWA) et différents niveaux d'importance pour les différents objectifs (il contient en particulier WOWA), il permet de modéliser des interactions positives ou négatives entre objectifs. L'intégrale de Choquet est paramétrée par une fonction d'ensemble v que l'on appelle *capacité* et qui permet d'associer un poids à chaque objectif et à chaque coalition d'objectifs possible :

Définition 1.12. *Une capacité* v *est une fonction d'ensemble définie de* 2^N *dans* $[0,1]$ *et telle que :*
- $v(\emptyset) = 0$,
- $v(N) = 1$,
- $v(A) \leq v(B)$ *pour tout* $A \subseteq B \subseteq N$.

On définit alors l'intégrale de Choquet d'une solution par :

Définition 1.13. *Étant donnée une capacité* v*, pour toute solution* $x \in \mathcal{X}$ *de vecteur de performances* $u(x) \in \mathcal{Y}$*, on a :*

$$C_v(x) = \sum_{i=1}^{n} (v(X_{(i)}) - v(X_{(i+1)}))u_{(i)}(x) \tag{1.8}$$

$$= \sum_{i=1}^{n} [u_{(i)}(x) - u_{(i-1)}(x)]v(X_{(i)}) \tag{1.9}$$

où $u_{(i)}(x)$ *est la* i^e *plus petite composante du vecteur* $u(x)$ *et* $X_{(i)}$ *contient l'ensemble des objectifs* $k \in N$*, pour lesquels* $u_k(x) \geq u_{(i)}(x)$.

On peut facilement voir, grâce à l'équation (1.8), que WS, OWA et WOWA sont des cas particuliers de cet opérateur. En effet, si la capacité est additive, i.e., $\forall A \subseteq N, v(A) = \sum_{i \in A} p_i$, on retrouve la formulation de la somme pondérée. En prenant une capacité symétrique, i.e., $\forall A \subseteq N, v(A) = v(B)$ pour tout $B \subseteq N$ tel que $|B| = |A|$, alors le coefficient de $u_{(i)}(x)$ dans (1.8) ne dépend pas des éléments de $X_{(i)}$ mais uniquement de sa taille (donc de l'indice i). En posant $w_i = v(X_{(i)}) - v(X_{(i+1)})$ on retrouve la formulation d'un OWA. Finalement, en prenant une capacité telle que $v(A) = \varphi(\sum_{i \in A} p_i)$ pour une fonction φ donnée, alors le calcul de l'intégrale de Choquet revient à calculer $\text{WOWA}_\varphi(x)$.

Le problème de décision traité se reformule comme le problème d'optimisation suivant : d'optimisation suivant :

$$(P_C) : \begin{cases} \max\ C_v(x) \\ x \in \mathcal{X} \end{cases} \tag{1.10}$$

Exemple 1.7 (suite) *Reprenons une dernière fois l'exemple précédent avec* $u(x) = (10, 12, 10), u(y) = (12, 7, 19)$ *et* $u(z) = (12, 9, 16)$. *Supposons que la capacité* v *soit définie par :*

X	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$
$v(X)$	0.1	0.2	0.3	0.5	0.5	0.5	1

On a alors :

$$C_v(x) = 10 + (10 - 10)v(\{2,3\}) + (12 - 10)v(\{2\}) = 10 + 2 \times 0.2 = 10.4$$
$$C_v(y) = 7 + (12 - 7)v(\{1,3\}) + (19 - 12)v(\{3\}) = 7 + 5 \times 0.5 + 7 \times 0.3 = 11.6$$
$$C_v(z) = 9 + (12 - 9)v(\{1,3\}) + (16 - 12)v(\{3\}) = 9 + 3 \times 0.5 + 4 \times 0.3 = 11.7$$

On en déduit la préférence $z \succsim y \succsim x$ *et donc que la solution préférée est la solution* z.

Dans le cas de l'intégrale de Choquet, le paramètre est donné par la capacité v qui permet de contrôler le comportement décisionnel modélisé par cet agrégateur. Lorsque cette capacité est convexe, i.e., telle que :

$$\forall A, B \subseteq N, v(A \cup B) + v(A \cap B) \geq v(A) + v(B) \tag{1.11}$$

alors l'intégrale de Choquet permet de privilégier les solutions ayant un vecteur de performances équilibré [Chateauneuf et Tallon, 2002; Galand *et al.*, 2010]. Au delà de cet aspect, l'avantage principal de l'utilisation d'une intégrale de Choquet est sa capacité à modéliser différentes interactions entre les objectifs [Grabisch et Labreuche, 2010] :

- si la capacité est telle que $v(\{i,j\}) > v(\{i\}) + v(\{j\})$ alors il existe une interaction positive entre les objectifs i et j ;
- si la capacité est telle que $v(\{i,j\}) = v(\{i\}) + v(\{j\})$ alors il n'existe aucune interaction entre i et j ;
- si la capacité est telle que $v(\{i,j\}) < v(\{i\}) + v(\{j\})$ alors l'interaction entre i et j est négative.

Dans l'exemple précédent, il existe une interaction positive entre les objectifs 1 et 2 car $v(\{1,2\}) = 0.5 > v(\{1\}) + v(\{2\}) = 0.1 + 0.2$ et il n'existe pas d'interaction entre les objectifs 2 et 3 car $v(\{2,3\}) = 0.5 = v(\{2\}) + v(\{3\}) = 0.2 + 0.3$. Aucune interaction n'est négative dans cet exemple.

Il existe une autre formulation de cet opérateur fondée sur l'utilisation de l'inverse de Möbius $m_v : 2^N \to \mathbb{R}$ associée à la capacité v. Elle est définie par

$$m_v(A) = \sum_{B \subseteq A} (-1)^{|A \setminus B|} v(B), \forall A \subseteq N$$

La quantité $m_v(A)$ est appelée la *masse de Möbius* de l'ensemble A, elle permet de réécrire la capacité $v(A)$ pour tout $A \subseteq N$ de la manière suivante :

$$v(A) = \sum_{B \subseteq A} m_v(A)$$

On obtient alors la formulation suivante de l'intégrale de Choquet :

$$C_v(\mathbf{a}) = \sum_{A \subseteq N} m_v(A) \min_{i \in A} u_i(x)$$

Grâce à cette formulation, il est plus simple de définir une intégrale de Choquet permettant de privilégier les solutions aux vecteurs de performances équilibrés. En effet, il est facile de voir que toute capacité dont les masses de Möbius sont positives est nécessairement convexe. Or, il peut être plus simple de construire des masses m_v positives (ou de vérifier la positivité de masses existantes) que de construire une capacité convexe (ou de vérifier la formule 1.11 pour toute paire (A, B) de sous-ensembles possibles). Un autre avantage de ces masses est qu'elles permettent de définir assez simplement une famille de classes de capacités relativement importantes : les capacités *k-additives*.

Une capacité v est dite *k-additive* si :

$$\begin{cases} m_v(A) = 0 & \text{pour tout } A \subseteq V \text{ tel que } |A| > k \\ et & \\ m_v(A) > 0 & \text{pour au moins un ensemble } A \subseteq N \text{ tel que } |A| = k \end{cases}$$

Si $k = 1$, on retrouve la propriété classique d'additivité, i.e., $\forall A, B \subseteq N, v(A \cup B) + v(A \cap B) = v(A) + v(B)$. Plus la valeur de k est grande, plus le pouvoir descriptif de l'intégrale de Choquet est important, mais cette expressivité se fait au prix de la simplicité effective du modèle, le grand nombre de paramètres qu'une telle capacité entraîne causant un coût computationnel important. Or, pour des valeurs $k > 1$ relativement petites, les capacités *k-additives* peuvent offrir une très bonne expressivité des interactions positives et négatives entre objectifs avec un nombre réduit de paramètres.

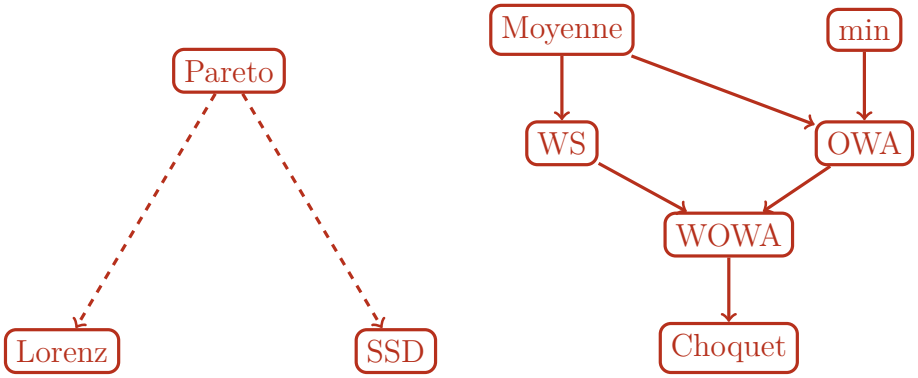
Pour finir, l'intégrale de Choquet est un opérateur *idempotent* et *monotone* par rapport à la règle de dominance FSD [2] [Wakker, 1990; Chateauneuf, 1991] (notons que dans ce cas la relation de préférence induite par C_v peut être représentée par un modèle RDU [Wakker, 1990]). Elle représente un modèle de décision possédant une capacité descriptive très intéressante et bien supérieure à d'autres modèles. Néanmoins, cela vient au prix d'un nombre de paramètres supérieur aux modèles que nous avons vus précédemment, ce qui cause (de manière évidente) une difficulté supplémentaire lors de la résolution du problème de décision considéré. Il reste possible de modéliser un très large éventail de comportements décisionnels avec un nombre réduit de paramètres en ayant recours à des capacités k-additives.

b. Récapitulatif des liens entre les opérateurs d'agrégation

Le schéma ci-dessous résume les différents liens entre les relations de dominance de Pareto, Lorenz et SSD (à gauche), ainsi que les liens entre les opérateurs d'agrégation cités plus haut (à droite). Une flèche (en pointillé) entre deux règles de dominance signifie que l'existence d'une relation de dominance entre deux solutions au sens de la règle se trouvant à l'extrémité initiale implique l'existence d'une relation de dominance entre ces deux mêmes solutions au sens de la règle se trouvant à l'extrémité finale. Une flèche entre

2. La dominance FSD (pour First Stochastic Dominance) est la relation définie par $x \succsim_{\text{FSD}} y \Leftrightarrow \forall z \in \mathbb{R}, F_x(z) \leq F_y(z)$, avec $F_x(z) = v(\{i \in N \mid u_i(x) \leq z\})$.

deux opérateurs signifie que l'opérateur situé à l'extrémité initiale est un cas particulier de l'opérateur situé à l'extrémité finale. Afin de simplifier le schéma, certaines relations ne sont pas représentées mais peuvent être déduites par transitivité.



Plus l'opérateur est général, plus il permet de modéliser des comportements complexes. Néanmoins, du fait de leur non-linéarité et de leur complexité, les modèles plus expressifs sont souvent les plus difficiles à optimiser, particulièrement sur domaine combinatoire. Un compromis peut être à faire entre expressivité et efficacité computationnelle lors du choix de l'opérateur à utiliser pour modéliser les préférences du décideur. On utilise généralement le modèle le plus simple possible (au vu des informations dont on dispose) : par exemple, pour un problème de décision multi-agents, on utilisera un OWA et non une intégrale de Choquet si le décideur déclare rechercher une solution équitable et qu'il considère que les agents ont tous la même importance.

On présente maintenant un bref historique des méthodes de résolutions existantes pour exploiter les modèles cités plus haut dans une logique d'optimisation.

1.2.3 Optimisation d'une fonction d'agrégation

Une fois que l'on a choisi la fonction d'agrégation adaptée à la modélisation des préférences du décideur, déterminer une recommandation satisfaisante pour lui revient à optimiser la valeur de cette fonction d'agrégation. Lorsque l'ensemble des solutions possibles $\mathcal{X}$ est défini de manière explicite, déterminer une solution optimale se fait simplement en comparant l'évaluation de chaque solution de $\mathcal{X}$ selon l'opérateur considéré. À l'inverse, lorsque $\mathcal{X}$ est défini de manière implicite, la détermination d'une solution optimale peut s'avérer être plus complexe à cause du nombre exponentiel de solutions réalisables à comparer. Si l'opérateur utilisé est linéaire, on peut facilement avoir recours à la programmation linéaire, à des algorithmes constructifs ou à la programmation dynamique pour résoudre le problème. L'utilisation de ce type de méthodes est par contre plus difficile dans le cas d'opérateurs d'agrégation comme le min, OWA, WOWA et l'intégrale de Choquet qui ne respectent pas le principe d'optimalité de Bellman. En effet, du fait de l'opération de tri des valeurs de performances lors du calcul de $\text{OWA}_w(x)$, $\text{WOWA}_\varphi(x)$ ou encore $C_v(x), x \in \mathcal{X}$, ces opérateurs sont non-linéaires. Différents travaux ont été proposés dans la littérature dans le but de concevoir des méthodes d'optimisation efficaces pour ces opérateurs. On peut les séparer en deux catégories : les méthodes d'*optimisation* d'un modèle de décision dont les valeurs de paramètres sont précisément *fixées*, et les méthodes

d'*optimisation* et d'*élicitation* d'un modèle de décision dont les valeurs de paramètres ne sont *pas* précisément *fixées*.

a. Optimisation d'une fonction d'agrégation dont le paramètre est fixé

Il existe différents travaux fondés sur une *linéarisation* du modèle d'agrégation. On peut citer la linéarisation d'un OWA à poids décroissants, introduite par Ogryczak et Śliwiński [2003], et qui exploite la formulation d'OWA par les composantes de Lorenz (équation (1.4)). Une alternative à cette formulation est celle introduite par Chassein et Goerigk [2015], ou encore dans un cadre plus général, une linéarisation d'un OWA dans le cas d'un vecteur de pondération quelconque [Boland *et al.*, 2006]. Il existe également des linéarisations de WOWA [Ogryczak et Śliwiński, 2007], ainsi que de l'intégrale de Choquet [Lesca et Perny, 2010; Martin et Perny, 2020].

Ces formulations sont souvent efficaces en pratique, mais perdent en efficacité lorsque le nombre d'objectifs augmente. En effet, ces linéarisations nécessitent l'introduction de nouvelles variables (et de contraintes associées) au programme linéaire modélisant le problème initial, et le nombre de variables et de contraintes additionnelles dépend du nombre d'objectifs à optimiser. De plus, l'efficacité computationnelle varie également (de manière évidente) avec la nature du problème traité. Lorsque ce problème est tel que ces formulations sont inefficaces (ou peu efficaces), d'autres méthodes de résolution peuvent être utilisées : des procédures d'énumérations ordonnées [Galand et Perny, 2006, 2007] ou encore de séparation et évaluation pour la détermination de solutions OWA optimales et Choquet optimales [Galand *et al.*, 2010]. Dans les deux cas précédents, il n'est pas question de linéarisation mais la résolution s'appuie sur une borne sur la valeur optimale, borne calculée en se ramenant à un opérateur linéaire. Cette borne est utilisée, d'une part, afin de guider l'exploration de l'espace des solutions, et d'autre part, afin de garantir que la solution retournée est bien OWA optimale ou Choquet optimale.

b. Difficulté de l'extension au cas où le paramètre n'est pas fixé

Les méthodes citées plus haut ne s'appliquent que lorsque le paramètre de la fonction d'agrégation est précisément connu. Pourtant, il arrive très souvent dans des situations réelles que l'on ne connaisse pas de manière précise les préférences du décideur. Dans ce cas, le paramètre du modèle peut prendre un grand nombre de valeurs, voire une infinité de valeurs, et on ne connaît pas celle qui correspond le mieux au comportement décisionnel du décideur. Or, il est (très) souvent très difficile (voire impossible) pour un décideur d'expliciter son système de valeurs, il ne sait même pas toujours précisément le type de solutions qu'il recherche. Par exemple, il est assez simple d'exprimer le fait de vouloir une solution équitable, mais il existe une infinité de compromis entre utilitarisme et égalitarisme pouvant définir cette équité (comme nous avons pu le voir précédemment), et le décideur est rarement capable de décrire précisément quelle est *sa* définition de cette notion. Cette difficulté ne concerne pas seulement l'*expression* d'une notion de préférence mais également *sa connaissance*. En effet, s'il peut être très simple de savoir que l'on préfère une solution à une autre en comparant leur structure ou l'équilibre de leur vecteur de performances, il est par contre beaucoup plus difficile de savoir, a priori, quelle est la forme précise du vecteur de performances que l'on souhaite obtenir, d'autant plus que cette forme, si elle est connue, n'est peut-être même pas réalisable.

Dans une telle situation, le paramètre de l'opérateur d'agrégation peut également être vu comme une variable du problème. Ainsi, les formulations linéaires citées plus haut ne conviennent pas car on se retrouve à optimiser, dans le meilleur des cas, un objectif quadratique. Par exemple, l'objectif du programme défini par $\max_{x\in\mathcal{X},w\in W} \mathrm{OWA}_w(x) = \sum_{i=1}^{n} w_i u_{(i)}(x)$ devient quadratique puisque w et x sont toutes deux des variables. Cet aspect quadratique entraîne une difficulté relativement importante lors de la résolution du problème. D'autre part, une telle optimisation n'a souvent pas de sens lors de la conception d'un système de recommandation. En effet, comme nous avons pu le voir précédemment, les modèles de décision que l'on considère sont entièrement caractérisés par leurs paramètres qui modélisent les préférences du décideur et qui, en variant, peuvent permettre d'aboutir à des solutions de Pareto très différentes. Ainsi, proposer une recommandation pertinente au décideur ne revient pas à déterminer une solution ayant une valeur agrégée maximum pour tout paramètre possible (par exemple, $\max_{x\in\mathcal{X},w\in W} \mathrm{OWA}_w(x)$), mais de trouver une solution ayant une valeur agrégée maximum pour *le* paramètre ou *le sous-ensemble* de paramètres qui convient pour modéliser les *préférences particulières du décideur* ($\max_{x\in\mathcal{X}} \mathrm{OWA}_{\hat{w}}(x)$ si $\hat{w}$ modélise les préférences particulières du décideur). C'est pourquoi il est important de bien maîtriser le choix des paramètres afin de garantir que la recommandation proposée au décideur corresponde à ses préférences et à ce qu'il recherche. Il faudra alors avoir recours à une méthode d'*élicitation des préférences* afin de récolter des informations préférentielles auprès du décideur et de modéliser ses préférences le plus efficacement possible.

Notons que les linéarisations citées ne sont pas les seules méthodes à ne pas convenir à une connaissance partielle des préférences du décideur. Les méthodes d'énumération citées [Galand et Perny, 2006, 2007; Galand *et al.*, 2010] ne peuvent pas non plus être appliquées telles quelles lorsque la valeur des paramètres n'est pas précisément fixée. Néanmoins, nous verrons qu'elles peuvent être adaptées à ce contexte et combinées à l'élicitation des préférences afin de déterminer une solution optimale pour le décideur (cf. chapitre 2).

1.3 Élicitation des préférences

L'élicitation des préférences est une procédure d'interaction entre le système d'aide à la décision et le décideur. Elle vise à récolter les informations préférentielles les plus utiles possibles afin de guider la recherche vers une solution pouvant correspondre au mieux aux préférences du décideur. La qualité de la solution recommandée par le système dépend alors fortement de l'efficacité de la procédure d'élicitation. Cette efficacité est souvent mesurée par le nombre de questions qu'elle requiert pour la détermination d'une bonne recommandation, mais également par la pertinence des questions posées (d'un point de vue du caractère informatif), par l'effort cognitif qu'elle impose au décideur, et par le temps et la complexité de traitement des informations acquises. La difficulté de la conception d'une méthode *efficace* réside dans le fait que la simplicité cognitive ne va pas toujours de pair avec l'efficacité informative, notamment lorsque les préférences du décideur sont complexes (exigence d'équilibre dans le vecteur de performances, interactions entre objectifs, ...). Un modèle décisionnel bien choisi peut permettre de traduire des préférences complexes à partir de relations de préférences simples données par le décideur. Le plus souvent, et c'est le cas dans cette thèse, le système détermine (à l'aide d'un critère de choix

à définir) une paire de solutions $(x, y) \in \mathcal{X}^2$ qu'il demande au décideur de comparer. Sa réponse, sous la forme "$x \succsim y$" ou "$y \succsim x$", est utilisée afin de mettre à jour l'information sur les paramètres préférentiels. Il est généralement plus facile de répondre à ces questions que de répondre à d'autres questions du type : "pouvez vous classer ces k solutions ?" (avec $k > 2$), "la valeur que vous associez à telle solution est-elle supérieure à α ?" (pour $\alpha \in \mathbb{R}$), "si la recommandation x ne vous convient pas, indiquez un objectif pouvant être dégradé ainsi qu'une borne sur la valeur de cette dégradation", etc. En plus d'être (souvent) simples, les questions de comparaison par paires peuvent être très informatives si elles sont correctement choisies. Notons que lorsque les préférences sont transitives, alors toute relation de préférence concernant une paire de solutions apporte une information préférentielle non seulement sur la paire, mais également sur d'autres paires par transitivité. Lorsque l'on fait l'hypothèse qu'il convient de modéliser les préférences du décideur par un modèle particulier, un très grand nombre d'informations préférentielles supplémentaires peuvent être induites :

Exemple 1.8. *Supposons que l'on dispose de l'ensemble de solutions $\mathcal{X}$ dont l'image dans l'espace des objectifs est $\mathcal{Y} = \{(2, 10), (9, 2), (6, 9), (7, 8), (8, 6)\}$, et que l'on obtienne l'information $(6, 9) \succsim (7, 8)$. A ce stade, la seule conclusion possible est que la solution associée au vecteur de performances $(7, 8)$ n'est pas optimale. Supposons maintenant que les préférences du décideur sont modélisées par une somme pondérée WS_w de paramètre $w \in \{w \in \mathbb{R}^2_+ | w_1 + w_2 = 1\}$. La relation de préférences obtenue se traduit alors par $6w_1 + 9w_2 \geq 7w_1 + 8w_2$, ou encore $w_2 \geq w_1$. Il est facile de vérifier que cette information nous permet de déduire que la solution associée à $(6, 9)$ est préférée aux solutions associées à $(8, 6)$ et $(9, 2)$ qui ne sont donc pas optimales. Ainsi, après avoir obtenu une information sur* une *paire de solutions, seules deux solutions sont encore à comparer pour identifier une solution préférée.*

On se place dans un cadre où les préférences du décideur sont modélisées par une fonction d'agrégation f_ω de paramètre ω. Cette fonction associe à toute solution $x \in \mathcal{X}$ une valeur scalaire $f_\omega(x)$, où la définition précise de f (WS, OWA, WOWA ou intégrale de Choquet) dépend des préférences du décideur et s'écrit en fonction de ω et de $u(x)$. Nous distinguons ici deux types de problématiques :

1. Élicitation des valeurs de performances. On suppose ici que les valeurs de performances $u_i(x), i \in N$, qu'associe le décideur à chaque solution $x \in \mathcal{X}$, ne sont pas connues de manière précise. Il faut alors les éliciter afin de pouvoir déterminer une solution qui puisse convenir au décideur. Il existe différentes méthodes permettant d'effectuer cette élicitation. Ces méthodes font généralement l'hypothèse que la fonction f_ω et son paramètre ω sont connus de manière précise. Sous cette hypothèse, la méthode UTA (UTilité Additive) [Jacquet-Lagreze et Siskos, 1982] utilise les informations préférentielles disponibles (sous forme d'ensemble de solutions triées par ordre de préférence) pour déterminer un ensemble de valeurs de performances compatibles avec ces informations. Cet ensemble peut alors être exploité de différentes manières : on peut choisir, à l'aide d'une heuristique, une instanciation possible de valeurs de performances afin de déterminer une solution f_ω-optimale ou une solution approchée avec une garantie de performance ; on peut également utiliser la totalité des instanciations possibles pour déterminer des solutions *potentiellement* et *nécessairement* optimales [Slowinski *et al.*, 2005; Greco *et al.*, 2009] (voir le

chapitre 2 pour les notions d'optimalité potentielle et nécessaire), ou pour déterminer une recommandation pertinente à l'aide de critères tels que le minimax regret [Savage, 1954] (cf. sous-section 1.3.2). On peut également citer la méthode MACBETH [E Costa et Vansnick, 1995], qui élicite les préférences du décideur en posant des questions de comparaisons par paires de solutions (du type "$x \succsim y$?"), ainsi que des questions de différences d'intensités de préférences entre paires de solutions (du type "$f_\omega(x) - f_\omega(y) \geq f_\omega(z) - f_\omega(t)$?"). La méthode détermine ensuite par programmation linéaire des valeurs de performances compatibles avec les préférences du décideur. Un processus interactif est également proposé dans le cas où les informations fournies par le décideur ne sont pas cohérentes. Notons qu'il existe un outil graphique d'aide à la décision implémentant la méthode MACBETH (`http://m-macbeth.com/`).

2. Élicitation des valeurs de paramètres. On suppose ici que les arbitrages que fait le décideur entre les différents objectifs ne sont pas connus. L'élicitation des préférences porte donc sur la valeur des paramètres ω qui modélisent les préférences du décideur face à une situation de décision multi-objectifs. Dans ce type de méthodes, on suppose que la nature des préférences du décideur (exigence d'équité ou de robustesse par exemple) est connue et permet de déterminer le type d'opérateur défini par f. On suppose également que les valeurs de performances $u_i(x)$ pour tout $i \in N$, sont connues de manière précise (lorsqu'il s'agit d'évaluations objectives comme des prix ou des distances par exemple), ou qu'elles ont été élicitées en amont (lorsqu'il s'agit d'évaluations subjectives comme des valeurs de satisfaction par exemple). Cette élicitation préalable peut se faire en appliquant des méthodes telles que celles présentées dans le paragraphe précédent à chaque $u_i, i \in N$.

Dans cette thèse, nous nous focalisons sur le second type de problématiques car on cherche à éliciter le comportement décisionnel du décideur et non les valeurs de performances qu'il associe à chaque solution ou sous-solution. Il est important de noter que, comme nous le verrons dans la suite, l'objectif de cette élicitation n'est pas de déterminer *le* jeu de paramètres modélisant les préférences du décideur de manière précise, mais de réduire suffisamment l'incertitude concernant la valeur de ω pour pouvoir déterminer une recommandation de bonne qualité. En effet, une élicitation complète de la valeur des paramètres peut être fastidieuse lors de la résolution de problèmes définis sur domaines explicites à cause du grand nombre de questions à poser : $n(n-1)$ questions pour n solutions. Elle devient même inenvisageable dans le cas de la résolution de problèmes définis sur domaine combinatoire car elle peut nécessiter un nombre de questions excessivement grand entraînant des interactions interminables avec le décideur. En outre, une élicitation complète et précise n'est pas nécessaire dans le cas de la problématique de choix. Nous verrons dans la suite du document qu'il existe un vaste ensemble de méthodes permettant de déterminer une solution optimale malgré une incertitude encore relativement grande sur la valeur exacte du paramètre. L'exemple suivant illustrant ce propos :

Exemple 1.9. *Supposons que l'on dispose d'un ensemble de solutions* $\{x^0, \ldots, x^{10}\}$ *dont les vecteurs de performances sont donnés par* $u(x^i) = (i, 10-i), i \in \{0, \ldots, 10\}$. *Ces vecteurs sont représentés sur la figure 1.6.*

Supposons que les préférences du décideur peuvent être modélisées par une somme pondérée de paramètre $w \in \{w \in \mathbb{R}^2 | w_1 + w_2 = 1\}$*, et que l'on dispose de l'information préférentielle suivante :* "le décideur préfère strictement la solution x^7 à la solution x^4"*, que*

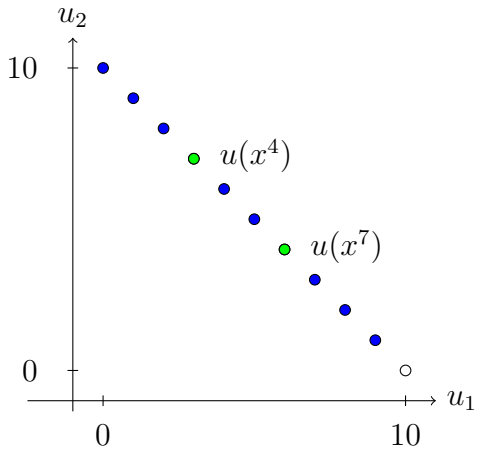


FIGURE 1.6 – Exemple d'instance dans l'espace bicritère.

l'on traduit par la contrainte $\mathrm{WS}_w(x^7) > \mathrm{WS}_w(x^4)$*, ou encore* $w_1 > w_2$ *(soit* $w_1 \in (1/2, 1]$ *avec* $w_1 + w_2 = 1$*). Dans ce cas, il n'est pas nécessaire de connaître la valeur précise de* w *pour savoir que la solution* x^{10} *(représentée par un rond blanc sur la figure) est optimale. En effet, on a* $\mathrm{WS}_w(x^i) = iw_1 + (10 - i)w_2 = iw_1 + (10 - i)(1 - w_1)$*. Il est facile de vérifier que* $\mathrm{WS}_w(x^i) < 10w_1 = \mathrm{WS}_w(x^{10})$ *pour tout* $w_1 \in (1/2, 1]$*. On peut donc détecter l'optimalité de la solution* x^{10} *avec une connaissance très imprécise de la valeur du paramètre.*

Nous allons maintenant présenter un état de l'art (non-exhaustif) des méthodes d'élicitation des préférences. Même si nous nous intéressons ici à l'élicitation des *paramètres* d'un opérateur d'agrégation, nous parlerons également de la littérature concernant l'élicitation des valeurs de performances car les deux littératures se complètent. Nous séparons ici les méthodes d'élicitation en deux catégories : les méthodes d'élicitation utilisant des *exemples* et les méthodes d'élicitation *incrémentale*. Il existe deux différences principales entre ces deux catégories : 1) l'approche utilisant des *exemples* effectue une élicitation qui vise à déterminer une instanciation de paramètres compatible avec les informations préférentielles disponibles, le modèle obtenu peut ensuite être utilisé afin de déterminer une solution préférée. Contrairement à cette approche, l'approche *incrémentale* a pour objectif de déterminer une recommandation en posant le moins de questions possible ; 2) les méthodes fondées sur l'utilisation d'*exemples* utilisent des informations préférentielles données en entrée (informations imposées), alors que les méthodes *incrémentales* sont fondées sur la conception de questionnaires personnalisés et visent à déterminer une recommandation pertinente en posant le moins de questions possible.

1.3.1 Approche d'élicitation utilisant des exemples

En apprentissage, un *exemple* est une donnée contenant une information sur les préférences du décideur (ou d'un ensemble de décideurs). Les exemples sont, le plus souvent, des données *imposées* (elle ne correspondent pas à des informations choisies et demandées par l'algorithme). Ces méthodes se focalisent généralement sur l'analyse des informations préférentielles *disponibles* afin de déterminer des valeurs de paramètres *précises* et *représentatives* des préférences. La spécificité de chaque méthode relevant de cette approche

réside dans le choix de la valeur des paramètres car il arrive souvent que plusieurs valeurs soient compatibles avec les informations préférentielles dont on dispose. Une fois les valeurs de paramètres choisies, une solution optimale peut être déterminée.

Notons que la terminologie d'*exemple* est souvent utilisée dans un cadre de classification binaire, où chaque exemple est constitué d'une entrée, généralement sous la forme d'un vecteur, et d'une sortie sous la forme d'une classe ou d'une étiquette (0 ou 1 dans le cas binaire). Il s'agit alors de déterminer une fonction dite *de classification* qui explique au mieux le lien entre les données et leur étiquette. Elle est ensuite utilisée pour estimer la classe de tout nouvel exemple non étiqueté. Les problèmes que nous traitons dans cette thèse peuvent facilement être ramenés à des problèmes de classification binaire : lorsque les questions préférentielles que l'on pose au décideur sont de la forme " $x \succsim y$?", deux réponses sont possibles (*oui* et *non*) ; on peut alors considérer que la paire (x, y) est une entrée dont la classe est 1 si la réponse est oui, et 0 si la réponse est non. Estimer la réponse à une question revient à estimer la classe de la paire de solutions à comparer. Une fonction permettant d'étiqueter correctement les exemples en entrée est donc une fonction modélisant des préférences compatibles avec les informations préférentielles véhiculées par ces exemples.

Les exemples peuvent être obtenus de manière *statique* au début de l'algorithme, ou *dynamique* au cours du temps. Dans le second cas, le modèle estimé est révisé à chaque fois qu'il n'est pas compatible avec un nouvel exemple (ou groupe d'exemples) étiqueté(s).

a. Élicitation utilisant des exemples statiques

Ces méthodes consistent à récolter un maximum d'informations préférentielles en amont de la méthode de résolution, puis à estimer le mieux possible le modèle représentant les préférences du décideur afin de déterminer une bonne recommandation à lui présenter. Une approche d'élicitation utilisant des exemples statiques peut être qualifiée d'approche en deux phases : la première phase est celle d'*élicitation*, et se focalise sur l'analyse des informations préférentielles *disponibles* afin de déterminer des valeurs de paramètres *représentatives* des préférences. La seconde phase est celle de *résolution* ; elle consiste à utiliser le modèle obtenu afin de déterminer la solution optimale parmi n'importe quel ensemble de solutions possibles.

L'une des premières méthodes développées dans ce cadre est l'algorithme du *perceptron*, proposé par Rosenblatt [1958], qui est un algorithme de classification binaire. Il consiste à classifier des données à l'aide d'un séparateur [3] *linéaire* (ou fonction de classification) défini par une somme pondérée. Le jeu de poids de cette somme pondérée est obtenu, en utilisant les informations préférentielles disponibles, par une méthode similaire à l'algorithme du gradient. L'algorithme du perceptron a été largement étudié, et de nombreuses variantes ont été proposées dans la littérature. Par exemple, Amit *et al.* [1989] développent une variante du perceptron permettant de traiter le cas où l'on contraint à la positivité la somme des poids utilisés dans la somme pondérée ; Stephen [1990] propose une variante où la contrainte de séparabilité linéaire des données est relaxée et où il est possible de gérer la présence d'incohérences dans les exemples. Un autre type de méthodes de classification utilisant des exemples statiques est constitué par les algorithmes de type

3. Le terme *séparateur* provient du fait que cette fonction *sépare* les données du problème en deux classes ou plus (0 et 1 par exemple).

SVM (pour *Support Machine Vector*) [Cortes et Vapnik, 1995], qui consistent à séparer linéairement des données en utilisant un critère de marge maximum. La marge étant la distance entre la droite séparatrice et les vecteurs (entrées des exemples) les plus proches que l'on appelle des *vecteurs supports*.

Dans le cadre de la résolution de problèmes d'optimisation, l'une des premières contributions a été faite par Charnetski et Soland [1978] pour la résolution d'un problème multi-attributs [4] où le modèle considéré est la somme pondérée des utilités des valeurs des attributs. Cette contribution est importante pour nos travaux puisqu'elle a introduit la notion de *polyèdres d'optimalité* (sur laquelle nous reviendrons dans la suite) : le polyèdre d'optimalité associé à une solution x est le sous-ensemble de paramètres possibles pour lequel la solution x est WS-optimale. Les auteurs construisent, dans un premier temps, l'ensemble d'incertitude comme l'ensemble des paramètres compatibles avec les données en entrée. Dans un second temps, les polyèdres d'optimalité de toutes les solutions sont calculés : s'il n'existe qu'un seul polyèdre d'optimalité non-vide, alors la solution associée est optimale. Sinon ils associent, à chaque solution, une probabilité d'optimalité proportionnelle à l'hypervolume de son polyèdre d'optimalité. Finalement, la solution recommandée est la solution ayant la plus grande probabilité. La figure 1.7 donne un exemple simple d'un tel découpage : les zones hachurées correspondent à des zones exclues de l'espace des paramètres à l'aide d'informations préférentielles données par le décideur ; le polyèdre restant est ensuite divisé en trois polyèdres d'optimalité. La probabilité d'optimalité des solutions associées aux polyèdres sont données sur la figure.

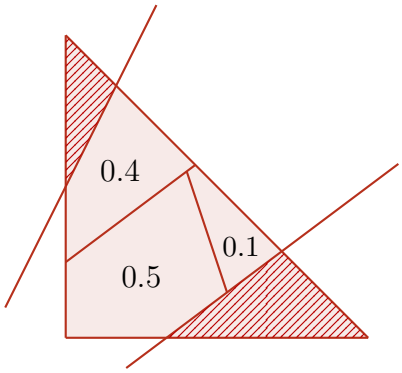


FIGURE 1.7 – Exemple de partitionnement en polyèdres d'optimalité.

Lahdelma *et al.* [1998] se sont fondés sur cette notion de *polyèdres d'optimalité* pour définir la méthodologie SMAA (*Stochastic Multicriteria Acceptability Analysis*) pour la résolution de problèmes multicritères où les préférences du décideur sont également modélisées par une somme pondérée. Cette méthodologie est destinée au traitement de l'imprécision ou de l'incertitude dans les données du problème (les valeurs de performances $u_i, i \in N$, des solutions de $\mathcal{X}$). Pour cela, ils considèrent les valeurs de performances comme des variables aléatoires leur permettant de définir des *indices d'acceptabilité* et un *facteur de confiance* donnant, respectivement, la probabilité que chaque solution soit optimale suivant le principe de Charnetski et Soland, et le degré de précision des données afin de déterminer la pertinence d'une recommandation. La méthodologie s'adapte

4. Un problème multi-attributs est un problème où le décideur évalue une solution sur un ensemble d'attributs, l'évaluation de la solution dépendant de son jugement sur chaque attribut.

à tout modèle linéaire en son paramètre ; elle a notamment été adaptée aux intégrales de Choquet 2-additives par Angilella *et al.* [2015].

Toujours dans un cadre de décision multicritère, plusieurs méthodes d'élicitation ont été développées pour l'élicitation de la capacité d'une intégrale de Choquet. On peut citer par exemple [Marichal et Roubens, 2000], où une capacité compatible avec les informations préférentielles dont on dispose est déterminée par programmation linéaire. En classification binaire multi-attribut, Tehrani *et al.* [2012] introduisent une méthode de régression "Choquistique" consistant à remplacer le terme linéaire de la régression logistique classique par une intégrale de Choquet. Les auteurs déterminent ensuite une masse de Möbius compatible avec les exemples étiquetés donnés en entrée en résolvant un programme d'optimisation quadratique.

Pour finir, dans le cadre de l'élicitation des valeurs de performances u_i des solutions de $\mathcal{X}$, on retrouve également les méthodes UTA [Jacquet-Lagreze et Siskos, 1982] et MACBETH [E Costa et Vansnick, 1995] citées plus haut.

Le principal avantage des méthodes utilisant des exemples statiques est qu'elles sont indépendantes de l'instance traitée, ce qui simplifie leur (ré)utilisation. En effet, une fois le modèle estimé, il peut être utilisé pour résoudre n'importe qu'elle instance dès lors que les objectifs ainsi que les préférences sont les mêmes. Ces méthodes peuvent être relativement efficaces en pratique mais sont très sensibles à la quantité et à la qualité des exemples disponibles. En effet, la recommandation faite par l'algorithme peut être très loin des aspirations du décideur si les exemples disponibles sont trop peu nombreux ou qu'ils ne sont pas suffisamment représentatifs des préférences du décideur. Néanmoins, un nombre d'exemples trop grand, même s'il peut être plus représentatif, implique une plus grande probabilité de voir apparaître des incohérences. Dans ce cas, selon la méthode utilisée pour estimer les valeurs de paramètres, on peut se retrouver dans deux situations différentes : l'algorithme se trouve dans l'incapacité de déterminer des valeurs de paramètres, et donc dans l'incapacité de déterminer une recommandation, ou bien il peut recommander une solution mais qui peut être de mauvaise qualité. Or, ces méthodes sont conçues de manière à s'interrompre totalement lorsque la phase de résolution est terminée ; il n'est donc pas possible de donner au décideur l'opportunité d'exprimer une insatisfaction ou de corriger une incohérence. Il n'est de toute façon pas possible de déterminer une autre recommandation. Ainsi, aucune garantie ne peut être faite, a priori, sur la qualité de la recommandation.

b. Élicitation en ligne à l'aide d'exemples

Ces méthodes suivent le même principe que les méthodes précédentes à la différence que les exemples ne sont pas tous disponibles en un bloc au début de l'algorithme, ils arrivent dans un certain ordre et au fur et à mesure de l'exécution de l'algorithme. On peut dire de ces exemples qu'ils arrivent de manière dynamique. A chaque nouvel exemple étiqueté, l'estimation courante du modèle décisionnel est mise à jour si elle n'est pas compatible avec l'information préférentielle véhiculée par cet exemple. Lorsqu'un nouvel exemple non étiqueté arrive, l'estimation permet de prédire le comportement du décideur face à ce nouvel exemple. Ainsi, une fois que tous les exemples étiquetés ont été obtenus et que la phase d'estimation est terminée, le modèle obtenu peut être réutilisé pour la résolution de n'importe quelle instance.

Beaucoup d'algorithmes utilisant des exemples statiques peuvent être adaptés très facilement à l'arrivée dynamique des exemples. Par exemple, des variantes de l'algorithme du perceptron utilisant des exemples dynamiques ont été proposées par Frean [1992]; Heskes [2000]; Shivaswamy et Joachims [2015]. Ces variantes ont la particularité d'être "à mémoire courte" : lorsque de nouveaux exemples incompatibles avec l'estimation courante du modèle arrivent, une nouvelle estimation est faite sans prendre en compte les exemples précédents. Ainsi, une mise à jour faite à une étape i peut mener à un modèle incompatible avec des exemples arrivés à une étape $j < i$. Cette particularité peut donc mener à la détermination d'un modèle qui n'est pas compatible avec toutes les informations récoltées. Néanmoins, cela présente l'avantage de rendre ces méthodes tolérantes à la présence d'incohérences dans les exemples. Les algorithmes de type SVM s'adaptent également à ce cadre, e.g., [Zheng *et al.*, 2013]. Notons que cette catégorie d'algorithmes ne se limite pas à l'adaptation d'algorithmes initialement conçus pour l'utilisation d'exemples statiques, e.g., [Duchi *et al.*, 2011].

Tout comme les méthodes *hors ligne*, ces méthodes en ligne peuvent être très sensibles à la qualité des informations préférentielles disponibles. Même si elles permettent de gérer des exemples arrivant de manière dynamique, elle ne permettent pas de choisir les requêtes préférentielles, ou de demander au décideur d'apporter de nouvelles informations afin d'ajuster le modèle lorsque la recommandation ne lui convient pas. Ce problème vient du fait que l'élicitation et la résolution se font de manière séquentielle, et que l'algorithme ne contrôle ni le nombre d'exemples ni la nature de ces exemples. La sous-section suivante présente des méthodes palliant cet inconvénient.

1.3.2 Approche d'élicitation incrémentale des préférences

L'approche d'élicitation *incrémentale* ou *active*[5] est celle qui nous intéresse dans cette thèse. Elle consiste à alterner questions préférentielles et exploration de l'espace des solutions jusqu'à pouvoir déterminer une solution à recommander. L'idée est de pouvoir adapter le questionnaire à l'instance traitée et aux réponses du décideur afin de concentrer l'effort d'élicitation sur les informations préférentielles *utiles* et *nécessaires* à la détermination d'un choix optimal. Ainsi, contrairement à l'approche précédente, les informations préférentielles ne sont pas imposées mais correspondent aux réponses du décideur à des questions choisies par l'algorithme. Lors de l'élaboration d'une telle méthode, plusieurs problématiques se posent : quelle est la nature des questions posées ? comment sont-elles choisies ? à quel moment les poser ? comment modéliser les réponses du décideur ? comment explorer l'espace des solutions ? Les travaux de la littérature abordent ces problématiques de différentes manières. L'approche incrémentale est généralement guidée par l'utilisation d'un modèle de décision pour représenter les préférences du décideur. Néanmoins, les premières méthodes que nous présentons ici utilisent des modèles pour guider l'exploration de l'espace des solutions sans pour autant supposer qu'ils modélisent les préférences du décideur.

5. Le terme d'élicitation *active* ou d'apprentissage *actif* est utilisé en apprentissage automatique (*machine learning*). Notons également qu'en machine learning, le terme d'élicitation *incrémentale* est plutôt utilisé pour désigner l'élicitation *en ligne* à l'aide d'exemples (cf. partie *b.* de la sous-section 1.3.1).

a. Élicitation par exploration interactive du front de Pareto

Ces méthodes, dites *interactives*, sont des méthodes fondées sur une exploration du front de Pareto guidée par des interactions avec le décideur. Le principe général est le suivant : à chaque étape, une nouvelle recommandation est générée parmi les solutions de Pareto, le plus souvent par la minimisation d'une distance de Tchebychev pondérée et augmentée à un point de référence (que l'on rappelle dans le papragraphe suivant), ou encore par l'optimisation d'une fonction d'utilité additive pondérée. Le décideur doit alors dire s'il est satisfait ou non par cette recommandation, et dans le second cas il dit pourquoi il n'est pas satisfait. Sa réponse est utilisée afin de diriger la recherche d'une nouvelle recommandation en modifiant la pondération du modèle utilisé (distance de Tchebychev pondérée et augmentée ou utilité additive pondérée), ou en restreignant l'espace des solutions ou le domaine des critères. Notons que la manière dont le décideur doit exprimer la raison de son insatisfaction, ainsi que la manière dont cette information est traitée sont des éléments propres à chaque méthode, et représentent un point déterminant de l'efficacité de la méthode.

Avant de donner quelques exemples de travaux concernant les méthodes interactives, on rappelle la définition de la distance de Tchebychev pondérée et augmentée qui est largement utilisée dans ce type de méthodes. La distance d d'une solution x, de vecteur de performances $u(x)$, à un point de référence r est donnée par Steuer et Choo [1983] :

$$d(x) = \max_{i \in N} w_i(r_i - u_i(x)) + \varepsilon \sum_{i \in N} u_i(x)$$

où w est un vecteur de pondération permettant à la fois de normaliser les valeurs des critères (lorsqu'elles sont exprimées sur des échelles différentes) et de pondérer les critères selon leur importance relative, et où ε est une quantité positive choisie arbitrairement petite pour que le second terme de la somme définisse un second critère, après celui de la distance de Tchebychev pondérée, afin de départager d'éventuels ex æquo.

Notons que dans cette partie de la section, il est souvent question de décision multicritère car la plupart des méthodes d'élicitation interactive ont été développé en aide à la décision multicritère.

La méthode STEM. L'une des premières méthodes interactives développées est la méthode STEM (pour *STEp Method*) [Benayoun *et al.*, 1971]. Elle consiste à déterminer, à chaque étape, une solution de Pareto à l'aide d'une norme de Tchebychev pondérée augmentée, puis à la présenter au décideur. S'il est satisfait par la recommandation, l'algorithme s'arrête. Sinon, le décideur doit indiquer un critère sur lequel il est prêt à dégrader la performance (par rapport à la recommandation qu'il refuse) ainsi qu'une borne supérieure sur la valeur de cette dégradation. Ces informations mènent à des contraintes strictes ajoutées à l'espace des critères définissant ainsi une zone de recherche de plus en plus restreinte. Une approche similaire à STEM a été proposée par Roy [1976] où plusieurs critères peuvent être dégradés à la fois et où il est demandé au décideur de donner un point de référence correspondant à des niveaux d'aspiration qu'il souhaite atteindre (pour améliorer la recommandation courante). Dans cette approche, les réponses du décideur permettent, en plus de définir une zone de recherche, de définir une direction de recherche en modifiant le poids de la norme de Tchebychev en fonction de ces réponses. Une particularité importante de cette méthode par rapport à la méthode STEM, est que

la zone de recherche définie à l'étape i ne dépend pas de celle de l'étape $i-1$: elle est définie directement à partir de la zone de recherche initiale, ce qui a pour avantage de permettre de gérer la présence d'incohérences dans les réponses du décideur. Une autre méthode, introduite par Vincke [1976], étend la méthode STEM en définissant un panel de questions plus riche. Lorsque le décideur n'est pas satisfait de la recommandation courante, il doit successivement répondre aux questions suivantes : quel critère améliorer ? est-il possible de faire une concession sur un critère ? si oui, lequel ? est-il possible de relâcher une contrainte ? si oui, laquelle ? faut-il être plus restrictif sur une contrainte ? si oui, laquelle ?

Une amélioration de STEM. L'inconvénient principal des méthodes issues de la méthode STEM est que les questions posées peuvent imposer au décideur un effort cognitif trop important. Il n'est en effet pas toujours simple de désigner un critère (ou plusieurs) à dégrader, ni de déterminer une borne numérique *précise* et *raisonnable* sur la valeur de cette dégradation. Dans une méthode développée par Geoffrion *et al.* [1972], ce problème est résolu en posant les questions d'une manière différente : le décideur doit comparer une solution représentée par son vecteur de performances à une autre solution obtenue par le transfert d'une quantité de performance d'un critère à l'autre. Ici, aucune solution n'est écartée de l'ensemble des recommandations possibles ; les réponses aux questions posées sont utilisées afin de déterminer une direction de recherche donnée par une formule de gradient. Une nouvelle recommandation est obtenue par un déplacement dans l'espace des critères dans la direction déterminée, avec une intensité déterminée par une interaction graphique avec le décideur. Notons que les questions posées durant cet algorithme peuvent être très nombreuses, demandant ainsi au décideur de fournir un effort trop important. Une autre méthode, proposée par Vanderpooten et Vincke [1989], oriente la recherche d'une recommandation en exploitant les réponses du décideur à deux types de requêtes : 1) à partir d'une recommandation courante quels sont les critères à améliorer pour obtenir une solution plus satisfaisante, 2) la comparaison d'une recommandation courante et d'une amélioration possible de cette recommandation. Ici également, aucune solution n'est écartée de l'espace de recherche : seule la réponse la plus récente est prise en compte pour orienter la recherche d'une nouvelle recommandation.

Des méthodes interactives récentes. Des méthodes interactives ont été développées jusqu'à très récemment. Par exemple, un algorithme proposé par Lokman *et al.* [2018] utilise des questions de comparaison par paires afin de déterminer des cônes de dominance dans l'espace des critères. Ces paires sont composées d'une recommandation courante et d'un adversaire déterminés par la résolution de programmes linéaires dont la fonction objectif est une somme pondérée. Les cônes de dominance permettent d'exclure des solutions dominées de l'espace des solutions et ainsi de mieux orienter la recherche d'une solution optimale. Dans un contexte d'optimisation robuste en présence de préférences incertaines, Zhou-Kangas et Miettinen [2019] présentent une méthode similaire à STEM dans la recherche des solutions à recommander, mais où les questions posées consistent à demander au décideur quel degré de performance de la solution recommandée il est prêt à sacrifier pour améliorer sa robustesse. Pour cela, ils utilisent une mesure de *prix à payer pour la robustesse* et de *gain de robustesse* afin d'aider le décideur à examiner les différents compromis entre robustesse et qualité de la solution.

Dans un algorithme interactif, il appartient souvent au décideur de déterminer le moment où l'exploration de l'espace des solutions s'arrête. En effet, lorsqu'il est satisfait de la recommandation présentée par l'algorithme, la procédure s'interrompt. Or, le décideur ayant une vision très partielle de l'espace des solutions, il peut arriver qu'il interrompe la procédure prématurément, avant que l'algorithme ait pu explorer une zone de recherche qui pourrait fortement l'intéresser. Malheureusement, les algorithmes interactifs ne permettent pas de détecter ce type de situations, ni de donner une garantie sur la qualité de la solution retournée. Cela est dû au fait que, même si ces méthodes utilisent des modèles pour déterminer des recommandations (distance de Tchebychev pondérée et augmentée par exemple), elles n'en utilisent pas pour modéliser les préférences spécifiques d'un décideur, ce qui rend difficile la mesure d'une distance entre la recommandation et la solution optimale pour le décideur.

b. Élicitation incrémentale des paramètres d'un modèle décisionnel

Ces méthodes sont des méthodes d'élicitation *incrémentale* (ou *active*) qui tirent profit de la modélisation des préférences à l'aide de modèles de décision personnalisés tels que ceux présentés dans la section 1.2. Il s'agit de méthodes où, après avoir choisi le modèle f_ω qui convient à la modélisation des préférences du décideur, on définit un *ensemble d'incertitude* $\mathcal{W}$ contenant toutes les valeurs de paramètres ω pouvant être compatibles avec les informations préférentielles disponibles. L'ensemble d'incertitude est ensuite mis à jour à chaque acquisition d'une nouvelle information préférentielle. La figure 1.8 illustre cette approche. Sur cette figure, le carré rouge représente le poids (caché) modélisant les préférences du décideur.

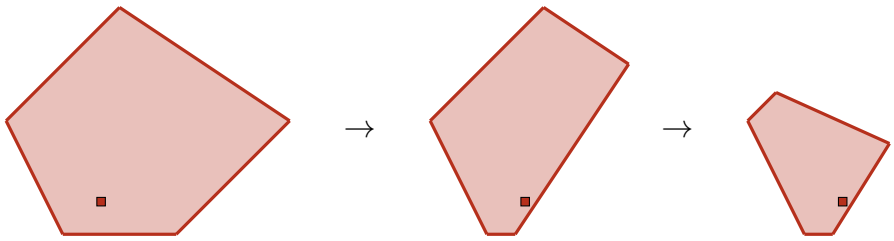


FIGURE 1.8 – Exemple de mise à jour de l'ensemble d'incertitude.

Les informations récoltées sont généralement le résultat de la comparaison de deux solutions x et y par le décideur. Une déclaration du type *"je préfère la solution x à la solution y"* mène à la contrainte $f_\omega(x) \geq f_\omega(y)$ que l'on ajoute à la définition de $\mathcal{W}$. Notons que $\mathcal{W}$ est initialement représenté par un polyèdre convexe car il est généralement initialisé par le simplexe ou par un sous-ensemble convexe du simplexe. De plus, la contrainte $f_\omega(x) \geq f_\omega(y)$ est linéaire dès lors que la fonction d'agrégation f_ω est linéaire en son paramètre ω. Ainsi, l'ajout de toute contrainte induite par une telle information préférentielle conserve la convexité de $\mathcal{W}$, ce qui peut fortement simplifier l'optimisation de fonctions linéaires (en ω) sur cet ensemble. Une procédure d'élicitation incrémentale fondée sur l'utilisation d'un modèle consiste à alterner exploration de l'espace des solutions et questions préférentielles permettant de réduire l'ensemble d'incertitude jusqu'à pouvoir recommander une solution de bonne qualité au décideur. L'objectif de cette élicitation n'est pas de déterminer la valeur exacte du paramètre ω modélisant les préférences du

décideur, mais de réduire suffisamment $\mathcal{W}$ pour pouvoir déterminer une recommandation qui puisse satisfaire le décideur. On peut observer ici que, le paramètre ω n'étant jamais précisément instancié, il est également nécessaire de développer des méthodes spécifiques pour l'exploration de l'espace des solutions.

La méthode ISMAUT. L'une des premières méthodes d'élicitation incrémentales développées est la méthode ISMAUT (pour *Imprecisely Specified Multiattribute Utility Theory*), introduite dans White III *et al.* [1984] dans le cadre de l'élicitation des paramètres d'une utilité multi-attributs additive pondérée. La méthode définit l'ensemble d'incertitude initial $\mathcal{W}$, ainsi que l'ensemble des solutions non-dominées $\mathrm{ND}_{\mathcal{W}}$ au sens de l'utilité additive pondérée, par :

$$\mathrm{ND}_{\mathcal{W}}(\mathcal{X}) = \{x \in \mathcal{X} | \forall y \in \mathcal{X}, y \succsim_{\mathcal{W}} x \Rightarrow x \succsim_{\mathcal{W}} y\}$$

où $\succsim_{\mathcal{W}}$ est la relation binaire définie par :

$$x \succsim_{\mathcal{W}} y \Leftrightarrow \forall \omega \in \mathcal{W}, f_\omega(x) \geq f_\omega(y)$$

L'ensemble ND est alors présenté au décideur qui doit, s'il le peut, choisir la solution qu'il préfère. S'il ne peut pas se décider pour une solution de cet ensemble, de nouvelles informations sont récoltées afin de réduire l'ensemble d'incertitude et de réduire, par conséquent, l'ensemble ND. Les informations récoltées se présentent sous différentes formes : par exemple, par des comparaisons par paires, des informations (potentiellement imprécises) sur le jeu de poids modélisant ses préférences, ou encore des informations (potentiellement imprécises) sur les niveaux d'utilités minimum concernant des alternatives ou leurs conséquences. Ces informations sont récoltées jusqu'à ce que l'ensemble ND soit suffisamment restreint pour que le décideur puisse déterminer facilement une solution préférée. La méthode ISMAUT est focalisée sur la représentation de l'incertitude concernant les valeurs de paramètres ω modélisant les préférences du décideur, la problématique de l'exploration de l'espace des solutions et celle du choix de la question à poser ne sont pas abordées. C'est au décideur de choisir une solution préférée dans ND ou de déclarer qu'il est dans l'incapacité de le faire, et de déterminer une information préférentielle à donner au système de recommandation. Or, il peut être très difficile pour lui de déterminer un choix parmi un ensemble ND de solutions très proches ou très différentes et/ou très nombreuses. Il en est de même pour la détermination d'une information préférentielle utile. Les approches présentées dans la suite abordent ces problématiques afin de pouvoir réduire l'effort cognitif demandé au décideur.

Élicitation incrémentale fondée sur la stratégie CSS. Afin de poser les questions les plus utiles possibles et de réduire le nombre de questions posées au décideur, un intérêt particulier a été porté à la conception de questionnaires simples et personnalisés. Il s'agit d'élaborer une stratégie du choix de la question à poser, c'est-à-dire d'analyser l'ensemble des questions possibles et de déterminer une question simple, mais qui soit la plus informative possible. Une stratégie très répandue en théorie de la décision est la stratégie *CSS* (pour *Current Solution Strategy*) introduite par Wang et Boutilier [2003] pour l'élicitation des valeurs de performances u associées aux solutions de $\mathcal{X}$. Cette stratégie utilise le critère de choix du minimax regret, introduit par Savage [1954] dans le cadre

de la décision dans l'incertain lorsque les valeurs des performances u sont incertaines. Ce critère de choix ainsi que la stratégie CSS ont ensuite été adaptés au cas où l'incertitude porte sur les valeurs de paramètres ω d'une fonction d'agrégation f_ω et non sur les valeurs de performances $u_i, i \in N$, [Salo et Hamalainen, 2001; Boutilier *et al.*, 2009]. C'est naturellement sur ce dernier cadre que nous nous focalisons.

Le critère minimax consiste à évaluer chaque solution du problème par la perte de performance maximum engendrée par le choix de cette solution relativement aux autres solutions du problème et relativement au choix du paramètre ω dans $\mathcal{W}$. Plus formellement, la perte maximum engendrée par le choix de la solution x au lieu de la solution y est définie par le Pairwise Max Regret (PMR) :

$$\mathrm{PMR}(x, y, \mathcal{W}) = \max_{\omega \in \mathcal{W}} \{f_\omega(y) - f_\omega(x)\} \tag{1.12}$$

On peut ensuite définir la pire perte de performance engendrée par le choix x au lieu de toute autre solution de $\mathcal{X}$ par le Max Regret (MR) :

$$\mathrm{MR}(x, \mathcal{X}, \mathcal{W}) = \max_{y \in \mathcal{X}} \mathrm{PMR}(x, y, \mathcal{W}) \tag{1.13}$$

Le critère de choix minimax regret consiste alors à recommander au décideur la solution ayant la meilleure valeur de regret pire cas, c'est à dire le plus petit MR, d'où le nom de MiniMax Regret (MMR) :

$$\mathrm{MMR}(\mathcal{X}, \mathcal{W}) = \min_{x \in \mathcal{X}} \mathrm{MR}(x, \mathcal{X}, \mathcal{W})$$

La stratégie CSS consiste alors à comparer deux *"bonnes solutions"* au sens du critère minimax : on demande au décideur de comparer la meilleure recommandation courante choisie par le critère du minimax regret $x^* \in \arg\min_{x \in \mathcal{X}} \mathrm{MR}(x, \mathcal{X}, \mathcal{W})$, à la solution $y^* \in \arg\max_{y \in \mathcal{X}} \mathrm{PMR}(x^*, y, \mathcal{W})$ réalisant le pire regret possible pour x^*. Même s'il est difficile de démontrer théoriquement le pouvoir informatif des questions posées par cette stratégie, les auteurs ont démontré dans de nombreux travaux qu'en pratique, cette stratégie permet de réduire relativement vite l'incertitude concernant les préférences du décideur (voir par exemple [Wang et Boutilier, 2003; Boutilier *et al.*, 2006a]). La stratégie CSS présente un certain nombre d'avantages (discutés plus en détail dans le chapitre suivant) : le minimax regret donne à la fois un critère de choix, une question à poser ainsi qu'une évaluation de la distance entre la recommandation et la (vraie) solution optimale. Notons que cette distance tend vers 0 au fur et à mesure que des questions sont posées au décideur ; elle peut donc définir un critère d'arrêt pour la recherche d'une solution optimale. Même s'il existe d'autres stratégies du choix de la question, par exemple [French, 1986; Chajewska *et al.*, 2000; Boutilier, 2002], la stratégie CSS reste l'une des stratégies les plus utilisées dans la littérature. Elle est, par exemple, utilisée dans [Braziunas et Boutilier, 2007; Benabbou *et al.*, 2014; Benabbou et Perny, 2017; Benabbou *et al.*, 2017].

Extension de la stratégie CSS pour la prise de décision sur domaine combinatoire. Les travaux cités plus haut traitent le cas de la résolution de problèmes définis sur domaines explicites. Dans ce cas, l'application de la stratégie CSS est simple, il suffit de calculer la valeur de regret maximum $\mathrm{MR}(x, \mathcal{X}, \mathcal{W})$ de toute solution $x \in \mathcal{X}$ et de comparer ces valeurs afin de déterminer une question ou une recommandation.

L'extension au domaine combinatoire n'est pas évidente puisque, d'une part, on se retrouve avec un nombre exponentiel de valeurs de regret à comparer, et d'autre part, le calcul de chaque regret défini par l'équation (1.13) relève maintenant de l'optimisation quadratique : par exemple, pour le cas simple d'une somme pondérée, il s'écrit $\mathrm{MR}(x, \mathcal{X}, \mathcal{W}) = \max_{y \in \mathcal{X}} \max_{\omega \in \mathcal{W}} \{\sum_{i=1}^{n} \omega_i u_i(y) - \sum_{i=1}^{n} \omega_i u_i(x)\}$, où ω *et* y sont toutes deux des variables.

Ce problème est étudié dans plusieurs travaux étendant la stratégie CSS au domaine combinatoire. Pour l'élicitation d'opérateurs f_ω linéaires en x et en ω, Boutilier *et al.* [2006a] étendent la stratégie CSS au cas combinatoire dans le cas de l'élicitation d'une fonction d'utilité. L'approche consiste à réduire progressivement l'incertitude concernant les valeurs d'utilité possibles en alternant calcul de regrets (en utilisant une approche par génération de contraintes) pour déterminer les questions à poser, et réduction de l'ensemble de valeurs d'utilité possibles en fonction des réponses données. La procédure s'arrête lorsque la valeur du MMR est nulle ou inférieure à un seuil prédéfini. On peut également citer des approches plus spécifiques, notamment pour l'élicitation des poids d'une somme pondérée : Benabbou et Perny [2015b] se sont intéressés à la résolution d'un problème de chemins multi-objectifs dans un graphe défini implicitement par une fonction successeur. Cette méthode utilise une variante de l'algorithme MOA* (Multi-Objective A*) pour alterner exploration de l'espace de recherche et élicitation des préférences fondée sur la stratégie CSS appliquée à des sous-ensembles explicites de solutions partielles (complétées de manière heuristique). Les réponses aux questions posées permettent de réduire l'espace des paramètres, ce qui permet de filtrer l'ensemble des solutions partielles et de guider le choix du prochain nœud à développer. Une autre méthode a été proposée pour un problème d'arbres couvrants multi-objectifs [Benabbou et Perny, 2015c] ; l'approche est similaire à la méthode précédente mais l'exploration se fait à l'aide d'une extension multi-objectifs de l'algorithme de Prim.

Notons que, même si on se focalise ici sur la stratégie CSS, la difficulté liée à l'élicitation sur domaine combinatoire n'est pas propre à cette stratégie. De manière générale, une stratégie de choix de la question détermine un critère pour évaluer la qualité informative d'une question, évalue *chaque* question possible, puis pose la question la plus informative selon ce critère (voir par exemple, [Gelain *et al.*, 2010; Weng et Zanuttini, 2013; Teso *et al.*, 2016]). Une prise en compte de l'ensemble des questions possibles est donc nécessaire, et peut s'avérer poser un problème épineux sur domaine combinatoire.

Extension de la stratégie CSS au cas d'une fonction non-linéaire (sur domaine combinatoire). Les méthodes citées dans le dernier paragraphe (qui font appel à des algorithmes constructifs, à la programmation linéaire, etc) tirent profit de la linéarité du modèle f_ω en x et en ω (linéarité de l'objectif de l'équation (1.12) et des contraintes induites par les réponses). Lorsque le modèle représentant les préférences du décideur est non-linéaire en x, aborder de front la résolution du problème sur domaine combinatoire et l'élicitation des paramètres est plus difficile. En effet, en plus de la difficulté de l'équation (1.13) qui relève de l'optimisation quadratique, le principe d'optimalité de Bellman n'est plus respecté du fait de l'opération de tri nécessaire au calcul de $f_\omega(x)$, ce qui entraîne deux difficultés principales : les méthodes de résolution standards de l'optimisation combinatoire ne sont plus valides, et l'élicitation doit nécessairement porter sur des solutions entières car on ne peut plus inférer une préférence sur des solutions complètes à partir

d'une préférence sur des solutions partielles.

Il existe dans la littérature très peu de travaux proposant des méthodes permettant de traiter à la fois la difficulté liée au domaine combinatoire et celle liée à la non-linéarité du modèle. On peut citer [Benabbou et Perny, 2015a] pour l'élicitation d'une intégrale de Choquet pour la résolution d'un problème de recherche dans un graphe à espace d'états. Le choix des questions se fait par la stratégie CSS appliquée à des sous-ensembles de solutions (on se ramène au cas d'ensembles définis explicitement). On peut également citer [Benabbou et Perny, 2016; Benade *et al.*, 2017] pour la détermination d'un sac à dos optimal en appliquant des règles de vote. Dans ces deux derniers travaux, la fonction d'agrégation est précisément connue et l'élicitation porte sur les valeurs de performances. Nos premières contributions visent à proposer des méthodes de résolution *génériques* permettant d'éliciter les *paramètres* d'une fonction d'agrégation non-linéaire pour la résolution d'un problème défini sur domaine combinatoire.

Extension de l'approche au cas de réponses incohérentes. Toutes les méthodes citées plus haut, qui utilisent une réduction de l'espace des paramètres, reposent sur l'hypothèse forte que le décideur est toujours dans la capacité de répondre de manière correcte aux questions posées, et ce même lorsque les solutions sont difficiles à comparer. Or, cette hypothèse n'est pas toujours vérifiée, des erreurs/incohérences peuvent se produire lorsque le décideur n'est pas d'une précision extrême et se trompe car les solutions proposées sont très proches et/ou difficiles à comparer, ou simplement lorsqu'il change d'avis au cours du temps. Dans ce cas, la réduction systématique et irréversible de l'espace des paramètres peut impacter significativement l'efficacité de la méthode d'élicitation. En effet, elle peut mener à deux situations différentes selon le type d'erreurs :

1. L'ensemble d'incertitude est vide ; l'algorithme se trouve alors dans l'incapacité de déterminer un paramètre compatible avec les informations préférentielles acquises. Cette situation se produit lorsque le décideur donne deux réponses contradictoires.

2. Le paramètre correspondant aux préférences réelles du décideur est exclu de l'espace des paramètres. Cette situation se produit lorsque le décideur donne une réponse qui est en contradiction avec ses préférences mais qui n'est pas en contradiction avec ses autres réponses. Dans ce cas, il est possible de trouver un paramètre (ou un ensemble de paramètres) compatible avec les réponses données, mais celui-ci peut modéliser des préférences très différentes de celles du décideur.

Selon la situation, le comportement de l'algorithme est très différent. Dans le premier cas (qui ne peut pas se produire avec la stratégie CSS, voir la sous-section 2.2.3), l'incohérence est trivialement détectée et aucune recommandation ne peut être faite sans remettre en question les informations préférentielles acquises. Dans le second, il est impossible de détecter une telle erreur, les préférences du décideur étant inconnues. L'algorithme continue donc normalement en prenant une mauvaise direction de recherche, il retourne alors une solution pouvant être de très mauvaise qualité. Pour remédier à ce problème, les méthodes suivantes s'intéressent à une modélisation plus souple et réversible de l'incertitude concernant les préférences du décideur.

c. Élicitation incrémentale tolérante aux erreurs fondée sur l'utilisation d'un modèle

Les problèmes liés à la présence d'erreurs sont dus à l'ajout *irréversible* des contraintes qui n'offre pas la possibilité de revenir sur un jugement de préférence exprimé antérieurement. En présence d'informations préférentielles erronées, la vraie valeur de paramètre (correspondant aux préférences du décideur et que l'on cherche à cerner), peut être rapidement écartée de l'espace des paramètres, sans possibilité de détecter cette situation ou d'y remédier. Ainsi, les méthodes procédant par réduction systématique de l'espace des paramètres sont très sensibles à la présence d'erreurs dans les réponses. Une première solution consiste à ajouter des variables d'écart aux contraintes induites par les informations préférentielles données par le décideur, et à ajouter à la fonction objectif un terme (généralement, la somme des écarts) pénalisant la valeur de l'objectif lorsque les écarts sont strictement positifs (de manière à faire tendre les écarts vers 0 autant que possible). Ces variables permettent d'introduire une marge d'erreur dans les informations préférentielles et offrent une certaine souplesse dans le respect des contraintes induites par ces informations. Cette approche est notamment utilisée par Branke *et al.* [2016] où les auteurs présentent non seulement une méthode qui a recours à des variables d'écart, mais définissent également un processus de suppression de contraintes lorsque l'espace des paramètres est vide malgré les variables d'écart. Néanmoins, ce processus repose sur une heuristique sur l'ordre dans lequel les contraintes sont supprimées. Cette solution n'est pas idéale car elle peut mener à la suppression de contraintes correspondant à des préférences valides.

Une autre solution, très souvent utilisée, consiste à avoir recours à une distribution de probabilité afin de modéliser l'incertitude concernant les réponses du décideur. Dans ce cas, lorsqu'une nouvelle information est acquise, la distribution de probabilité est mise à jour en utilisant cette information. Cette approche est illustrée sur la figure 1.9 où le carré rouge représente le paramètre (caché) modélisant les préférences du décideur.

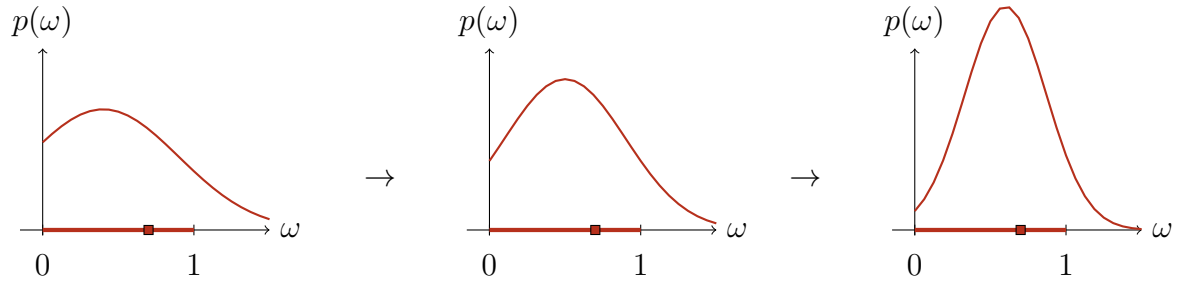


FIGURE 1.9 – Exemple de mise à jour d'une densité de probabilité.

De cette manière, les préférences du décideur sont prises en compte mais pas de manière irréversible, ce qui diminue significativement l'impact d'éventuelles réponses incohérentes sur la qualité de la recommandation finale. Cette approche a été utilisée dans différents cadres. Par exemple, pour la résolution de problèmes multi-attributs en présence de valeurs d'utilités imprécises, une méthode présentée par Chajewska *et al.* [2000] associe une distribution de probabilité aux conséquences des décisions possibles. Cette distribution de probabilité est mise à jour par des questions de comparaison de loteries du type : *"Quelle*

est l'option préférée parmi : une utilité u de manière sûre ou *une utilité $u_\top$ avec une probabilité p et une utilité $u_\bot$ avec une probabilité $1-p$?"* (où $u_\top$ et $u_\bot$ sont des valeurs prédéfinies indiquant respectivement la pire et la meilleure performance). Afin de choisir la question à poser, les auteurs évaluent l'intérêt de chaque question possible par une mesure, appelée *valeur d'information*, qui prend en compte la distribution de probabilité courante. Elle consiste à évaluer, pour chaque question, l'amélioration espérée de la qualité d'une décision après avoir obtenu une réponse à cette question. Ce travail ne traite pas la problématique de la mise à jour de la distribution de probabilité, des techniques de mise à jour Bayésienne de distributions Gaussiennes sont utilisées [Bishop, 2006]. Plus tard, Guo et Sanner [2010] ont proposé une extension de ce travail où la charge cognitive du décideur est réduite grâce à de l'utilisation de questions plus faciles. Le décideur doit comparer des paires de solutions et ne donner un ordre de préférence que s'il en est certain : il donne un ordre de préférence entre deux solutions lorsqu'il est certain de sa préférence car leur différence d'utilité est relativement grande, et déclare être indifférent aux deux solutions si la préférence entre les deux est trop peu certaine du fait de la proximité de leur valeur de performances. Dans un cadre d'optimisation multi-objectifs, Sauré et Vielma [2019] présentent une méthode utilisant une distribution de probabilité Gaussienne afin de définir une *région ellipsoïdale de confiance*, qui est mise à jour après chaque réponse donnée par le décideur. Une telle mise à jour est illustrée par la figure 1.10 [6], où l'ellipsoïde bleue représente l'ellipsoïde avant de poser la question représentée par un hyperplan vertical, et où l'ellipsoïde en rouge pointillée représente l'ellipsoïde a posteriori (après mise à jour Bayésienne).

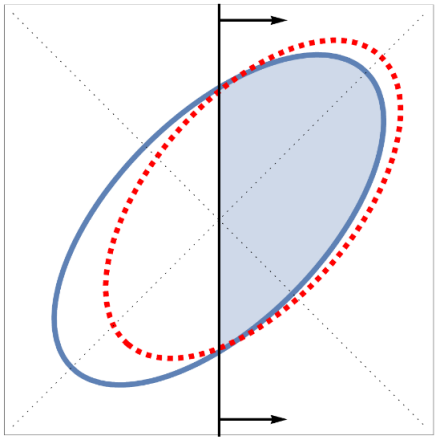


FIGURE 1.10 – Exemple de mise à jour d'une ellipsoïde de confiance.

Plus récemment, Guillot et Destercke [2019] ont présenté une méthode étendant la stratégie CSS au cadre de l'utilisation de fonctions de croyances pour modéliser l'incertitude concernant les préférences du décideur. Une autre méthode est proposée par Vendrov *et al.* [2020] pour la résolution de problèmes multi-attributs définis sur domaine explicite mais contenant un très grand nombre de solutions (de l'ordre de centaines de milliers). Dans cette méthode, les auteurs utilisent une relaxation de l'espace des solutions, ce qui permet de pouvoir utiliser une approche de gradient. Cette méthode utilise le critère de la *Valeur d'Information Espérée* ou encore EVOI (pour *Expected Value Of Information*) pour

6. Figure empruntée à [Sauré, 2016].

le choix de la question à poser. Ce critère consiste à déterminer une question qui maximise l'utilité espérée d'une recommandation conditionnellement à la réponse du décideur. Ils proposent également une stratégie du choix de la question fondée sur des comparaisons de solutions partielles, ce qui réduit l'effort cognitif imposé au décideur lorsque l'ensemble d'attributs est grand.

Une approche différente consiste à utiliser un *processus Gaussien*. On suppose qu'il existe une fonction inconnue f, associant une valeur $f(x)$ à toute solution x, qui modélise les préférences du décideur. Pour approximer cette fonction et déduire le comportement décisionnel qu'elle reflète, on suppose que la valeur $f(x)$ associée à tout x est une variable aléatoire suivant une distribution Gaussienne, sa moyenne donnant la valeur espérée de $f(x)$, et la variance exprimant l'incertitude concernant cette même valeur. Le processus est utilisé à la fois pour déterminer une recommandation et pour déterminer une question à poser au décideur. A chaque nouvelle information, le processus Gaussien est mis à jour de manière Bayésienne, réduisant peu à peu l'incertitude concernant les valeurs $f(x), x \in \mathcal{X}$. La figure 1.11 [7] illustre le fonctionnement d'un processus Gaussien. Sur cette figure, la courbe noire pointillée donne la fonction f (cachée) que l'on souhaite éliciter et la surface en bleu donne la région de confiance déterminée par le processus Gaussien, à partir de la moyenne et de variance en chaque point, à l'aide des informations préférentielles données ici par des points en rouge. Finalement, la courbe bleue donne l'estimation de f en utilisant le processus Gaussien.

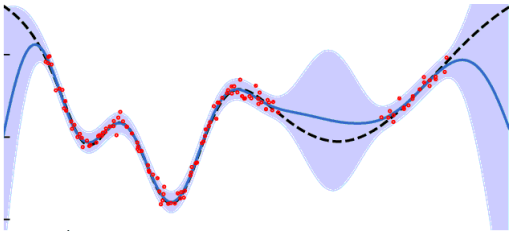


FIGURE 1.11 – Exemple d'estimation d'une fonction f à l'aide d'un processus Gaussien.

Différents travaux utilisent des processus Gaussiens, par exemple [Houlsby *et al.*, 2012; Zintgraf *et al.*, 2018]. La différence principale avec les méthodes précédentes est qu'aucune hypothèse n'est faite sur la forme de la fonction f et que l'élicitation ne porte pas sur la définition de f mais uniquement sur les valeurs $f(x), x \in \mathcal{X}$.

Notons que les méthodes présentées concernent toutes l'élicitation d'une fonction linéaire sur domaine explicite (à l'exclusion des processus Gaussiens qui ne font aucune hypothèse sur f). Nos contributions dans ce cadre consistent à traiter des cas non encore étudiés (à notre connaissance), et qui concernent l'élicitation (tolérante aux erreurs) des paramètres d'une fonction non-linéaire et/ou la résolution de problèmes définis sur domaine combinatoire.

1.4 Conclusion du chapitre

Dans ce chapitre nous avons présenté un bref état de l'art concernant la modélisation et l'élicitation des préférences dans le cadre de la prise de décision multi-objectifs. Dans

7. Figure empruntée à [Leclercq, 2018].

un premier temps, nous avons présenté la problématique de choix multi-objectifs. Nous avons ensuite présenté la méthode standard pour résoudre un tel problème en théorie de la décision, et qui consiste à utiliser un modèle décisionnel qui agrège les évaluations des solutions afin de se ramener à un problème d'optimisation mono-objectif. Cette approche permet non seulement de simplifier la formulation et la résolution du problème, mais également de simplifier la procédure d'élicitation des préférences car la spécification du modèle de décision permet de déduire un grand nombre de relations de préférences autres que celles apportées par le décideur. Il existe dans la littérature de nombreux modèles permettant de représenter des préférences très différentes : nous avons présenté les différents modèles auxquels nous allons nous intéresser dans cette thèse. Ces modèles sont tous caractérisés par des paramètres permettant de modéliser une grande variété de comportements décisionnels différents, et ainsi de s'adapter aux préférences particulières de chaque décideur. Nous avons ensuite présenté un état de l'art concernant les méthodes d'élicitation des préférences permettant de suffisamment réduire l'incertitude concernant la valeur des *paramètres* pour pouvoir déterminer une recommandation personnalisée qui puisse satisfaire le décideur. Même si on ne s'intéresse dans cette thèse qu'à l'élicitation des préférences du décideur, qui se traduit par l'élicitation des paramètres du modèle décisionnel et non des valeurs de performances, nous avons également présenté des méthodes concernant l'élicitation des valeurs de performances, car ces deux sujets se complètent et se mélangent parfois dans la littérature (par exemple, l'utilisation de la stratégie CSS, initialement conçue pour l'élicitation des valeurs de performances, a ensuite été employée pour l'élicitation des valeurs de paramètres). Nous avons pu voir l'intérêt de l'utilisation d'un modèle pour simplifier l'élicitation des préférences d'une part, ainsi que pour guider la recherche d'une solution optimale d'autre part.

Dans la suite de ce document, nous présentons l'essentiel des travaux effectués durant cette thèse, et qui consistent à étudier des problématiques peu traitées (voir non traitées) dans la littérature. On introduit alors de nouvelles méthodes de résolution *génériques* permettant de traiter des problèmes combinant plusieurs des difficultés principales suivantes :

1. le paramètre du modèle de décision n'est pas précisément connu,
2. l'ensemble des solutions réalisables est défini sur domaine combinatoire,
3. les préférences du décideur sont complexes et nécessitent l'utilisation d'un modèle de décision non-linéaire,
4. les réponses du décideur contiennent des erreurs et/ou des incohérences.

Dans les chapitres 2 et 3 on s'intéresse à la combinaison des trois premières difficultés, tandis que les chapitres 4 et 5 concernent la combinaison des difficultés 1 et 4 avec l'une des deux problématiques restantes.

Chapitre 2

Élicitation des préférences par réduction de l'espace des paramètres

Résumé du chapitre

Ce chapitre est consacré à la conception d'algorithmes d'élicitation incrémentale des préférences pour la décision collective ou dans le risque sur domaine combinatoire. On suppose ici que les préférences du décideur sont modélisées par des opérateurs non-linéaires tels que les sommes pondérées ordonnées (OWA pour Ordered Weighted Averages) et les sommes doublement pondérées ordonnées (WOWA pour Weighted OWA). Ces opérateurs sont paramétrés par des vecteurs ou des fonctions de pondération qui permettent un contrôle fin de l'exigence d'équité ou de robustesse et qui peuvent donc permettre de modéliser correctement les préférences du décideur. Les méthodes proposées ici procèdent en alternant exploration de solutions potentiellement intéressantes pour le décideur et questions permettant des réductions systématiques de l'espace des paramètres possibles. Les travaux présentés dans ce chapitre sont issus de deux publications : [Bourdache et Perny, 2017] pour l'élicitation des paramètres d'OWA ou de WOWA pour la résolution des problèmes d'affectation et de plus court chemin robustes, et [Bourdache et Perny, 2019] pour l'élicitation des paramètres d'OWA pour la résolution du problème du sac à dos multi-agents.

2.1 Introduction

On s'intéresse dans ce chapitre à la résolution de problèmes multi-objectifs définis sur domaine combinatoire. On se place dans un cadre où les préférences du décideur qui supervise la prise de décision ne sont pas précisément connues mais que l'on sait être d'une nature complexe : exigence d'équité ou aversion au risque. On modélise alors ces préférences à l'aide de modèles non-linéaires capables de traduire de tels comportements tels qu'OWA (pour Ordered Weighted Average) ou WOWA (pour Weighted OWA). Ces opérateurs sont paramétrables de manière à s'adapter aux préférences spécifiques d'un décideur particulier puisqu'ils permettent un contrôle fin du degré d'équité ou de robustesse imposé dans la solution recommandée. Les préférences du décideur n'étant pas précisément connues, la valeur de ces paramètres ne peut être fixée a priori, on travaille alors sur l'*ensemble d'incertitude* contenant tous les paramètres possibles. Cet ensemble est ensuite réduit au fur et à mesure de l'acquisition de nouvelles informations préférentielles afin de ne garder que les valeurs de paramètres compatibles avec ces informations. Dans ce cadre, les méthodes de résolution à concevoir doivent permettre d'extraire des solutions intéressantes malgré une connaissance partielle de la valeur des paramètres. On devra également concevoir une procédure d'élicitation efficace qui nous permet de réduire rapidement l'espace des paramètres de manière à pouvoir présenter au décideur une recommandation pertinente tout en veillant à ne pas lui imposer une charge cognitive trop importante.

La conception de ce type de méthodes constitue un défi important puisque plusieurs difficultés sont à traiter simultanément : 1. l'ensemble des solutions réalisables est défini implicitement par des contraintes linéaires et peut contenir un nombre exponentiel d'éléments, 2. les paramètres des modèles considérés sont également des variables du problème, l'optimisation d'OWA ou de WOWA relève alors, au mieux, de la programmation quadratique, et 3. les modèles que l'on utilise offrent, du fait de leur non-linéarité, une capacité descriptive des préférences bien supérieure à celle de modèles linéaires, mais cette non-linéarité entraîne des difficultés supplémentaires telles que le non respect du principe d'optimalité de Bellman qui, d'une part nous empêche d'utiliser des méthodes constructives, de programmation linéaire ou de programmation dynamique, et d'autre part contraint la procédure d'élicitation par le fait qu'une information préférentielle sur des sous-solutions ne nous apporte aucune information préférentielle sur l'évaluation des solutions complètes. Ces trois points seront explicités plus concrètement à la fin de la section 2.2.

Il existe dans la littérature différents travaux portant sur la prise de décision multi-objectifs avec une connaissance partielle des préférences. La plupart de ces travaux traitent une ou deux des trois difficultés énoncées ci-dessus : élicitation d'un modèle de décision linéaire ou non pour la résolution d'un problème défini sur domaine explicite (difficultés 2. et/ou 3.), e.g., [White III *et al.*, 1984; Ha et Haddawy, 1997; Chajewska *et al.*, 2000; Wang et Boutilier, 2003; Braziunas et Boutilier, 2007; Perny *et al.*, 2016; Benabbou *et al.*, 2017], élicitation d'une fonction linéaire pour la résolution d'un problème défini sur domaine combinatoire (difficultés 1. et 2.), par exemple, [Boutilier *et al.*, 2006b; Weng et Zanuttini, 2013; Benabbou et Perny, 2015b, 2017; Kaddani *et al.*, 2017; Brero *et al.*, 2018]. Néanmoins, peu de travaux traitent les trois difficultés à la fois : la méthode introduite dans [Benabbou et Perny, 2015a] traite le problème d'élicitation incrémentale pour la recherche d'une solution Choquet-optimale dans un graphe à espace d'états, mais

utilise une méthode d'exploration de l'espace des solutions spécifique au problème traité. Une autre méthode proposée dans [Benade *et al.*, 2017] permet une approximation du sac à dos multi-agents optimal mais s'intéresse à l'élicitation des *utilités individuelles* des agents en supposant que la valeur des paramètres de l'opérateur d'agrégation est précisément connue. Plus récemment, [Benabbou *et al.*, 2020] proposent une méthode de résolution approchée utilisant une approche d'élicitation fondée sur une exploration de l'espace de recherche en utilisant un algorithme génétique. Nos contributions visent à enrichir le pan de la littérature, peu étudié, concernant l'élicitation d'un modèle de décision *non-linéaire* pour la résolution de problèmes multi-objectifs définis sur *domaine combinatoire*. On suppose ici que les valeurs de performances (utilités individuelles des agents, valeurs des critères ou évaluations selon différents scénarios) sont connues de manière précise et que l'on élicite les *paramètres* des opérateurs d'agrégation. Notre objectif est de concevoir des algorithmes *génériques* permettant de déterminer des solutions optimales ou quasi-optimales avec garantie de performance. La définition de ces algorithmes sera ensuite adaptée à la résolution de certains problèmes peu traités dans ce contexte : le problème d'affectation robuste, le problème du plus court chemin robuste ainsi que le problème du sac à dos multi-agents.

Contexte et notations. On considère dans ce chapitre que l'on résout un problème où l'espace des solutions $\mathcal{X}$ est défini par un ensemble de contraintes linéaires (à l'exception de la section 2.2 où il est défini explicitement). Chaque solution de cet ensemble est évaluée selon n fonctions objectifs : point de vue de n agents ou évaluation selon n scénarios. L'image de $\mathcal{X}$ dans l'espace des critères est donnée par $\mathcal{Y} = \{u(x) \in \mathbb{R}^n, \forall x \in \mathcal{X}\}$ où $u(x)$ est le vecteur de performance associé à la solution x et dont la i^e composante représente la performance de x selon la fonction objectif i. On se place dans un cadre où aucune information concernant les préférences du décideur n'est connue à l'exception d'une exigence plus ou moins forte d'équité ou de robustesse. On modélise alors les préférences du décideur par un OWA à poids décroissants ou par l'opérateur WOWA avec une fonction de déformation croissante et convexe (ces restrictions sur les paramètres permettent de privilégier les solutions associées à des vecteurs de performance équilibrés cf. paragraphes dédiés à OWA et WOWA section 1.2.2). Dans le cas de l'utilisation d'un OWA pour modéliser les préférences, on définit l'ensemble d'incertitude par l'ensemble de vecteurs de pondération $W = \{w \in \mathbb{R}^n_+ | \sum_{i=1}^n w_i = 1, w_1 \geq \cdots \geq w_n\}$. Dans le cas de l'utilisation de WOWA pour la modélisation des préférences, l'ensemble d'incertitude définissant l'ensemble des fonctions de déformation possibles est défini par $\Phi = \{\varphi : [0,1] \to [0,1] | \varphi$ continue, croissante, convexe et telle que $\varphi(0) = 0$ et $\varphi(1) = 1\}$. Les questions que l'on pose au décideur pour éliciter w ou φ consistent à lui demander de comparer des paires de solutions que l'on choisit, le plus souvent, par une adaptation de la stratégie *CSS* que l'on rappelle plus loin. Ces comparaisons permettent de réduire l'incertitude sur les préférences par le biais de contraintes linéaires réduisant systématiquement et irréversiblement l'espace des paramètres. Sauf mention contraire, les algorithmes présentés sont décrits dans le cadre d'une maximisation de l'opérateur considéré.

Organisation du chapitre. On présente tout d'abord, dans la section 2.2, les notions de base de la résolution d'un problème avec une connaissance partielle des préférences du décideur. On présente un algorithme simple, issu de l'état de l'art, permettant de calculer

des solutions intéressantes pour le décideur lorsque $\mathcal{X}$ est défini sur domaine explicite et que le modèle de préférences utilisé est linéaire en son paramètre. On montre que cet algorithme peut être simplement appliqué à un modèle comme OWA, qui est non-linéaire (en les solutions x) mais linéaire en son paramètre, mais pas à un modèle non-linéaire en son paramètre comme WOWA. On introduit ensuite une nouvelle formulation de la fonction φ de WOWA afin d'adapter la méthode à cet opérateur. Finalement, on montre comment alterner cet algorithme avec un processus d'élicitation incrémentale afin de déterminer une recommandation pertinente, puis on énonce les principales difficultés rencontrées lors de l'extension de cette méthode au domaine combinatoire. La section 2.3 est consacrée à la présentation de nouvelles méthodes permettant de pallier ces difficultés. On y résout le problème traité en effectuant une exploration partielle et méthodique de l'espace des solutions par un algorithme de *ranking* (énumération ordonnée) ou de *branch and bound* (séparation et évaluation). Nous présentons également des résultats numériques montrant l'efficacité pratique de nos méthodes. Enfin, la section 2.4 conclut le chapitre.

2.2 Solutions potentiellement optimales et élicitation des préférences avec les modèles OWA et WOWA

Lorsque la connaissance des préférences du décideur est partielle, on s'intéresse souvent aux notions d'*optimalité potentielle* et d'*optimalité nécessaire*. L'ensemble de solutions *potentiellement optimales* est défini par l'ensemble des solutions réalisables du problème qui sont optimales pour au moins une instanciation possible des valeurs des paramètres. La notion d'*optimalité potentielle* est un raffinement de la dominance de Pareto qui nous permet d'extraire de l'ensemble de solutions $\mathcal{X}$ un sous-ensemble de solutions que l'on sait intéressant pour le décideur au vu des informations préférentielles acquises. Une procédure d'élicitation incrémentale nous permet, par l'acquisition de nouvelles informations préférentielles, de réduire l'ensemble d'incertitude et de mieux discriminer entre les solutions potentiellement optimales jusqu'à aboutir à une ou plusieurs solutions *nécessairement optimales*, i.e., un ensemble de solutions optimales quelle que soit l'instanciation de valeurs des paramètres considérée (au vu des informations acquises). Cet ensemble n'existe qu'une fois que l'on a acquis suffisamment d'informations sur les préférences du décideur pour que l'ensemble des solutions potentiellement optimales soit réduit à un seul élément (ou plusieurs dont l'image dans l'espace des critères agrégés est la même). On doit alors mettre au point une procédure d'élicitation incrémentale efficace qui nous permet de déterminer ces solutions en posant le moins de questions possible au décideur.

2.2.1 Calcul de solutions potentiellement OWA-optimales sur domaine explicite

On suppose ici que l'ensemble de solutions $\mathcal{X}$ est défini explicitement et que l'on modélise les préférences du décideur à l'aide de l'opérateur OWA (cf. définition 1.10). Pour un vecteur de pondération $w \in W$ donné, toute solution $x \in \mathcal{X}$ associée au vecteur de performances $u(x)$ est donc évaluée par :

$$\mathrm{OWA}_w(x) = \sum_{i=1}^{n} w_i u_{(i)}(x)$$

où $u_{(i)}$ est la i^e plus petite composante du vecteur de performances $u(x)$ et le vecteur de pondération w modélise le compromis que fait le décideur entre utilitarisme (moyenne) et égalitarisme (minimum).

On définit l'ensemble des solutions *potentiellement OWA-optimales* comme l'ensemble des solutions de $\mathcal{X}$ qui sont OWA_w-optimales pour au moins un paramètre possible $w \in W$. Plus formellement, on donne la définition suivante (adaptation de la définition de PO pour la minimisation d'une somme pondérée [Benabbou et Perny, 2015b]) :

Définition 2.1.

$$PO_W(\mathcal{X}) = \bigcup_{w \in W} \arg\max_{x \in \mathcal{X}} \text{OWA}_w(x)$$

Exemple 2.1. *On définit l'ensemble de solutions* $\mathcal{X} = \{x_1, ..., x_5\}$ *dont l'image dans l'espace bicritère est donnée par :* $\mathcal{Y} = \{(6,8), (11,5), (4,18), (5,3), (22,2)\}$. *L'ensemble d'incertitude est donné par* $W = \{w \in \mathbb{R}^2_+ | w_1 + w_2 = 1, w_1 \geq w_2\}$. *On représente sur la figure 2.1 la valeur* $y_i = \text{OWA}_w(x_i), i \in \{1, ..., 5\}$, *en fonction de* w_1 *(comme* $w_2 = 1 - w_1$, *on peut exprimer* $\text{OWA}_w(x)$ *uniquement en fonction de* w_1*) :*

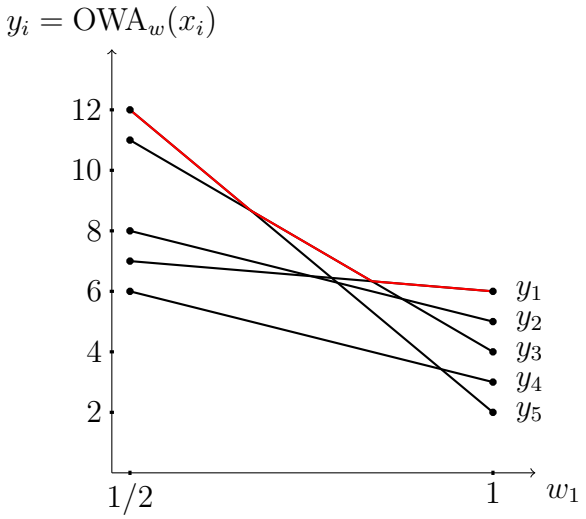


FIGURE 2.1 – Images $\text{OWA}_w(x^i)$ de $x^i \in \mathcal{X}$ et $\bigcup_{w \in W} \max_{x^i \in \mathcal{X}} \text{OWA}_w(x^i)$ (en rouge).

On peut voir graphiquement que les segments en rouge représentent $\max_{x \in \mathcal{X}} \text{OWA}_w(x)$ *pour toute valeur* $w_1 \in [\frac{1}{2}, 1]$. *L'ensemble des solutions potentiellement optimales est alors* $PO_W(\mathcal{X}) = \{x_1, x_3, x_5\}$.

Puisque $\mathcal{X}$ est un ensemble de solutions défini explicitement, $PO_W(\mathcal{X})$ peut être calculé par simple programmation linéaire grâce aux relations de $W-$*dominance* et de $W-$*dominance d'ensembles* définies par (adaptation des définitions de Benabbou et Perny [2015b]) :

Définition 2.2. *Soient* x *et* y *deux solutions de* $\mathcal{X}$*, on a :*

$$x \succ_W y \iff \forall w \in W, \text{OWA}_w(x) > \text{OWA}_w(y)$$

on dit alors que x $W-$*domine* y *au sens de l'opérateur* OWA.

Autrement dit, une solution x $W-$domine une solution y si l'évaluation de x selon l'opérateur OWA est meilleure que celle de y pour toute instanciation possible du paramètre w. Cette relation peut être généralisée en une relation de $W-$dominance d'ensembles :

Définition 2.3. *Soient $Y \subseteq \mathcal{X}$ et $Z \subseteq \mathcal{X}$ deux sous-ensembles de solutions, on a :*

$$Y \succ_W Z \iff \forall z \in Z, \forall w \in W,\ \exists y \in Y, \mathrm{OWA}_w(y) > \mathrm{OWA}_w(z)$$

on dit que l'ensemble Y $W-$domine l'ensemble Z au sens de l'opérateur OWA.

Notons que lorsqu'un ensemble de solutions Z est $W-$dominé par un ensemble de solutions Y, toute solution $z \in Z$ est $W-$dominée par l'ensemble Y. Cette relation est plus discriminante que la relation de $W-$dominance simple, puisqu'une solution z $W-$dominée par l'ensemble Y est comparée à une solution fictive constituée de la meilleure partie de chaque solution de Y (par exemple les segments rouges de la figure 2.1 issus des solutions x_1, x_3 et x_5). De ce fait, la solution z peut n'être $W-$dominée par aucune solution particulière de Y mais être $W-$dominée par Y comme le montre l'exemple suivant :

Exemple 2.1 (suite). *Dans l'exemple précédent, on peut voir graphiquement (figure 2.1) que la solution x_4 est $W-$dominée par chacune des solutions x_1, x_2 et x_3. On peut également voir que même si la solution x_2 n'est $W-$dominée par aucune solution de $\mathcal{X}$, le singleton $\{x_2\}$ est $W-$dominé par l'ensemble $\{x_1, x_3, x_5\}$ car pour toute valeur de w_1, x_2 sera dominée par x_1, x_3 ou x_5.*

On peut voir à travers cet exemple, que la relation de $W-$dominance d'ensembles peut être utilisée pour déterminer si une solution x est potentiellement OWA-optimale ou non en vérifiant si elle est $W-$dominée par l'ensemble $\mathcal{X}$, i.e., si $\mathcal{X} \succ_W \{x\}$. Notons que $\mathcal{X}$ n'est pas forcément l'ensemble minimal qui $W-$domine $\{x\}$ mais la détermination de l'ensemble minimal n'est pas toujours simple. On utilise alors la totalité de l'ensemble $\mathcal{X}$ puisque, par définition de la dominance d'ensembles, si un sous-ensemble $\mathcal{X}' \subseteq \mathcal{X}$ $W-$domine $\{x\}$ alors $\mathcal{X}$ $W-$domine $\{x\}$ également.

La proposition ci-dessous justifie l'utilisation de la $W-$dominance d'ensembles pour le calcul de $PO_W(\mathcal{X})$ (adaptation de la Proposition 1. de Benabbou et Perny [2015b] donnée dans le cadre de la minimisation d'une somme pondérée) :

Proposition 2.1. *S'il existe deux ensembles $Y, Z \subseteq \mathcal{X}$ tels que $Y \succ_W Z$ alors $Z \cap PO_W(\mathcal{X}) = \emptyset$.*

Démonstration. Considérons deux sous-ensembles $Y, Z \subseteq \mathcal{X}$ tels que $Y \succ_W Z$. On sait, d'après la définition 2.3, que pour tout $z \in Z$ et pour tout vecteur de pondération $w \in W$, il existe une solution $y \in Y$ telle que $\mathrm{OWA}_w(y) > \mathrm{OWA}_w(z)$. De ce fait, aucun élément $z \in Z$ ne peut être OWA-optimal quel que soit le paramètre $w \in W$. Par conséquent, aucun élément de Z n'appartient à l'ensemble $PO_W(\mathcal{X})$. □

On sait donc que, si x est une solution $W-$dominée par l'ensemble $\mathcal{X}$, elle n'est pas potentiellement optimale et peut être écartée de $\mathcal{X}$. Le calcul de $PO_W(\mathcal{X})$ se fait alors par éliminations successives de toute solution de $\mathcal{X}$ qui est $W-$dominée par $\mathcal{X}$. On parle alors d'algorithme de filtrage, l'approche est détaillée dans l'algorithme 2.1 [Benabbou et Perny, 2015b] :

Algorithme 2.1 : Filtrer($\mathcal{X}$, W)

Entrées : $\mathcal{X}$: ensemble de solutions ; W : ensemble d'incertitude.

1 **pour** $x \in \mathcal{X}$ **faire**
2 **si** $\mathcal{X} \succ_W \{x\}$ **alors**
3 $\mathcal{X} \leftarrow \mathcal{X} \setminus \{x\}$
4 **fin**
5 **fin**
6 **retourner** $\mathcal{X}$

L'algorithme 2.1 a été introduit dans le cadre de la minimisation d'une somme pondérée mais s'adapte facilement à la maximisation de l'opérateur OWA. Notons que l'ordre de parcours des solutions (ligne 1) peut être arbitrairement choisi sans impacter l'ensemble des solutions retourné (Lemme 1. [Benabbou et Perny, 2015b]).

Exemple 2.1 (suite) *Appliquons l'algorithme 2.1 à l'ensemble $\mathcal{X}$ en parcourant les solutions x_i de $\mathcal{X}$ dans l'ordre croissant des indices i : on peut voir graphiquement que la solution x_1 est optimale dans l'intervalle $[5/6, 1]$ et n'est donc W-dominée par aucune solution ni aucun ensemble, on la garde donc dans l'ensemble $\mathcal{X}$. La solution x_2 est $W-$dominée par l'ensemble $\{x_1, x_3, x_5\}$ et donc $W-$dominée par $\mathcal{X}$, on l'élimine alors de $\mathcal{X}$. La prochaine solution que l'on élimine est la solution x_4 car, par exemple, $x_1 \succ_W x_4$. On trouve donc : $PO_W(\mathcal{X}) = \{x_1, x_3, x_5\}$, ce qui correspond aux solutions que l'on a trouvé graphiquement.*

La vérification de la condition $\mathcal{X} \succ_W \{x\}$ (que l'on fait graphiquement dans l'exemple) peut se faire simplement par programmation linéaire. En effet, cette relation de préférence se traduit, par définition de la relation de dominance, par :

$$\forall w \in W, \exists y \in \mathcal{X}, \mathrm{OWA}_w(y) - \mathrm{OWA}_w(x) > 0$$

ce qui est équivalent à :

$$\forall w \in W \max_{y \in \mathcal{X}} \mathrm{OWA}_w(y) - \mathrm{OWA}_w(x) > 0$$

ou encore

$$\min_{w \in W} \max_{y \in \mathcal{X}} \mathrm{OWA}_w(y) - \mathrm{OWA}_w(x) > 0$$

ce qui revient à vérifier le signe de la valeur optimale du programme suivant :

$$(P_{\mathrm{OWA},x}) : \begin{cases} \min t \\ t \geq \mathrm{OWA}_w(y) - \mathrm{OWA}_w(x) \quad \forall y \in \mathcal{X} \\ w \in W \end{cases} \tag{2.1}$$

$(P_{\mathrm{OWA}_w,x})$ définit un programme linéaire car les contraintes $t \geq \mathrm{OWA}_w(y) - \mathrm{OWA}_w(x)$ sont linéaires puisque x et y sont fixées, et puisque OWA_w est linéaire en w. De plus, l'ensemble W est également défini par des contraintes linéaires. En effet, W est initialement défini par des contraintes linéaires (positivité des composantes, normalisation et

contraintes des composantes décroissantes) définissant ainsi un polyèdre convexe. L'ajout de toute contrainte induite par la comparaison d'une paire de solutions conserve alors la convexité de W car ces contraintes, de la forme $\text{OWA}_w(x) \geq \text{OWA}_w(y), x, y \in \mathcal{X}$, sont linéaires en w.

2.2.2 Calcul de solutions potentiellement WOWA-optimales sur domaine explicite

Lorsque les préférences du décideur sont représentées par l'opérateur WOWA, le paramètre modélisant les préférences du décideur est la fonction de déformation φ (cf. définition 1.11). On rappelle que l'évaluation d'une solution $x \in \mathcal{X}$ selon WOWA, pour une fonction φ donnée, s'écrit :

$$\text{WOWA}\varphi(x) = \sum_{i=1}^{n} [u_{(i)}(x) - u_{(i-1)}(x)]\varphi(\sum_{k=i}^{n} p_{(k)})$$

où $u_{(i)}$ est la i^e plus petite composante du vecteur de performances $u(x)$ et la fonction φ modélise le comportement décisionnel du décideur.

On définit alors l'ensemble des solutions potentiellement optimales comme l'ensemble des solutions étant WOWA-optimales pour au moins une fonction de déformation possible $\varphi \in \Phi$:

Définition 2.4.

$$PO_\Phi(\mathcal{X}) = \bigcup_{\varphi \in \Phi} \arg\max_{x \in \mathcal{X}} \text{WOWA}_\varphi(x)$$

On peut également définir les relations de $\Phi-$dominance et de $\Phi-$dominance d'ensembles de la même manière que pour OWA :

Définition 2.5. *Soient x et y deux solutions de $\mathcal{X}$, on a :*

$$x \succ_\Phi y \iff \forall \varphi \in \Phi, \text{WOWA}_\varphi(x) > \text{WOWA}_\varphi(y)$$

on dit que la solution x $\Phi-$domine la solution y au sens de WOWA.

Définition 2.6. *Soient $Y \subseteq \mathcal{X}$ et $Z \subseteq \mathcal{X}$ deux sous-ensembles de solutions, on a :*

$$Y \succ_\Phi Z \iff \forall z \in Z, \forall \varphi \in \Phi, \exists y \in Y, \text{WOWA}_\varphi(y) > \text{WOWA}_\varphi(z)$$

on dit que l'ensemble Y $\Phi-$domine l'ensemble Z au sens de WOWA.

De la même manière que pour OWA, on peut utiliser la proposition suivante pour montrer que la relation de $\Phi-$dominance d'ensembles permet de déterminer l'ensemble $PO_\Phi(\mathcal{X})$:

Proposition 2.2. *S'il existe deux ensembles $Y, Z \subseteq \mathcal{X}$ tels que $Y \succ_\Phi Z$ alors $Z \cap PO_\Phi(\mathcal{X}) = \emptyset$.*

Démonstration. Considérons deux sous-ensembles $Y, Z \subseteq \mathcal{X}$ tels que $Y \succ_\Phi Z$. On sait, d'après la définition 2.6, que pour tout $z \in Z$, pour toute fonction $\varphi \in \Phi$, il existe une solution $y \in Y$ telle que $\text{WOWA}_\varphi(y) > \text{WOWA}_\varphi(z)$. De ce fait, aucun élément $z \in Z$ ne peut être WOWA-optimal quelle que soit la fonction $\varphi \in \Phi$ considérée. Par conséquent, aucun élément de Z n'appartient à l'ensemble $PO_\Phi(\mathcal{X})$. □

On peut alors tenter d'utiliser l'algorithme 2.1 pour éliminer successivement de $\mathcal{X}$ toute solution $\Phi-$dominée par $\mathcal{X}$ en remplaçant la condition $\mathcal{X} \succ_W \{x\}$ (ligne 2 de l'algorithme) par la condition $\mathcal{X} \succ_\Phi \{x\}$. Néanmoins, le programme permettant de vérifier cette relation pour une solution x donnée :

$$(P_{\text{WOWA}_\varphi,x}) : \begin{cases} \min t \\ t \geq \text{WOWA}_\varphi(y) - \text{WOWA}_\varphi(x) \quad \forall y \in \mathcal{X} \\ \varphi \in \Phi \end{cases}$$

ne représente pas un programme linéaire car φ n'est généralement pas une fonction linéaire (notamment dans le cas étudié ici où elle est convexe) et l'ensemble Φ n'est pas représenté par un polyèdre convexe. En effet, Φ est un ensemble continu de fonctions sur lequel il est difficile d'optimiser une fonction directement.

On propose alors d'utiliser une formulation différente de la fonction φ. Afin d'obtenir une formulation linéaire, nous allons l'exprimer comme une combinaison convexe des fonctions d'une *base génératrice* constituée de fonctions particulières. Supposons que l'on dispose d'une telle base génératrice et notons $s_1, \ldots, s_m$ les m fonctions de cette base. Soit $\alpha \in \mathbb{R}_+^m$ un vecteur de poids positifs tel que $\sum_{i=1}^m \alpha_i = 1$. La fonction φ s'écrit alors sous la forme :

$$\varphi(r) = \sum_{i=1}^{m} \alpha_i s_i(r), \forall r \in [0,1] \tag{2.2}$$

ce qui correspond à une formulation linéaire en α. Ici, la définition de la fonction φ dépend entièrement des coefficients α_i qui modélisent maintenant les préférences du décideur et sont donc les nouveaux paramètres à éliciter. Le nouvel espace des paramètres $\mathcal{A}$ est défini par l'ensemble des vecteurs de coefficients possibles, i.e. $\mathcal{A} = \{\alpha \in \mathbb{R}_+^m | \sum_{i=1}^m \alpha_i = 1\}$.

L'intérêt de cette formulation est : 1. que φ est linéaire en α, ce qui implique que l'expression $\text{WOWA}_\varphi(y) - \text{WOWA}_\varphi(x), x \in \mathcal{X}, y \in \mathcal{X}$ est linéaire en α. De ce fait, les contraintes $t \geq \text{WOWA}_\varphi(y) - \text{WOWA}_\varphi(x), \forall y \in \mathcal{X}$, du programme $(P_{\text{WOWA}_\varphi,x})$ sont maintenant linéaires. 2. l'ensemble des paramètres possibles $\mathcal{A}$ est représenté par un polyèdre convexe. En effet, $\mathcal{A}$ est initialement défini par le simplexe qui est un polyèdre convexe et, par 1., les contraintes induites par des comparaisons de paires de solutions (sous la forme $\text{WOWA}_\varphi(y) - \text{WOWA}_\varphi(x) \geq 0, x, y \in \mathcal{X}$), sont linéaires et conservent la convexité de $\mathcal{A}$. Ainsi, avec une telle formulation, on peut facilement appliquer l'algorithme 2.1 au calcul de l'ensemble de solutions potentiellement WOWA optimales.

Utilisation de fonctions splines

Pour former une base de fonctions génératrice adaptée, il faut trouver un ensemble de fonctions représentatif de l'espace Φ, i.e., le plus large éventail possible de fonctions s_i vérifiant les conditions suivantes :

1. s_i est une fonction continue,

2. s_i est strictement croissante et telle que $s_i(0) = 0$ et $s_i(1) = 1$ (donc positive),

et lorsque l'on souhaite modéliser une aversion au risque ou une exigence d'équité :

3. s_i est convexe.

On propose ici de définir les fonctions s_i comme des fonctions splines. Les fonctions splines sont des fonctions définies par morceaux par des polynômes ; elles sont souvent utilisées en interpolation car elles peuvent représenter différentes formes complexes et lisses Ramsay *et al.* [1988]. Une propriété intéressante de ces fonctions est qu'elles peuvent être obtenues par combinaisons linéaires d'autres fonctions splines. C'est pourquoi on propose de voir φ comme une fonction spline que l'on obtient à partir d'une base formée d'autres fonctions splines. On montre d'abord comment obtenir une base de fonctions vérifiant les conditions 1 et 2 (continuité et croissance de $[0,1]$ vers $[0,1]$) pour représenter un opérateur WOWA général. On utilise pour cela des fonctions I-Splines obtenues à partir d'intégrales de fonctions positives M-Splines (représentation proposée par Perny *et al.* [2016] pour représenter la fonction de déformation du modèle RDU, Rank Dependant Utility). Puis on définira les fonctions C-Splines que l'on obtient par intégration de fonctions I-Splines afin de vérifier la condition 3, i.e. la convexité de φ qui favorise les solutions équilibrées.

Représentation d'une fonction positive - M-Splines

Afin de définir une base génératrice de fonctions positives, on peut utiliser les fonctions M-Splines. Les fonctions M-Splines d'ordre k sont des fonctions positives, polynomiales par morceaux où chaque morceau est d'ordre $k-1$. Elles peuvent être utilisées pour générer des fonctions positives de formes plus ou moins complexes.

On définit une famille (ou base) de fonctions M-Splines d'ordre k à m paramètres libres dans l'intervalle $[a,b]$ à l'aide d'un ensemble de nœuds $t = \{t_1, .., t_{m+k}\}$ tel que :

- $a = t_1 = .. = t_k$,
- $b = t_{m+1} = .. = t_{m+k}$,
- $t_i < t_{i+k}, \forall i \in \{1, \ldots, m\}$.

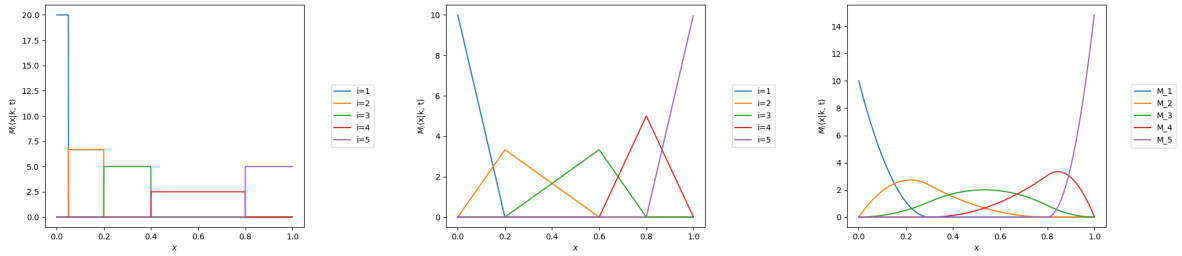
La base est alors constituée de m fonctions $M_1, \ldots, M_m$ d'ordre k, où chaque fonction M_i est définie, pour tout $r \in [a,b]$, par :

$$M_i(r|k,t) = \begin{cases} \frac{1}{t_{i+1}-t_i} & \text{si k = 1 et } t_i \leq r < t_{i+1} \\ \frac{k[(r-t_i)M_i(r|k-1,t)+(t_{i+k}-r)M_{i+1}(r|k-1,t)]}{(k-1)(t_{i+k}-t_i)} & \text{si } k > 1 \\ 0 & \text{sinon} \end{cases}$$

Exemple 2.2. *Cet exemple montre le type de bases de fonctions que l'on obtient lorsque l'on utilise des fonctions M-Splines d'ordre k avec différentes valeurs de k, et pour différents nœuds t donnés. Par exemple, pour générer* 5 *fonctions d'ordre* 2 *dans l'intervalle* $[0,1]$*, on doit définir* $t = \{t_1, \ldots, t_7\}$ *de manière à respecter :*

- $t_1 = t_2 = 0$,
- $t_6 = t_7 = 1$,
- $t_3 \leq t_4 \leq t_5$,
- $t_1 < t_3, t_2 < t_4, t_3 < t_5, t_4 < t_6$ *et* $t_5 < t_7$.

On peut par exemple prendre $t = \{0, 0, 0.2, 0.6, 0.8, 1, 1\}$ *qui respecte toutes les contraintes énoncées. On obtient ainsi la base de fonctions donnée par la figure 2.2(b). Si on souhaite obtenir des courbes plus lisses, on optera alors pour une base de fonctions d'ordre* 3 *comme dans le cas des fonctions de la figure 2.2(c).*



(a) $k = 1$
$t = (0, 0.05, 0.2, 0.4, 0.8, 1)$

(b) $k = 2$
$t = (0, 0, 0.2, 0.6, 0.8, 1, 1)$

(c) $k = 3$
$t = (0, 0, 0, 0.3, 0.8, 1, 1, 1)$

FIGURE 2.2 – Exemples de bases de fonctions M-Splines pour $k = 1, 2$ et 3 sur l'intervalle $[0, 1]$.

On peut utiliser une telle base pour générer différentes formes de fonctions positives. Le type de fonctions obtenu dépendra alors de l'ordre k choisi, on prendra par exemple $k \geq 3$ pour représenter des fonctions continues et lisses. Notons qu'une telle base ne convient pas à la définition d'une base génératrice de fonctions de l'espace Φ car les fonctions M-Splines ne sont pas toujours croissantes et convexes.

Représentation d'une fonction positive et croissante - I-Splines

Lorsque l'on souhaite représenter la fonction de déformation d'un opérateur WOWA général, il est nécessaire de définir une base de fonctions continues, croissantes et telles que $\varphi(0) = 0$ et $\varphi(1) = 1$. Dans ce cas, on peut utiliser une base de fonctions I-Splines définies par des intégrales de fonctions M-Splines. Ces fonctions sont donc, par définition, croissantes et positives.

On définit une base constituée de m fonctions I-Splines d'ordre k dans l'intervalle $[a, b]$ à l'aide d'un ensemble de nœuds $t = \{t_1, .., t_{m+k+2}\}$, vérifiant :

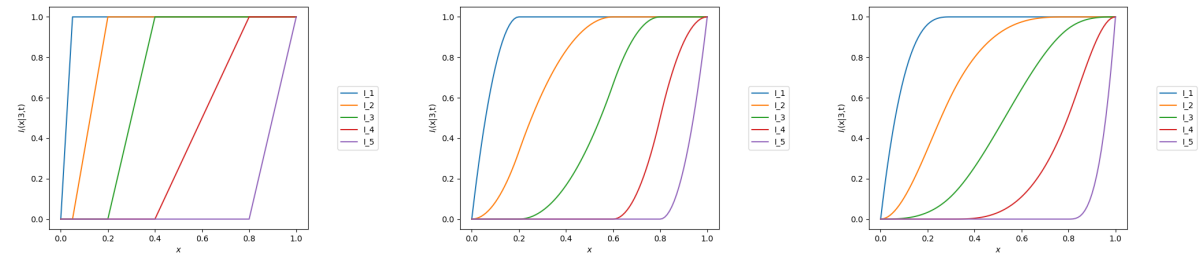
- $a = t_1 = .. = t_{k+1}$,
- $b = t_{m+2} = .. = t_{m+k+2}$,
- $t_i < t_{i+k}, \forall i \in \{1, \ldots, m+1\}$.

La fonction I_i de la base s'écrit alors :

$$I_i(r|k,t) = \int_a^r M_i(u|k,t)\mathrm{d}u = \begin{cases} 0 & \text{si } j < i \\ 1 & \text{si } j > i+k-1 \\ \sum_{z=i}^{j} \frac{t_{z+k+1} - t_z}{k+1} M_z(u|k+1,t) & \text{sinon} \end{cases} \quad (2.3)$$

où j est l'indice de t tel que $t_j \leq r < t_{j+1}$.

Exemple 2.2 (suite). *On montre ici des exemples de bases de fonctions I-Splines obtenues en intégrant les fonctions M-Splines de la base de la figure 2.2.*

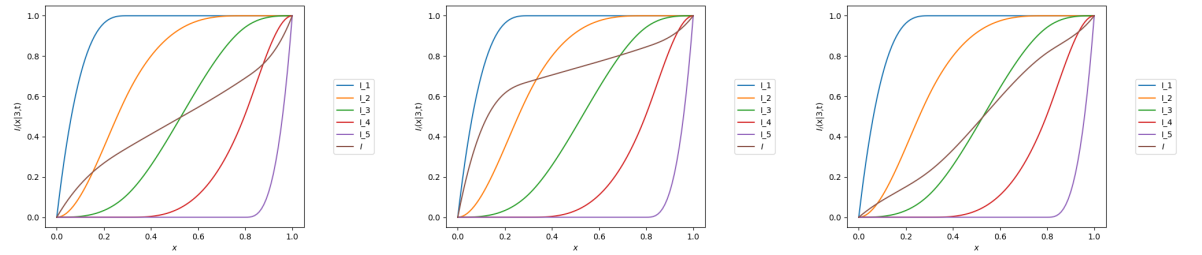


(a) $k = 1$ $t = (0, 0, 0.05, 0.2, 0.4, 0.8, 1, 1)$ (b) $k = 2$ $t = (0, 0, 0, 0.2, 0.6, 0.8, 1, 1, 1)$ (c) $k = 3$ $t = (0, 0, 0, 0, 0.3, 0.8, 1, 1, 1, 1)$

FIGURE 2.3 – Exemples de bases de fonctions I-Splines pour $k = 1, 2$ et 3 sur l'intervalle $[0, 1]$.

Les bases de I-splines permettent alors de définir un large éventail de fonctions continues, croissantes et telles que $\varphi(0) = 0$ et $\varphi(1) = 1$. Par conséquent, en reprenant l'idée proposée par Perny *et al.* [2016] (pour représenter la fonction de déformation du modèle RDU, Rank Dependant Utility), on peut voir la fonction φ de l'opérateur WOWA général comme une combinaison de fonctions I-Splines définies par (2.3), i.e., $\varphi(r) = \sum_{i=1}^{m} \alpha_i I_i(r|k, t)$, où α est un vecteur de pondérations qui permet de contrôler la forme de la fonction obtenue. La figure 2.4 montre différentes fonctions que l'on peut obtenir en faisant varier le paramètre α :

Exemple 2.2 (suite). *La figure 2.4 montre trois exemples de fonctions que l'on peut obtenir en effectuant des combinaisons linéaires des fonctions de la base donnée par la figure 2.3(c). Les trois courbes marron sont obtenues en utilisant les trois différents vecteurs de pondération α donnés :*



(a) $\alpha = (0.2, 0.2, 0.2, 0.2, 0.2)$ (b) $\alpha = (0.6, 0.1, 0.1, 0.1, 0.1)$ (c) $\alpha = (0.1, 0.1, 0.6, 0.1, 0.1)$

FIGURE 2.4 – Exemples de fonctions obtenues par combinaisons linéaires de fonctions I-Splines.

Les fonctions I-Splines peuvent être très utiles pour obtenir une formulation linéaire (en α) de la fonction de déformation d'un WOWA général. Néanmoins, on peut voir sur la figure 2.4 que les fonctions obtenues par combinaisons de fonctions I-Splines ne sont pas toujours convexes, elles ne conviennent donc pas à la définition de φ si on veut modéliser une exigence d'équité ou une aversion au risque. C'est pourquoi nous proposons de définir une nouvelle famille de fonctions splines.

Représentation d'une fonction de déformation par des C-Splines

Afin d'obtenir un opérateur WOWA qui privilégie les solutions équilibrées, nous proposons de reformuler la fonction φ comme une combinaison de fonctions *C-Splines*. Nous définissons ces fonctions comme des intégrales de fonctions *I-Splines*, qui sont alors par définition, des fonctions continues, croissantes et convexes :

Définition 2.7. *Étant donné un ensemble de nœuds* $t = \{t_1, \ldots, t_{m+k+4}\}$ *vérifiant :*
- $a = t_1 = .. = t_{k+2}$,
- $b = t_{m+3} = .. = t_{m+k+4}$,
- $t_i < t_{i+k}, \forall i \in \{1, \ldots, m+2\}$.

on définit, pour tout $r \in [a, b]$*, les fonctions* $C_1, \ldots, C_m$ *de la base de C-Splines d'ordre* k *par :*

$$C_i(r|k,t) = \int_a^r I_i(u|k,t)\ \mathrm{d}u$$

Par définition des fonctions I-Splines, cette intégrale se calcule par morceaux :

$$C_i(r|k,t) = \int_a^{t_j} I_i(u|k,t)\ \mathrm{d}u + \int_{t_j}^r I_i(u|k,t)\ \mathrm{d}u$$

où j est l'indice de t tel que $t_j \leq r < t_{j+1}$. Par définition de $I_i(u|k,t)$ sur l'intervalle $[a, t_j]$ la première partie de cette intégrale s'écrit :

$$\begin{array}{rcl} \int_a^{t_j} I_i(u|k,t)\ \mathrm{d}u & = & \int_a^{t_j} \sum_{z=i}^{j} \frac{t_{z+k+1}-t_z}{k+1} M_z(u|k+1,t)\ \mathrm{d}u \\ & = & \sum_{z=i}^{j} \int_a^{t_j} \frac{t_{z+k+1}-t_z}{k+1} M_z(u|k+1,t)\ \mathrm{d}u \\ & = & \sum_{z=i}^{j} \frac{t_{z+k+1}-t_z}{k+1} \int_a^{t_j} M_z(u|k+1,t)\ \mathrm{d}u \\ & = & \sum_{z=i}^{j} \frac{t_{z+k+1}-t_z}{k+1} I_z(t_j|k+1,t) \end{array}$$

Quant à la seconde partie de l'intégrale, elle consiste en une simple intégration de constante puisque I_i prend la valeur 1 sur l'intervalle $[t_j, r]$. Donc :

$$\int_{t_j}^r I_i(u|k,t)\ \mathrm{d}u = \int_{t_j}^r 1\ \mathrm{d}u = r - t_j$$

On obtient finalement :

$$C_i(r|k,t) = \begin{cases} 0 & \text{si } j < i \\ r - t_j & \text{si } j > i+k-1 \\ \sum_{z=i}^{j} \frac{t_{z+k+1}-t_z}{k+1} I_z(t_j|k+1,t) & \text{sinon} \end{cases} \tag{2.4}$$

Exemple 2.2 (suite). *La figure 2.5 montre un exemple de base de fonctions C-Splines. Cette base est obtenue en intégrant les fonctions I-Splines de la base donnée par la figure 2.3(c) :*

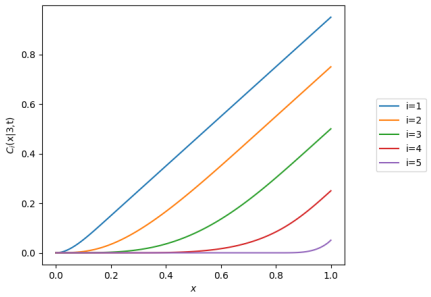


FIGURE 2.5 – Exemple d'une base de fonctions C-Splines pour $k = 3$.

On voit bien que les fonctions de la figure 2.5 sont continues, positives, croissantes et convexes. Néanmoins, la base de fonctions $C_i, i \in \{1, \ldots, m\}$, ne suffit pas encore à la définition de la fonction φ car les conditions $C_i(0|k,t) = 0$ et $C_i(1|k,t) = 1$ ne sont pas vérifiées. Nous définirons alors les fonctions de notre base de la manière suivante : on prend un intervalle $[a, b]$ (définition 2.7) tel que $a = 0$ et $b > 1$ et on pose pour tout $r \in [0, 1]$ et pour tout $i = 1, \ldots, m$:

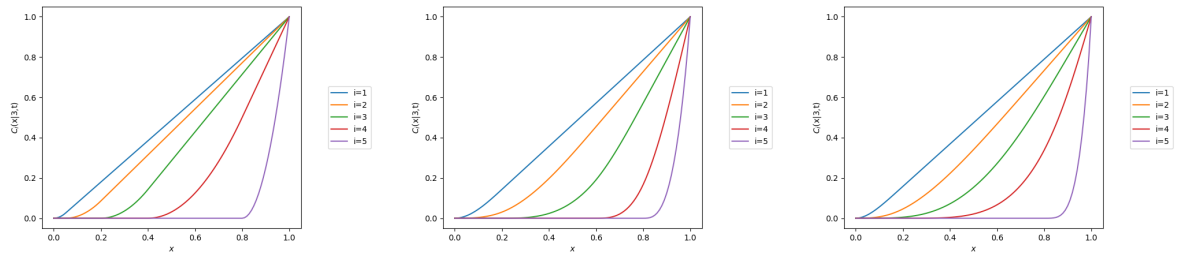
$$s_i(r) = C_i(r|k,t)/C_i(1|k,t) \tag{2.5}$$

pour des valeurs k et t fixées. Notons que pour tout $i \in \{1, \ldots, m\}$, $C_i(1|k,t)$ est une constante pour des valeurs de i, k et t fixées.

Proposition 2.3. *Les fonctions C-Splines normalisées $s_i, i = 1, \ldots, m$, définies par l'équation (2.5) sont continues, croissantes, convexes et telles que $s_i(0) = 0$ et $s_i(1) = 1$.*

Démonstration. Cette proposition est vraie par définition des fonctions s_i. En effet, pour tout $i \in \{1, \ldots, m\}$, s_i est *continue* car elle est dérivable en tout point $r \in [0, 1]$ puisqu'elle se définit par une intégrale de I-Spline. s_i est *croissante* (resp. *convexe*), car sa fonction dérivée $I_i(r|k,t)/C_i(1|k,t)$, où $C_i(1|k,t)$ est une constante, est positive (resp. croissante). De plus, les conditions $s_i(0) = 0$ et $s_i(1) = 1$ sont vérifiées de manière évidente. □

Exemple 2.2 (suite). *La figure 2.6 montre les fonctions C-Splines normalisées obtenues en intégrant les fonction I-Splines de la figure 2.3 :*



(a) $k = 1$ $t = (0, 0, 0, 0.05, 0.2, 0.4, 0.8, 1, 1, 1)$

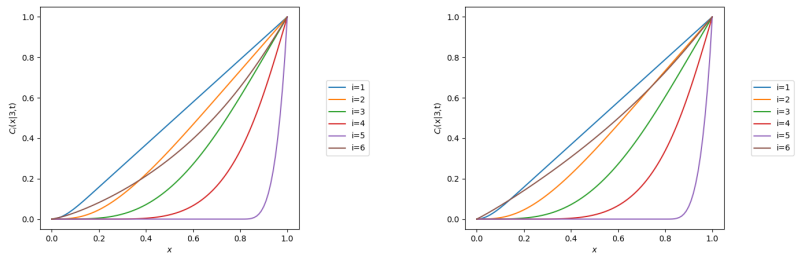
(b) $k = 2$ $t = (0, 0, 0, 0, 0.2, 0.6, 0.8, 1, 1, 1, 1)$

(c) $k = 3$ $t = (0, 0, 0, 0, 0, 0.3, 0.8, 1, 1, 1, 1, 1)$

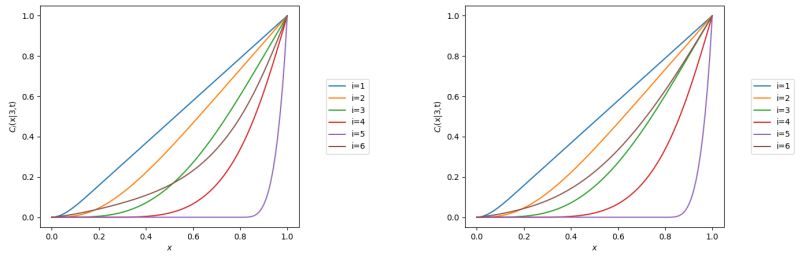
FIGURE 2.6 – Exemples de bases de fonctions C-Splines pour $k = 1, 2$ et 3 sur l'intervalle $[0, 1]$.

On peut voir que les fonctions C-Splines d'ordre 3 *offrent plus de souplesse que les fonctions d'ordre inférieur. En effet, les courbes des fonctions d'ordre* 3 *de cet exemple sont plus arrondies et semblent pouvoir permettre de générer un plus vaste panel de fonctions que les courbes obtenues pour* $k = 1$ *ou* $k = 2$*. Ces dernières fonctions sont en effet quasi linéaires ou nulles sur une bonne partie de l'intervalle* $[0, 1]$*.*

La Figure 2.7 montre différentes fonctions obtenues par combinaisons des fonctions de la figure 2.6.(c). On donne en marron la courbe $s(r) = \sum_{i=1}^{m} \alpha_i s_i(r)$ *selon différentes valeurs pour* α*.*



(a) $\alpha = (0.2, 0.2, 0.2, 0.2, 0.2)$ (b) $\alpha = (0.6, 0.1, 0.1, 0.1, 0.1)$



(c) $\alpha = (0.1, 0.1, 0.1, 0.1, 0.6)$ (d) $\alpha = (0.1, 0.1, 0.1, 0.6, 0.1)$

FIGURE 2.7 – Exemples de fonctions obtenues par combinaisons linéaires de fonctions C-Splines.

La fonction φ sera alors définie comme une combinaison convexe des fonctions d'une base de C-Splines normalisées, i.e., $\varphi(r) = \sum_{i=1}^{m} \alpha_i s_i(r)$ où chaque s_i est définie par l'équation (2.5). Le paramètre α nous permet alors de définir une infinité de fonctions vérifiant les conditions de la définition d'une fonction de déformation (proposition 2.3). En partant d'une telle base génératrice, on peut reformuler l'opérateur WOWA de la manière suivante :

Définition 2.8. *Soient* $s_i, i \in \{1, \ldots, m\}$*, les* m *fonctions d'une base de fonctions C-Splines normalisées. Soit* $\alpha \in \mathcal{A}$ *un vecteur de pondération et soit* x *une solution de* $\mathcal{X}$*, on écrit :*

$$\text{WOWA}_\alpha(x) = \sum_{i=1}^{n} [x_{(i)} - x_{(i-1)}] \sum_{j=1}^{m} \alpha_j s_j \left(\sum_{k=i}^{n} p_{(k)}\right) \tag{2.6}$$

$$= \sum_{j=1}^{m} \alpha_j \sum_{i=1}^{n} [x_{(i)} - x_{(i-1)}] s_j \left(\sum_{k=i}^{n} p_{(k)}\right) \tag{2.7}$$

On peut voir grâce à la formulation de l'équation (2.7) que l'expression de WOWA est maintenant linéaire en son nouveau paramètre α. Cette nouvelle formulation mène aux

définitions de l'ensemble de solutions potentiellement optimales, de $\mathcal{A}-$dominance et de $\mathcal{A}-$dominance d'ensembles suivantes :

Définition 2.9.

$$PO_{\mathcal{A}}(\mathcal{X}) = \bigcup_{\alpha \in \mathcal{A}} \arg\max_{x \in \mathcal{X}} \mathrm{WOWA}_{\alpha}(x)$$

Définition 2.10. *Soient x et y deux solutions dans $\mathcal{X}$, on a :*

$$x \succ_{\mathcal{A}} y \iff \forall \alpha \in \mathcal{A}, \mathrm{WOWA}_{\alpha}(x) > \mathrm{WOWA}_{\alpha}(y)$$

on dit alors que la solution x $\mathcal{A}-$domine la solution y au sens de la fonction WOWA_{α}.

Définition 2.11. *Soient $Y \subseteq \mathcal{X}$ et $Z \subseteq \mathcal{X}$ deux sous-ensembles de solutions, on a :*

$$Y \succ_{\mathcal{A}} Z \iff \forall z \in Z, \forall \alpha \in \mathcal{A}, \exists y \in Y, \mathrm{WOWA}_{\alpha}(y) > \mathrm{WOWA}_{\alpha}(z)$$

on dit alors que l'ensemble Y $\mathcal{A}-$domine l'ensemble Z au sens de la fonction WOWA_{α}.

Puisque l'on dispose d'une expression de WOWA qui est linéaire en son paramètre, on peut à présent utiliser l'algorithme 2.1 pour la détermination des solutions potentiellement WOWA-optimales. Le programme permettant de vérifier la relation $\mathcal{X} \succ_{\mathcal{A}} \{x\}$ pour toute solution $x \in \mathcal{X}$:

$$(P_{\,\mathrm{WOWA}_{\alpha},x}) : \begin{cases} \min t \\ t \geq \mathrm{WOWA}_{\alpha}(y) - \mathrm{WOWA}_{\alpha}(x) \quad \forall y \in \mathcal{X} \\ \alpha \in \mathcal{A} \end{cases}$$

constitue bien un programme linéaire.

2.2.3 Élicitation partielle des préférences

L'ensemble de solutions potentiellement optimales PO donne un ensemble de solutions pouvant intéresser le décideur au vu des informations acquises. Cependant, cet ensemble peut être de très grande taille, et l'acquisition de nouvelles informations peut montrer que certaines de ces solutions intéressent beaucoup moins le décideur que d'autres. On va alors tenter de réduire l'incertitude concernant les préférences du décideur afin d'écarter de telles solutions de l'ensemble PO. En d'autres termes, nous allons éliciter les préférences du décideur afin de réduire l'ensemble des paramètres possibles W ou $\mathcal{A}$, ce qui pourra nous permettre de réduire l'ensemble $PO_W(\mathcal{X})$ ou $PO_{\mathcal{A}}(\mathcal{X})$ (par définition de PO). L'objectif est de réduire suffisamment l'ensemble d'incertitude (sans pour autant le réduire à une seule valeur) pour pouvoir déterminer une solution *nécessairement-optimale* i.e. une solution optimale pour tout $w \in W$ ou pour tout $\alpha \in \mathcal{A}$:

Définition 2.12. *On note NO l'ensemble des solutions nécessairement optimales. Étant donnée une fonction d'agrégation f_{ω} de paramètre $\omega \in \mathcal{W}$ et une solution $x \in \mathcal{X}$:*

$$x \in NO_{\mathcal{W}}(\mathcal{X}) \iff \forall \omega \in \mathcal{W}, x \in \arg\max_{y \in \mathcal{X}} f_{\omega}(y)$$

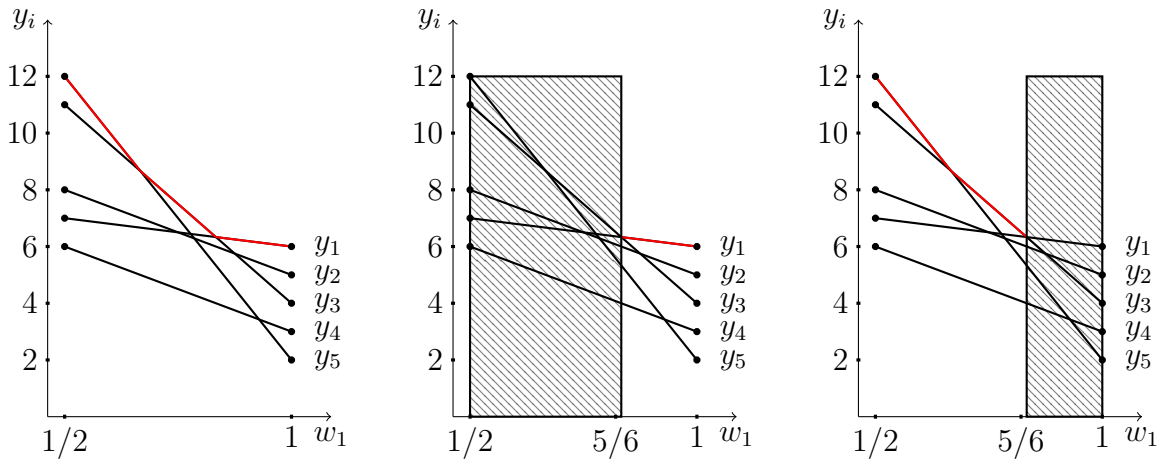
où f_{ω} peut représenter indifféremment OWA_w *ou* WOWA_{α}*, et où $\mathcal{W}$ est défini par W ou par $\mathcal{A}$.*

Pour pouvoir aboutir à une solution ou un ensemble de solutions NO, nous allons collecter auprès du décideur des informations du type *"je préfère la solution x à la solution y"*. La relation de préférences $x \succsim y$ se traduit par la contrainte $f_\omega(x) \geq f_\omega(y)$ permettant de réduire l'ensemble d'incertitude et, par conséquent, de mieux discriminer entre les solutions de l'ensemble PO. Comme nous avons pu le préciser précédemment, les contraintes $f_\omega(x) \geq f_\omega(y), x, y \in \mathcal{X}$, sont linéaires ($\text{OWA}_w$ est linéaire en w et WOWA_α est linéaire en α). De ce fait, l'ajout de telles contraintes conserve la convexité des polyèdres associés à W et $\mathcal{A}$.

Exemple 2.1 (suite) *Reprenons l'exemple 2.1 où l'ensemble des solutions est donné par* $\mathcal{X} = \{x_1, \ldots, x_5\}$ *dont l'image dans l'espace des critères est* $\mathcal{Y} = \{(6,8), (11,5), (4,18), (5,3), (22,2)\}$. *La figure 2.8(a) donne l'évaluation de chaque solution selon* OWA *en fonction de la valeur de la première composante du paramètre w (avec $w_2 = 1 - w_1$). Supposons que l'on demande au décideur de comparer les deux solutions x_1 et x_3 :*

Cas 1. *Le décideur préfère x_1, on ajoute alors la contrainte* $\text{OWA}_w(x_1) \geq \text{OWA}_w(x_3)$ *à la définition de l'ensemble des paramètres possibles. Autrement dit, on restreint W par la contrainte* $6w_1 + 8w_2 \geq 4w_1 + 18w_2$, *ou encore (en remplaçant w_2 par $1 - w_1$)* $w_1 \geq 5/6$. *On obtient alors le nouvel espace* $W = \{w \in \mathbb{R}^2_+ | w_1 + w_2 = 1, w_1 \geq 5/6\}$ *illustré par la figure 2.8(b) où la zone hachurée correspond au sous-ensemble de W que l'on supprime. Ici, l'ensemble des solutions PO est réduit à l'unique solution x_1 qui est donc nécessairement optimale.*

Cas 2. *Le décideur préfère la solution x_3, on ajoute alors la contrainte correspondante :* $\text{OWA}_w(x_1) \leq \text{OWA}_w(x_3)$ *à la définition de W, ce qui mène cette fois à la contrainte* $w_1 \leq 5/6$. *Ce cas est illustré par la Figure 2.8(c) où la zone hachurée correspond toujours au sous-ensemble de W que l'on supprime. L'ensemble des solutions PO est réduit aux deux solutions x_3 et x_5 et l'ensemble de solutions nécessairement optimales est vide. De nouvelles informations sont alors nécessaires à la détermination d'une solution NO.*



(a) $w_1 \in [1/2, 1]$ (b) $w_1 \in [5/6, 1]$ (c) $w_1 \in [0, 5/6]$

FIGURE 2.8 – Image des solutions de $\mathcal{X}$ en fonction de l'ensemble des paramètres W.

On voit à travers cet exemple, et plus particulièrement à travers le *cas 1* (figure 2.8(b)),

qu'une élicitation partielle des préférences du décideur suffit à la détermination d'une solution NO. En effet, dans ce cas nous avons identifié une solution NO sans avoir une connaissance précise de la valeur des paramètres modélisant les préférences du décideur. Les procédures d'élicitation des préférences que nous développons ici sont fondées sur cette observation. Nous alternerons calcul de solutions potentiellement optimales et questions préférentielles jusqu'à aboutir à une solution nécessairement optimale, sans chercher à connaitre les préférences exactes du décideur.

Afin de déterminer une procédure d'élicitation efficace, i.e., qui permet de déterminer une solution NO en un minimum de questions, il est nécessaire de disposer d'une bonne stratégie de choix de la question à poser à chaque itération. Dans ce chapitre on se basera essentiellement sur la stratégie *CSS* (Current Solution Strategy) introduite dans [Wang et Boutilier, 2003; Boutilier *et al.*, 2006a] dans le cadre de la prise de décision avec des valeurs de performances imprécisément connues.

Élicitation par la stratégie CSS

Cette stratégie utilise une notion de regrets permettant d'évaluer les solutions réalisables de $\mathcal{X}$ par la perte de performance qu'elles engendrent. Plus formellement, on définit les trois notions suivantes :

Définition 2.13. *Soit f_ω une fonction d'agrégation de paramètre $\omega \in \mathcal{W}$ et soient deux solutions x et y dans $\mathcal{X}$. On définit le regret maximum de la paire (Pairwise Maximum Regret) x, y par :*

$$\mathrm{PMR}(x, y, \mathcal{W}) = \max_{\omega \in \mathcal{W}} \{f_\omega(y) - f_\omega(x)\}$$

En d'autres termes, le PMR donne la valeur du pire regret engendré par le choix de la solution x au lieu de la solution y. Lorsque la fonction f_ω est linéaire en ω et lorsque $\mathcal{W}$ est un ensemble représenté par un polyèdre convexe (comme c'est le cas pour OWA_w et WOWA_α), le PMR se calcule par simple programmation linéaire.

Définition 2.14. *Soit f_ω une fonction d'agrégation de paramètre $\omega \in \mathcal{W}$ et soit x une solution de $\mathcal{X}$. On définit le regret maximum (Maximum Regret) de x par :*

$$\mathrm{MR}(x, \mathcal{X}, \mathcal{W}) = \max_{y \in \mathcal{X}} \mathrm{PMR}(x, y, \mathcal{W})$$

En d'autres termes, le MR donne la valeur du pire regret engendré par le choix de la solution x au lieu de n'importe quelle autre solution de $\mathcal{X}$.

Définition 2.15. *Soit f_ω une fonction d'agrégation de paramètre $\omega \in \mathcal{W}$. On définit le plus petit regret maximum (MiniMax Regret) de $\mathcal{X}$ par :*

$$\mathrm{MMR}(\mathcal{X}, \mathcal{W}) = \min_{x \in \mathcal{X}} \mathrm{MR}(x, \mathcal{X}, \mathcal{W})$$

Autrement dit, la valeur MMR nous donne la meilleure valeur du regret pire cas dans tout $\mathcal{X}$.

Les regrets donnés ci-dessus permettent de définir à la fois une stratégie de choix de la question à poser et un critère de décision. À n'importe quelle étape de l'algorithme, une bonne recommandation au sens de ce critère consiste à choisir une solution assurant

le plus petit regret maximum possible, i.e., on recommandera au décideur une solution $x^* \in \arg\min_{x\in\mathcal{X}} \mathrm{MR}(x, \mathcal{X}, \mathcal{W})$ [Savage, 1954]. Ainsi, on lui garantit que son regret pire cas est minimal. Ce regret pire cas constitue également une borne du regret réel :

Proposition 2.4. *Soit f_ω une fonction d'agrégation, soient $\hat{\omega}$ le paramètre (caché) représentant les préférences du décideur et $\hat{x}$ la solution $f_{\hat{w}}$-optimale associée, et soit $x^* \in \arg\min_{x\in\mathcal{X}} \mathrm{MR}(x, \mathcal{X}, \mathcal{W})$. On a alors :*

$$f_{\hat{\omega}}(\hat{x}) - f_{\hat{\omega}}(x^*) \leq \mathrm{MMR}(\mathcal{X}, \mathcal{W})$$

Démonstration. Cette proposition est vraie par définition du regret maximum de la solution x^* :

$$\begin{array}{rcll} \mathrm{MMR}(\mathcal{X}, \mathcal{W}) & = & \max_{y\in\mathcal{X}} \mathrm{MR}(x^*, \mathcal{X}, \mathcal{W}) & \\ & \geq & \mathrm{PMR}(x^*, y, \mathcal{W}) & \forall y \in \mathcal{X} \\ & \geq & f_\omega(y) - f_\omega(x^*) & \forall \omega \in \mathcal{W}, \forall y \in \mathcal{X} \end{array}$$

L'inégalité étant vraie pour toute solution y et pour tout paramètre ω, elle est vraie en particulier pour $y = \hat{x}$ et $\omega = \hat{\omega}$. □

Notons que le regret maximum de toute solution $x \in \mathcal{X}$ peut borner la valeur du regret réel. Le MMR donne alors la meilleure borne possible de ce regret pour l'ensemble d'incertitude courant $\mathcal{W}$.

Lorsque l'on souhaite approfondir nos connaissances sur les préférences du décideur, le choix de la question à poser dans la stratégie CSS consiste à demander au décideur de comparer deux alternatives intéressantes au sens des regrets pire cas au vu des informations dont on dispose. On lui proposera alors de comparer la solution assurant le plus petit regret maximum $x^* \in \arg\min_{x\in\mathcal{X}} \mathrm{MR}(x, \mathcal{X}, \mathcal{W})$ et la solution qui représente son pire adversaire au sens du regret maximum, i.e. $y^* \in \arg\max_{y\in\mathcal{X}} \mathrm{PMR}(x^*, y, \mathcal{W})$. On alternera alors calculs de regrets, questions de comparaisons de solutions et mise à jour de l'espace des paramètres jusqu'à annuler la valeur MMR ou jusqu'à vérifier $\mathrm{MMR}(\mathcal{X}, \mathcal{W}) \leq \delta$, où δ est un seuil positif. Cette stratégie offre plusieurs avantages qui motivent son utilisation dans ce travail :

1. Elle pose des questions qui mènent toujours à la réduction de l'espace des paramètres. En effet, les questions posées concernent toujours des solutions que l'on ne sait pas comparer au vu de la connaissance courante des préférences du décideur :

Proposition 2.5. *Soient $x^* \in \arg\min_{x\in\mathcal{X}} \mathrm{MR}(x, \mathcal{X}, \mathcal{W})$ et $y^* \in \arg\max_{y\in\mathcal{X}} \mathrm{PMR}(x^*, y, \mathcal{W})$ les deux solutions à comparer selon la stratégie CSS. On a :*

$$y^* \in PO_{\mathcal{W}}(\mathcal{X}) \text{ et } \exists \omega \in \mathcal{W} \text{ tel que } f_\omega(x^*) \geq f_\omega(y^*)$$

Démonstration. Démontrons d'abord que $y^* \in PO_{\mathcal{W}}(\mathcal{X})$: supposons que y^* n'est pas potentiellement optimale. Dans ce cas, pour tout paramètre $\omega \in \mathcal{W}$, il existe une solution $z_\omega \in \mathcal{X}$ telle que $f_\omega(z_\omega) > f_\omega(y^*)$. On a alors $f_\omega(z_\omega) - f_\omega(x^*) > f_\omega(y^*) - f_\omega(x^*), \forall \omega \in \mathcal{W}$, ce qui contredit le fait que $y^* \in \arg\max_{y\in\mathcal{X}} \mathrm{PMR}(x^*, y, \mathcal{W})$. Montrons maintenant par l'absurde que la solution x^* est meilleure que la solution y^* pour au moins une valeur de paramètres possibles. Supposons qu'il n'existe aucun poids $\omega \in \mathcal{W}$ tel que $f_\omega(x^*) \geq f_\omega(y^*)$, i.e., $\forall \omega \in \mathcal{W}, f_\omega(x^*) < f_\omega(y^*)$. Donc par définition du PMR,

pour toute solution $z \in \mathcal{X}, \text{PMR}(x^*, z, W) > \text{PMR}(y^*, z, W)$. Par définition du MR on en déduit que $\text{MR}(x^*, \mathcal{X}, W) > \text{MR}(y^*, \mathcal{X}, W)$, ce qui contredit le fait que $x^* \in \arg\min_{x \in X} \text{MR}(x, \mathcal{X}, \mathcal{W})$. Donc, il existe au moins un jeu de poids pour lequel la solution x^* est meilleure que la solution y^*. □

Cette proposition nous montre que la réponse à une question de cette stratégie mène forcément à la réduction de l'espace des paramètres puisque chacune des deux solutions possède au moins un jeu de poids la rendant f_ω-optimale.

2. La stratégie CSS pose des questions qui ne peuvent pas réduire $\mathcal{W}$ à l'ensemble vide : lorsqu'une question de la forme "$x \succsim_? y$" est choisie par cette stratégie, les deux réponses sont possibles (conséquence directe de la proposition 2.5).

3. En plus de réduire l'espace des paramètres, les questions peuvent également réduire la valeur du MMR ce qui permet de raffiner la borne du regret réel donnée dans la proposition 2.4. En effet, la réduction de l'espace des paramètres peut réduire le regret maximum de chaque solution $x \in \mathcal{X}$ mais ne peut jamais l'augmenter (par définition du MR), ce qui peut mener à la réduction de la valeur MMR. On a de plus :

Proposition 2.6. *Notons x^* une solution dans* $\arg\min_{x \in \mathcal{X}} \text{MR}(x, \mathcal{X}, \mathcal{W})$ *:*

$$\text{MMR}(\mathcal{X}, \mathcal{W}) = 0 \Leftrightarrow x^* \in NO_{\mathcal{W}}(\mathcal{X})$$

et pour tout seuil $\delta > 0$, en notant $\hat{\omega}$ le paramètre (caché) représentant les préférences du décideur et $\hat{x}$ la solution $f_{\hat{\omega}}$-optimale associée :

$$\text{MMR}(\mathcal{X}, \mathcal{W}) \leq \delta \Rightarrow f_{\hat{\omega}}(\hat{x}) - f_{\hat{\omega}}(x^*) \leq \delta$$

Démonstration. La relation d'équivalence est vraie par définition du MMR et de l'ensemble $NO_{\mathcal{W}}(\mathcal{X})$. En effet, supposons que la solution x^* est telle que $\text{MR}(x^*, X, \mathcal{W}) = 0$ alors $\max_{y \in \mathcal{X}} \max_{\omega \in \mathcal{W}} f_\omega(y) - f_\omega(x^*) = 0$, donc pour toute solution $y \in \mathcal{X}$, on a $\max_{w \in W} f_\omega(x^*) - f_\omega(y) = 0$. On peut en déduire que pour toute solution $y \in \mathcal{X}$ et pour tout paramètre $\omega \in \mathcal{W}$ on a $f_\omega(x^*) \geq f_\omega(y)$, ce qui correspond à la définition d'une solution nécessairement optimale. Donc, on a bien $\text{MMR}(\mathcal{X}, \mathcal{W}) = 0 \Rightarrow x^* \in NO_{\mathcal{W}}(\mathcal{X})$. En partant de $x^* \in NO_{\mathcal{W}}(\mathcal{X})$ et en appliquant le raisonnement inverse, on trouve l'implication $x^* \in NO_{\mathcal{W}}(\mathcal{X}) \Rightarrow \text{MMR}(\mathcal{X}, \mathcal{W}) = 0$.

Quant à la seconde partie de la proposition, l'implication découle directement de la proposition 2.4. □

C'est ce dernier point qui justifie l'utilisation de la valeur du MMR comme critère d'arrêt de la procédure d'élicitation de cette stratégie ($\text{MMR}(\mathcal{X}, \mathcal{W}) = 0$ ou $\text{MMR}(\mathcal{X}, \mathcal{W}) \leq \delta$ pour une résolution approchée). Notons qu'une telle procédure se termine même si elle peut, en théorie, nécessiter un nombre exponentiel de questions. Puisque dans le pire cas toutes les paires de solutions sont comparées, et une solution NO est déterminée, ce qui mène à une valeur de MMR nulle. Nous verrons qu'en pratique, cette stratégie s'avère être très efficace (voir également [Boutilier *et al.*, 2006a; Braziunas et Boutilier, 2007] par exemple). Nous verrons dans la sous-section suivante quelques difficultés rencontrées lors du calcul des regrets sur domaine combinatoire.

2.2.4 Extension de l'approche sur domaine combinatoire

Nous avons présenté dans cette section une méthode de calcul de l'ensemble $PO_{\mathcal{W}}(\mathcal{X})$ lorsque l'ensemble de solutions $\mathcal{X}$ est défini explicitement, ainsi qu'une méthode d'élicitation permettant d'aboutir à une solution nécessairement optimale. Lorsque l'ensemble des solutions réalisables $\mathcal{X}$ est défini de manière implicite, ces deux méthodes deviennent impraticables : tout d'abord, le nombre de solutions potentiellement optimales n'est pas borné et peut être exponentiel. Il est alors, dans ce cas, impossible de calculer l'ensemble PO en un temps raisonnable. De plus, la détermination de cet ensemble par un algorithme de filtrage tel que l'algorithme 2.1 devient inenvisageable car elle nécessite l'énumération de toutes les solutions de l'ensemble $\mathcal{X}$ (ligne 1 de l'algorithme) afin d'identifier les solutions dominées à écarter. Cette identification nécessite également un calcul ne pouvant être envisagé dans ce contexte car, pour toute solution $x \in \mathcal{X}$, la vérification de la condition $\mathcal{X} \succsim_{\mathcal{W}} \{x\}$ nécessite la résolution du programme $(P_{\mathrm{OWA}_w,x})$ ou du programme $(P_{\mathrm{WOWA}_\alpha,x})$ qui contient alors un nombre exponentiel de contraintes de la forme $t \geq \max_{y\in\mathcal{X}} f_\omega(y) - f_\omega(x), \forall y \in \mathcal{X}$.

Il est donc nécessaire de définir une méthode adaptée à la recherche de solutions potentiellement OWA et WOWA optimales sur domaine combinatoire. Notons que l'idée de l'algorithme 2.1 peut tout de même être exploitée comme nous le verrons par la suite.

Tout comme le calcul des solutions PO, la stratégie d'élicitation doit également être adaptée lorsque l'on traite un problème défini sur domaine combinatoire. La stratégie CSS (comme d'autres stratégies, voir par exemple [Teso *et al.*, 2016]) nécessite une énumération complète des solutions de $\mathcal{X}$ pour la détermination des valeurs MR et MMR. En effet, le regret maximum et le minimax regret, donnés par les définitions 2.14 et 2.15 s'écrivent :

$$\begin{aligned} \mathrm{MR}(x,\mathcal{X},W) &= \max_{y\in\mathcal{X}} \max_{\omega\in\mathcal{A}} f_\omega(y) - f_\omega(x) \\ \mathrm{MMR}(\mathcal{X},W) &= \min_{x\in\mathcal{X}} \max_{y\in\mathcal{X}} \max_{\omega\in\mathcal{A}} f_\omega(y) - f_\omega(x) \end{aligned}$$

Le calcul de ces regrets nécessite alors un nombre exponentiel de comparaisons par paires. De plus, ces formules sont quadratiques puisque les solutions x et y ainsi que les poids ω sont des variables du problème. De ce fait, la programmation linéaire n'est pas directement possible, d'autant plus que les opérateurs que l'on utilise ne sont pas linéaires.

Une première manière de surmonter ces difficultés est développée dans la suite de ce chapitre.

2.3 Combiner élication et optimisation sur domaine combinatoire avec les modèles OWA et WOWA

Comme nous avons pu le voir dans la section précédente, la notion de potentielle optimalité permet de définir un sous-ensemble de solutions intéressantes lorsque les préférences du décideur ne sont pas précisément connues. Néanmoins, la détermination des solutions potentiellement optimales semble être laborieuse lorsque l'on s'intéresse à la résolution d'un problème défini sur domaine combinatoire. Nous présentons ici une première famille d'algorithmes permettant de pallier cette difficulté. Ces algorithmes se fondent sur une exploration partielle et méthodique de l'espace des solutions et permettent de réduire la charge de calcul en considérant uniquement des sous-ensembles particuliers de

solutions. Outre le calcul des solutions potentiellement optimales, on s'intéressera aussi à l'adaptation de la stratégie CSS afin d'éliciter les préférences du décideur de manière efficace.

2.3.1 Résolution par énumération partielle des solutions du problème

L'idée de base des algorithmes introduits dans cette section est d'énumérer progressivement des solutions intéressantes et de travailler, par étapes, sur des sous-ensembles de solutions de taille raisonnable. De cette manière, on se ramène au cas du calcul de solutions potentiellement et nécessairement optimales sur domaine explicite. Il faut alors définir les trois éléments principaux suivants :

1. *une méthode d'énumération partielle efficace*, on doit pouvoir énumérer simplement et rapidement des solutions de bonne qualité au sens de l'opérateur choisi malgré une connaissance partielle de la valeur de ses paramètres. La procédure d'énumération devra alors être indépendante de cet opérateur tout en fournissant des solutions intéressantes au sens de ce dernier,
2. *un critère d'arrêt efficace*, l'algorithme doit nous permettre de détecter le plus rapidement possible la présence d'une solution optimale dans l'ensemble des solutions énumérées,
3. *une procédure d'élicitation incrémentale efficace*, elle sera alternée avec l'énumération de solutions et devra réduire le plus possible l'incertitude sur les préférences du décideur afin d'écarter les solutions énumérées les moins intéressantes pour le décideur.

Un tel algorithme aura la structure générale décrite par l'algorithme 2.2 :

Algorithme 2.2 : Énumération & élicitation($\mathcal{X}$, $\mathcal{W}$)

Entrées : $\mathcal{X}$: ensemble de solutions défini implicitement ; $\mathcal{W}$: ensemble d'incertitude ; K : taille des sous-ensembles de solutions

```
PO ← ∅ ;
k ← 1 ;
tant que Critère d'arrêt non vérifié faire
    x ← énumérer(X, k) ;
    si non(PO ≻_W {x}) alors
        PO ← PO ∪ {x} ;
        si |PO| = K alors
            W ← Élicitation(PO, W) ;
            Mettre à jour PO ;
        fin
    fin
    k ← k + 1 ;
fin
W ← Élicitation(PO, W) ;
Mettre à jour PO ;
retourner PO ;
```

Dans les méthodes que nous proposons dans cette section, la fonction *énumérer*($\mathcal{X}, k$) (ligne 4) et le critère d'arrêt sont très liés. Ces méthodes se basent sur la détermination d'une fonction linéaire f_{lin} proche de la fonction d'agrégation f_ω (OWA$_w$ ou WOWA$_\alpha$). La proximité de ces deux fonctions signifie que les solutions performantes au sens de f_{lin} seront également (en général) relativement performantes au sens de f_ω. L'intérêt de la définition d'une telle fonction est que, étant linéaire, f_{lin} respecte le principe d'optimalité de Bellman et offre la possibilité de définir des procédures d'énumération simples comme par exemple des méthodes constructives ou des méthodes fondées sur des échanges de sous-solutions. D'autre part, f_{lin} sera utilisée pour définir un critère d'arrêt de la procédure d'énumération. Par conséquent, plus f_{lin} est proche de f_ω plus le critère d'arrêt sera efficace. De ce fait, la proximité des fonctions f_{lin} et f_ω est très importante car elle constitue l'efficacité de l'algorithme.

Notons que la vérification de la relation $PO \succ_{\mathcal{W}} \{x\}$ (ligne 5 de l'algorithme) constitue un tour de boucle de l'algorithme de filtrage présenté dans l'algorithme 2.1. Quant aux mises à jour de l'ensemble PO (lignes 9 et 15), elles peuvent se faire en appliquant l'algorithme 2.1 à l'ensemble PO courant, i.e., par l'appel de la fonction *Filtrer(*$PO, \mathcal{W}$*)*, mais peuvent être faites plus efficacement en utilisant les valeurs de regrets comme on le verra dans la suite. Ainsi, les différents filtrages de solutions PO ainsi que les phases d'élicitation sont faits sur des sous-ensembles de solutions de taille K, où K est un paramètre préalablement fixé.

Avant de présenter les différentes méthodes d'énumération, on présente ici la phase d'élicitation des préférences plus en détail. Afin de préciser nos connaissances sur les préférences du décideur, on alternera énumération de solutions et phases d'élicitation des préférences en utilisant la stratégie CSS sur l'ensemble de solutions PO courant. À chaque fois que cet ensemble atteint une taille $|PO| = K$, on effectue une phase d'élicitation permettant de restreindre l'espace des paramètres $\mathcal{W}$ et de mieux discriminer entre les solutions de PO. L'algorithme 2.3 détaille le déroulement de ces phases :

Algorithme 2.3 : Élicitation($\mathcal{X}_{exp}$, $\mathcal{W}$)

Entrées : $\mathcal{X}_{exp}$: ensemble de solutions défini explicitement ; $\mathcal{W}$: ensemble d'incertitude.

1 **répéter**
2 | Calculer les regrets $\text{PMR}(x, y, \mathcal{W})$ pour toute paire de solutions $(x, y) \in \mathcal{X}_{exp}^2$;
3 | Calculer le regret maximum $\text{MR}(x, \mathcal{X}_{exp}, \mathcal{W})$ de chaque solution $x \in \mathcal{X}_{exp}$;
4 | Déterminer $x^* \in \arg\min_{x \in \mathcal{X}_{exp}} \text{MR}(x, \mathcal{X}_{exp}, \mathcal{W})$;
5 | Déterminer $y^* \in \arg\max_{y \in \mathcal{X}_{exp}} \text{PMR}(x^*, y, \mathcal{W})$;
6 | Demander au décideur s'il préfère x^* ou y^* ;
7 | **si** $x^* \succsim y^*$ **alors**
8 | | $\mathcal{W} \leftarrow \{\omega \in \mathcal{W} | f_\omega(x^*) \geq f_\omega(y^*)\}$
9 | **sinon**
10 | | $\mathcal{W} \leftarrow \{\omega \in \mathcal{W} | f_\omega(x^*) \leq f_\omega(y^*)\}$
11 | **fin**
12 **jusqu'à** $\text{MMR}(\mathcal{X}_{exp}, \mathcal{W}) = 0$;
13 **retourner** $\mathcal{W}$;

À chaque phase d'élicitation, la détermination des questions à poser nécessite alors le

calcul de $K \times (K-1)$ valeurs de PMR au lieu du nombre exponentiel nécessaire à l'application de la stratégie CSS à $\mathcal{X}$. Lorsque le $\text{MMR}(\mathcal{X}_{exp}, \mathcal{W})$ vaut 0, la phase d'élicitation s'arrête car nous pouvons déterminer un ensemble de solutions nécessairement optimales dans $\mathcal{X}_{exp}$ (proposition 2.6). De ce fait, toute solution x vérifiant $\text{MR}(x, \mathcal{X}_{exp}, \mathcal{W}) = 0$ domine toute solution ayant un regret maximum strictement positif. Les valeurs de regrets maximum que nous avons calculées peuvent alors également servir à filtrer PO : on ne gardera dans l'ensemble PO courant que les solutions ayant un regret maximum nul.

2.3.2 Algorithmes de ranking pour la recherche d'une solution OWA-optimale

Les premiers algorithmes présentés ici reposent sur une énumération ordonnée (ou ranking) des solutions de $\mathcal{X}$ selon la fonction moyenne. Cette approche s'inspire de la méthode de ranking présentée par Galand [2008] dans le cadre de l'optimisation d'une fonction d'agrégation f_ω dont le paramètre ω est connu et fixé a priori. L'approche consiste à passer à un espace des critères scalarisé où chaque vecteur de performances $u(x) \in \mathcal{Y}$ est remplacé par sa moyenne, i.e., $f_{lin}(x) = \sum_{i=1}^{n} u_i(x)$. Les solutions sont alors énumérées dans l'ordre décroissant de leur évaluation dans ce nouvel espace. Cette idée est fondée sur le résultat suivant (également exploité par Galand et Spanjaard [2012]) :

Proposition 2.7. *Soit $x \in \mathcal{X}$ une solution associée au vecteur de performances $u(x) \in \mathcal{Y}$. Pour tout vecteur de pondération $w \in W$ (à composantes décroissantes), on a :*

$$\text{OWA}_w(x) \leq \frac{1}{n} \sum_{i=1}^{n} u_i(x)$$

Nous donnons ici une nouvelle démonstration de ce résultat :

Démonstration. Soit Π l'ensemble de toutes les permutations possibles de $\{1, \ldots, n\}$. Pour toute permutation $\pi \in \Pi$, on note w_π le vecteur $(w_{\pi(1)}, \ldots, w_{\pi(n)})$. Soit π^i, $i \in \{1, \ldots, n\}$, la permutation de $\{1, \ldots, n\}$ définie par : $\pi^i(j) \equiv j + i - 1 \pmod{n}$ pour tout j dans $\{1, \ldots, n\}$. w étant à composantes décroissantes, on sait que $\text{OWA}_w(x) = \min\{w_\pi u(x) : \pi \in \Pi\}$, on en déduit que $\text{OWA}_w(x) \leq w_{\pi^i} u(x)$ pour tout $i = 1, \ldots, n$. En sommant ces inégalités sur i on obtient : $n\ \text{OWA}_w(x) \leq \sum_{i=1}^{n} w_{\pi^i} u(x)$. Or

$$\sum_{i=1}^{n} w_{\pi^i} u(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{\pi^i(j)} u_j(x) = \sum_{j=1}^{n} u_j(x) \sum_{i=1}^{n} w_{\pi^i(j)} = \sum_{j=1}^{n} u_j(x)$$

La dernière égalité vient du fait que

$$\sum_{i=1}^{n} w_{\pi^i(j)} = \sum_{i=1}^{n} w_{(j+i-1) mod\ n} = \sum_{i=0}^{n-1} w_{(j+i) mod\ n} = \sum_{i=1}^{n} w_i = 1$$

On en déduit alors le résultat $\text{OWA}_w(x) \leq \frac{1}{n} \sum_{i=1}^{n} u_i(x)$. □

La fonction moyenne nous donne donc une borne de la valeur d'un $\text{OWA}_w(x)$ lorsque w est à composantes décroissantes. Notons que cette borne est atteinte lorsque $w = (\frac{1}{n}, \ldots, \frac{1}{n})$. L'opérateur de moyenne étant linéaire, il est facile de passer de l'espace des

vecteurs de performances à l'espace des moyennes de ces vecteurs. Une fois les solutions du problème énumérées (dans l'ordre décroissant de leur évaluation selon la moyenne), elles sont évaluées selon l'opérateur OWA. Lorsque le vecteur poids w est *fixé* de manière précise, la proposition 2.8 décrit un critère d'arrêt possible pour une telle méthode d'énumération [Galand et Perny, 2006] :

Proposition 2.8. *Soit $\mathcal{X}^k = \{x^1, \ldots, x^k\}$ la liste (ordonnée) des k meilleurs éléments de $\mathcal{X}$ au sens de la moyenne des vecteurs de performances et soit x^{k^*} la solution de $\mathcal{X}^k$ avec la meilleure performance selon* OWA_w*, i.e., $x^{k^*} \in \arg\max_{x \in \mathcal{X}^k} \mathrm{OWA}_w(x)$. Étant donné un vecteur de pondération w à composantes décroissantes, on a :*

$$\mathrm{OWA}_w(x^{k^*}) \geq \frac{1}{n}\sum_{i=1}^{n} u_i(x^k) \Rightarrow x^{k^*} \in \arg\max_{x \in \mathcal{X}} \mathrm{OWA}_w(x)$$

Démonstration. $\mathcal{X}^k$ contient les k meilleures solutions du problème au sens de la moyenne. On sait alors que pour toute solution non encore énumérée $x' \in \mathcal{X} \setminus \mathcal{X}^k$:

$$\frac{1}{n}\sum_{i=1}^{n} u_i(x') \leq \frac{1}{n}\sum_{i=1}^{n} u_i(x^k) \leq \mathrm{OWA}_w(x^{k^*})$$

où la deuxième inégalité constitue l'hypothèse de la proposition. De plus, par la proposition 2.7 on sait que :

$$\mathrm{OWA}_w(x') \leq \frac{1}{n}\sum_{i=1}^{n} u_i(x')$$

On a alors $\mathrm{OWA}_w(x^{k^*}) \geq \mathrm{OWA}_w(x)$ pour toute solution $x \in \mathcal{X} \setminus \mathcal{X}^k$, avec de plus, par définition de x^{k^*}, $\mathrm{OWA}_w(x^{k^*}) \geq \mathrm{OWA}_w(x), \forall x \in \mathcal{X}^k$. On obtient alors :

$$x^{k^*} \in \arg\max_{x \in \mathcal{X}} \mathrm{OWA}_w(x)$$

□

Dans un cadre où les préférences du décideur sont précisément connues, cette proposition nous permet d'établir l'algorithme suivant : énumérer les solutions de $\mathcal{X}$ jusqu'à trouver une solution x^k vérifiant $\max_{x \in \mathcal{X}^k} \mathrm{OWA}_w(x) \geq \frac{1}{n}\sum_{i=1}^{n} u_i(x^k)$. L'exemple suivant illustre le déroulement d'un tel algorithme :

Exemple 2.3. *Prenons un exemple simple où l'ensemble des solutions $\mathcal{X} = \{x^1, \ldots, x^7\}$ est associé aux vecteurs de performances suivants : $u(x^1) = (6, 8), u(x^2) = (11, 5), u(x^3) = (16, 7), u(x^4) = (5, 3), u(x^5) = (17, 2), u(x^6) = (8, 14)$ et $u(x^7) = (10, 9)$. Les moyennes de ces vecteurs sont respectivement $7, 8, 11.5, 4, 9.5, 11$ et 9.5, les solutions sont donc énumérées dans l'ordre suivant : $x^3, x^6, x^5, x^7, x^2, x^1$ puis x^4. Supposons ici que l'on évalue ces vecteurs de performances par l'opérateur* OWA_w *de paramètre $w = (\frac{3}{5}, \frac{2}{5})$, ce qui nous mène aux valeurs* $\mathrm{OWA}_w(x^1) = 6.8$, $\mathrm{OWA}_w(x^2) = 7.4$, $\mathrm{OWA}_w(x^3) = 10.6$, $\mathrm{OWA}_w(x^4) = 3.8$, $\mathrm{OWA}_w(x^5) = 8$, $\mathrm{OWA}_w(x^6) = 10.4$ *et* $\mathrm{OWA}_w(x^7) = 9.4$ *La figure 2.9 décrit le déroulement de la procédure d'énumération ordonnée sur cet exemple. À l'énumération i, un point noir (resp. carré bleu) représente la moyenne du vecteur de performances associé à x^i (resp. la valeur* $\mathrm{OWA}_w(x^i)$*). Les courbes noire et bleue (en pointillé) représentent quant à elles la moyenne courante et la meilleure valeur selon* OWA_w *courante.*

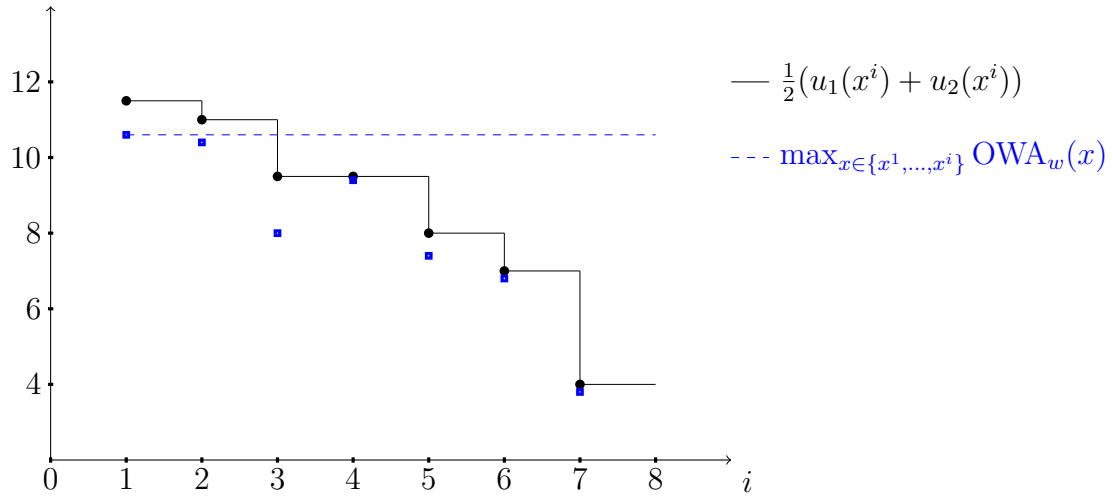


FIGURE 2.9 – Application du ranking à l'instance de l'exemple 2.3.

On voit bien sur cette figure que chaque point noir (moyenne) est situé au dessus du carré bleu correspondant (OWA), ce qui illustre la proposition 2.7. De plus, lorsqu'à l'étape 3 *la courbe bleue croise la courbe noire, on voit bien qu'il est inutile d'énumérer les solutions restantes qui ont une valeur d'OWA inférieure.*

Dans notre contexte, le poids w n'est pas fixé car les préférences du décideur ne sont pas précisément connues, il est donc impossible de calculer l'évaluation exacte d'une solution énumérée selon OWA. On ne peut donc ni déterminer la meilleure solution dans $\mathcal{X}^k$ selon cet opérateur, ni tester le critère d'arrêt donné par la proposition 2.8. Par conséquent, on ne peut pas identifier une solution optimale du problème même si elle a déjà été énumérée. On propose dans la suite une extension de cette méthode pour le calcul des solutions potentiellement OWA-optimales.

Calcul des solutions potentiellement OWA-optimales

Lorsque la valeur du paramètre w n'est pas précisément connue, on propose d'adapter la méthode de ranking au calcul des solutions potentiellement optimales de la manière suivante : les solutions sont toujours énumérées selon l'ordre décroissant des moyennes des vecteurs de performances, mais l'évaluation de toute solution énumérée est remplacée par une estimation pire cas de la valeur de l'OWA associée à x^i, i.e., par

$$\min_{w \in W} \mathrm{OWA}_w(x^i)$$

Notons que, pour une solution x fixée, la valeur $\min_{w \in W} \mathrm{OWA}_w(x)$ peut être calculée par simple programmation linéaire puisque OWA_w est linéaire en w. Cette estimation pire cas nous permet de définir un nouveau critère d'arrêt : à l'étape k de l'énumération, la meilleure estimation pire cas dans $\mathcal{X}^k$ nous informe sur la qualité des solutions énumérées et sur la nécessité de continuer l'exploration de l'espace des solutions. On arrêtera l'énumération lorsque l'on aura énuméré une solution x^k vérifiant la condition suivante :

$$\max_{x \in PO_W(\mathcal{X}^k)} \min_{w \in W} \mathrm{OWA}_w(x) > \frac{1}{n} \sum_{i=1}^{n} u_i(x^k)$$

Cette condition se justifie par la proposition suivante :

Proposition 2.9. *Soit $\mathcal{X}^k = \{x^1, \ldots, x^k\} \subseteq \mathcal{X}$ la liste (ordonnée) des k meilleurs éléments de $\mathcal{X}$ au sens de la moyenne des vecteurs de performances. On a :*

$$\max_{x \in PO_W(\mathcal{X}^k)} \min_{w \in W} \mathrm{OWA}_w(x) > \frac{1}{n} \sum_{i=1}^{n} u_i(x^k) \;\Rightarrow\; PO_W(\mathcal{X}) \subseteq PO_W(\mathcal{X}^k)$$

Démonstration. On montre que lorsque l'hypothèse de la proposition est vérifiée à l'étape k, alors tout élément n'appartenant pas à $\mathcal{X}_k$ ne peut être optimal. Si la condition est vérifiée à l'étape k, i.e., $\max_{x \in PO_W(\mathcal{X}^k)} \min_{w \in W} \mathrm{OWA}_w(x) > \frac{1}{n} \sum_{i=1}^{n} u_i(x^k)$, alors il existe une solution $y \in \mathcal{X}^k$ telle que

$$\min_{w \in W} \mathrm{OWA}_w(y) > \frac{1}{n} \sum_{i=1}^{n} u_i(x^k) \tag{2.8}$$

Considérons maintenant une solution non encore énumérée $x' \in \mathcal{X} \setminus \mathcal{X}^k$. Puisque x' arrive après x^k dans l'ordre d'énumération, on a :

$$\frac{1}{n} \sum_{i=1}^{n} u_i(x^k) \geq \frac{1}{n} \sum_{i=1}^{n} u_i(x') \tag{2.9}$$

À partir des équations (2.8) et (2.9) et du fait que $\frac{1}{n} \sum_{i=1}^{n} u_i(x') \geq \mathrm{OWA}_w(x')$ car w est à composantes décroissantes, on obtient, pour tout $w \in W$, $\mathrm{OWA}_w(y) > \mathrm{OWA}_w(x')$. On en déduit donc que $x' \notin PO_W(\mathcal{X})$ et, par conséquent, que $PO_W(\mathcal{X}) \subseteq \mathcal{X}^k$. De plus, un élément de $PO_W(\mathcal{X})$ ne peut être W-dominé par $\mathcal{X}^k$ puisque $\mathcal{X}^k \subseteq \mathcal{X}$. Alors $PO_W(\mathcal{X}) \subseteq PO_W(\mathcal{X}^k)$. □

La proposition 2.9 nous permet donc de définir une condition qui, lorsqu'elle est vérifiée, nous garantit que toute solution potentiellement OWA-optimale a déjà été énumérée, et ce, malgré une connaissance partielle des préférences.

Exemple 2.3 (suite). *Reprenons l'exemple précédent où on considère maintenant que le poids w n'est pas connu, la seule information dont on dispose est qu'il est défini dans W. La figure 2.10 illustre le déclenchement du critère d'arrêt de la proposition 2.9 sur cet exemple. Les points noirs représentent toujours la moyenne des vecteurs de performances, et les carrés bleus représentent maintenant l'estimation pire cas de la valeur de l'OWA de chaque solution. Ces estimations son obtenues pour $w = (1, 0)$ qui mène à $\mathrm{OWA}_w(x^i) = \min\{u_1(x^i), u_2(x^i)\}$.*

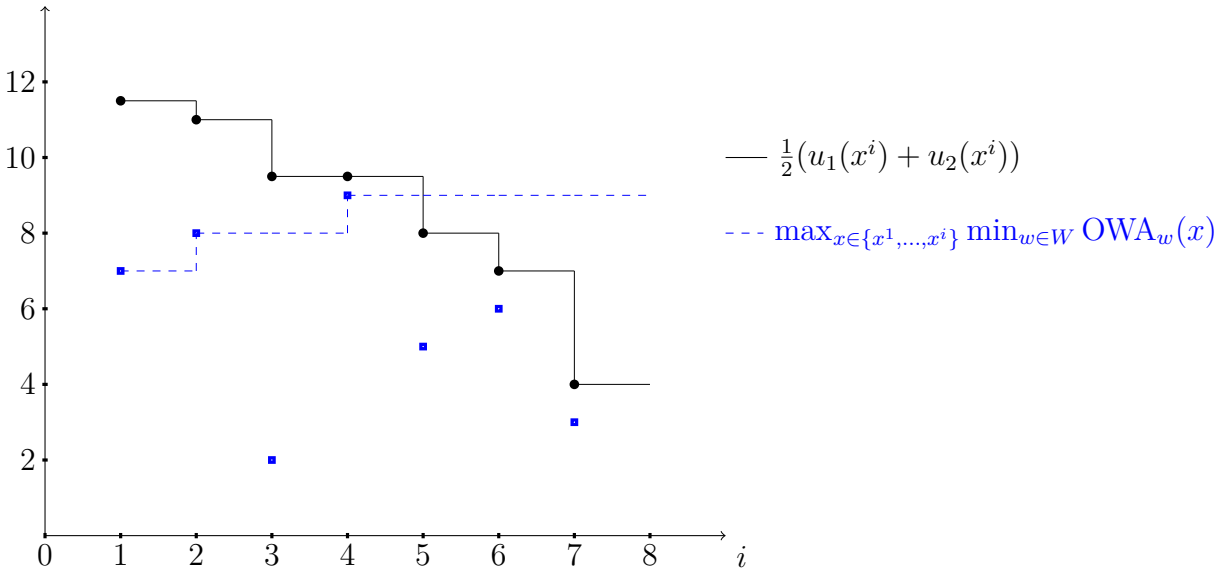


FIGURE 2.10 – Application du ranking à l'instance de l'exemple 2.3.

On voit bien sur cet exemple qu'avec une connaissance moins précise sur la valeur de w (par rapport à l'exemple précédent), on doit effectuer plus d'énumérations afin de déclencher le critère d'arrêt.

Algorithme de ranking et d'élicitation

Afin de réduire le nombre de solutions énumérées, et afin de pouvoir mieux discriminer entre ces solutions, on propose d'alterner énumération des solutions et élicitation des préférences pour préciser progressivement notre connaissance concernant la valeur de w. Ceci nous permettra, d'une part, de filtrer l'ensemble des solutions énumérées afin de ne garder que les solutions les plus intéressantes pour le décideur, et d'autre part, d'avoir des estimations pire cas des valeurs d'OWA des solutions énumérées de plus en plus proches des valeurs d'OWA réelles. On pourra alors obtenir un déclenchement du critère d'arrêt en moins d'énumérations que lorsque l'on effectue l'énumération seule. Une instanciation possible de l'algorithme général donné par l'algorithme 2.2 est alors donnée par l'algorithme 2.4.

La fonction *énumérer*$(\mathcal{X}, k)$ retourne la k^e meilleure solution de $\mathcal{X}$ au sens de la moyenne des vecteurs de performances. Si toutes les solutions du problème ont déjà été énumérées, la fonction retourne l'ensemble vide. La définition de cette fonction dépend entièrement du problème traité : il existe dans la littérature plusieurs algorithmes permettant l'énumération des solutions de différents problèmes. On présentera dans la suite les deux algorithmes que nous avons utilisés : l'algorithme de Murty [1968] pour l'énumération des solutions du problème d'affectation, et l'algorithme de Jiménez et Marzal [2003] pour l'énumération des solutions du problème du plus court chemin. En suivant le principe de l'algorithme général, des phases d'élicitation des préférences sont effectuées (ligne 9) dès lors que l'ensemble de solutions potentiellement optimales $PO_W(\mathcal{X}^k)$ atteint la taille K, où K est à définir expérimentalement. Pour cela, un compromis doit être fait :

Algorithme 2.4 : Ranking & élicitation d'un OWA

Entrées : $\mathcal{X}$: ensemble de solutions ; W : ensemble des paramètres possibles ; K : taille des sous-ensemble de solutions ; δ : seuil de tolérance.

Sorties : x^* : recommandation δ optimale

1 $PO \leftarrow \emptyset$; $k \leftarrow 1$;
2 Critère-d'arrêt $\leftarrow$ faux ;
3 **tant que** *Critère-d'arrêt = faux* **faire**
4 $\quad x^k \leftarrow$ énumérer$(\mathcal{X}, k)$;
5 $\quad$ **si** $x^k \neq \emptyset$ **alors**
6 $\quad\quad$ **si** *non(*$PO \succ_W \{x^k\}$*)* **alors**
7 $\quad\quad\quad PO \leftarrow PO \cup \{x^k\}$;
8 $\quad\quad\quad$ **si** $|PO| = K$ ***et*** $MMR(PO, W) > 0$ **alors**
9 $\quad\quad\quad\quad W \leftarrow$ Élicitation(PO, W) ;
10 $\quad\quad\quad\quad PO \leftarrow \{x \in PO : \mathrm{MR}(x, PO, W) = 0\}$;
11 $\quad\quad\quad$ **fin**
12 $\quad\quad$ **fin**
13 $\quad\quad$ **si** $\max_{x \in PO} \min_{w \in W} \mathrm{OWA}_w(x) > \frac{1}{n} \sum_{i=1}^{n} u_i(x^k) - \delta$ **alors**
14 $\quad\quad\quad$ Critère-d'arrêt $\leftarrow$ vraie ;
15 $\quad\quad$ **fin**
16 $\quad\quad k \leftarrow k + 1$;
17 $\quad$ **sinon**
18 $\quad\quad$ Critère-d'arrêt $\leftarrow$ vraie ;
19 $\quad$ **fin**
20 **fin**
21 $W \leftarrow$ Élicitation(PO, W) ;
22 $PO \leftarrow \{x \in PO : \mathrm{MR}(x, PO, W) = 0\}$;
23 **retourner** $x^* \in PO$;

un K trop petit peut nous mener à des questions trop fréquentes et pas suffisamment informatives, alors qu'une valeur de K trop grande peut nous mener à l'énumération inutile d'un grand nombre de solutions et à des phases d'élicitation trop peu fréquentes. Pour finir, une dernière phase d'élicitation est également effectuée lorsque le critère d'arrêt de la procédure d'énumération est vérifié (ligne 21) afin de déterminer l'ensemble de solutions nécessairement optimales (ligne 22). Cette dernière phase est effectuée car il peut arriver que la procédure d'énumération s'arrête alors que l'ensemble PO contient des solutions qui ne sont pas nécessairement optimales. Ce cas de figure peut se produire lorsque le critère d'arrêt se déclenche alors que l'ensemble PO est tel que $1 < |PO| < 5$ et contient des solutions qui ont un regret maximum strictement positif.

Le critère d'arrêt se déclenche sous deux conditions différentes : toutes les solutions du problème ont déjà été énumérées (ligne 18), ou lorsque la condition donnée ligne 13 est vérifiée. La valeur δ représente un seuil de tolérance à définir. Lorsque $\delta = 0$, vérifier la condition ligne 13 revient à vérifier la condition de la proposition 2.9. Dans ce cas, on résout le problème de manière exacte. Néanmoins, il peut exister des instances où le critère d'arrêt ne se déclenche qu'après avoir énuméré un très grand nombre de solutions. Afin de réduire la charge de calcul de l'algorithme, la condition d'arrêt peut être relâchée

en appliquant un seuil de tolérance $\delta > 0$. Cette relaxation permet d'obtenir, en un temps réduit, une bonne solution ayant une valeur à au plus δ de la valeur optimale (proposition 2.10).

Exemple 2.3 (suite). *Reprenons l'exemple précédent. On suppose que le poids (caché) qui représente les préférences du décideur est* $\hat{w} = (\frac{3}{5}, \frac{2}{5})$. *Supposons qu'après avoir énuméré les solutions* x^3, x^6 *et* x^5 *on pose une question au décideur en utilisant la stratégie CSS. Le décideur devra alors comparer la solution* x^3 *(déterminée par le calcul du plus petit regret maximum des solutions* x^3, x^5 *et* x^6*) et son meilleur adversaire* x^6 *(donné par le regret maximum de* x^3*). Il déclare préférer* x^3 *car* $\mathrm{OWA}_{\hat{w}}(x^3) = 10.6 > \mathrm{OWA}_{\hat{w}}(x^6) = 10.4$. *Cette réponse mène à la contrainte* $w_1 \leq 2/3$ *que l'on ajoute à la définition de l'ensemble* W. *La réduction de l'espace des paramètres nous permet de raffiner les estimations pire cas de l'OWA qui sont maintenant obtenues pour* $w = (\frac{2}{3}, \frac{1}{3})$. *Les estimations des solutions énumérées sont alors :* $\min_{w \in W} \mathrm{OWA}_w(x^3) = 10$, $\min_{w \in W} \mathrm{OWA}_w(x^5) = 7$, $\min_{w \in W} \mathrm{OWA}_w(x^6) = 10$. *La Figure 2.11 illustre le déroulement de la procédure d'énumération lorsque l'on prend en compte ces nouvelles estimations. À l'issue de la mise à jour de* W, *on voit sur cette figure que le critère d'arrêt se déclenche et que la procédure d'énumération peut s'arrêter. L'algorithme effectue alors un dernier calcul de regrets pour* $PO = \{x^3, x^5\}$ *et* $W = \{w \in \mathbb{R}^2, w_1 \geq 2/3\}$, *on trouve* $\mathrm{MMR}(PO, W) = 0$ *car* $x^3 \succ_W x^5$. *L'algorithme retourne alors la solution* x^3. *La figure 2.11 illustre le déroulement de l'algorithme sur cet exemple :*

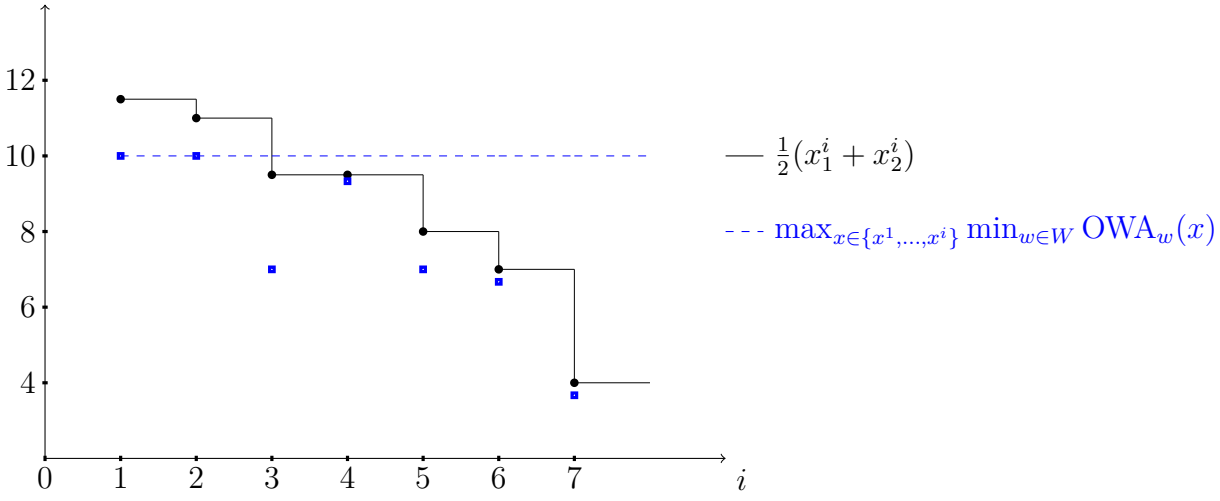


FIGURE 2.11 – Application du ranking à l'instance de l'exemple 2.3.

On voit bien ici que l'acquisition d'une information sur les préférences du décideur nous a permis de déclencher le critère d'arrêt plus tôt que dans le cas de la figure 2.10.

La pertinence de l'algorithme de ranking est donnée par la proposition suivante :

Proposition 2.10. *L'algorithme 2.4 termine et renvoie une solution* δ*-optimale.*

Avant de donner la démonstration de cette proposition, notons que si on souhaite résoudre le problème de manière exacte, on fixera la valeur de δ à 0. Ainsi, la proposition 2.10 traduit le fait que l'algorithme 2.4 retourne une solution nécessairement optimale.

Démonstration. La *terminaison* de l'algorithme est évidente : dans le pire cas, toutes les solutions du problème sont énumérées et le critère d'arrêt se déclenche. Montrons maintenant que le regret réel de la solution retournée est d'au plus δ : dans le cas où $\delta = 0$, au moment du déclenchement du critère d'arrêt, l'ensemble PO contient toutes les solutions potentiellement optimales de $\mathcal{X}$ (par la proposition 2.9). La dernière phase d'élicitation et de filtrage (lignes 21 et 22) élimine alors de PO toute solution qui n'est pas nécessairement optimale, d'où l'optimalité de x^*.

Supposons maintenant que $\delta > 0$. Notons $\hat{w}$ le poids (caché) modélisant les préférences du décideur et $\hat{x}$ la solution $\text{OWA}_{\hat{w}}$-optimale associée, et soit x^* la solution retournée par l'algorithme. Le regret réel de x^* est donné par $\text{OWA}_{\hat{w}}(\hat{x}) - \text{OWA}_{\hat{w}}(x^*)$. Notons tout d'abord que dans le cas où la solution optimale $\hat{x}$ à déjà été énumérée, la solution x^* est telle que $\text{OWA}_{\hat{w}}(\hat{x}) = \text{OWA}_{\hat{w}}(x^*)$ car à la fin de la dernière phase d'élicitation (ligne 21) on a $\text{MR}(\hat{x}, \mathcal{X}, W) = 0$ (car on sait que $\hat{x}$ est $\text{OWA}_{\hat{w}}$-optimale, que $\hat{w} \in W$ et que l'élicitation est effectuée jusqu'à atteindre $\text{MMR}(\mathcal{X}, W) = 0$), et l'opération de filtrage (ligne 22) élimine de PO toute solution x telle que $\text{MR}(x, \mathcal{X}, W) > 0$. La solution x^* est alors nécessairement optimale et donc telle que $\text{OWA}_{\hat{w}}(\hat{x}) = \text{OWA}_{\hat{w}}(x^*)$.

Supposons maintenant que la solution optimale $\hat{x}$ n'a pas encore été énumérée lorsque la procédure d'énumération est interrompue, i.e. $\hat{x} \in \mathcal{X} \setminus \mathcal{X}^k$. La solution $\hat{x}$ est alors telle que :

$$\frac{1}{n}\sum_{i=1}^{n} u_i(\hat{x}) \leq \frac{1}{n}\sum_{i=1}^{n} u_i(x^k)$$

car $\hat{x}$ doit être énumérée après la solution x^k. On a de plus :

$$\min_{w \in W} \text{OWA}_w(x^*) \leq \text{OWA}_{\hat{w}}(x^*) \leq \text{OWA}_{\hat{w}}(\hat{x})$$

puisque $\hat{w} \in W$ et puisque $\hat{x}$ est $\text{OWA}_{\hat{w}}$-optimale. On a alors :

$$\min_{w \in W} \text{OWA}_w(x^*) \leq \text{OWA}_{\hat{w}}(x^*) \leq \text{OWA}_{\hat{w}}(\hat{x}) \leq \frac{1}{n}\sum_{i=1}^{n} u_i(\hat{x}) \leq \frac{1}{n}\sum_{i=1}^{n} u_i(x^k)$$

Or, la solution retournée x^* est telle que $\frac{1}{n}\sum_{i=1}^{n} u_i(x^k) - \min_{w \in W} \text{OWA}_w(x^*) < \delta$. De plus, la distance $\text{OWA}_{\hat{w}}(\hat{x}) - \text{OWA}_{\hat{w}}(x^*)$ est donc bornée par la distance entre la moyenne courante $\frac{1}{n}\sum_{i=1}^{n} u_i(x^k)$ et l'estimation $\min_{w \in W} \text{OWA}_w(x^*)$ qui est elle-même bornée par δ (par définition du critère d'arrêt), i.e., :

$$\text{OWA}_{\hat{w}}(\hat{x}) - \text{OWA}_{\hat{w}}(x^*) \leq \frac{1}{n}\sum_{i=1}^{n} u_i(x^k) - \min_{w \in W} \text{OWA}_w(x^*) < \delta$$

□

Une conséquence de cette proposition est que l'algorithme 2.4 bénéficie d'une propriété intéressante : cet algorithme est *anytime*. On peut en effet s'arrêter à une étape k quelconque de l'algorithme en ayant une garantie sur la qualité de la solution renvoyée, i.e., la distance entre la valeur de la moyenne courante et la valeur $\min_{w \in W} \text{OWA}_w(x^*)$ nous procure une borne sur l'erreur commise en recommandant x^*.

Expérimentations numériques

Afin de tester l'efficacité de l'algorithme 2.4 (*Ranking & Élicitation d'un OWA*), nous l'avons implémenté et testé sur des instances des problèmes d'*affectation robuste* et de *plus court chemin robuste*. Avant de présenter les résultats numériques obtenus, nous commençons par définir ces problèmes ainsi que les algorithmes d'énumération associés que nous avons utilisés.

a. Problème d'affectation robuste

Le problème d'affectation consiste à attribuer des objets (ou des tâches) à des agents de manière à ce que chaque objet soit attribué à exactement un agent. Dans la version *robuste* du problème, différents scénarios sont envisagés et chaque agent associe une utilité à chaque objet dans chacun des scénarios considérés. La satisfaction globale des agents dans chaque scénario est donnée par la somme des utilités des agents (pour le scénario en question) pour les objets qui leur ont été attribués. Une solution donnée est évaluée selon l'utilité globale des agents dans tous les scénarios possibles, l'objectif est de trouver une solution robuste dans ce contexte, i.e., une solution assurant une bonne satisfaction globale quel que soit le scénario qui se réalise.

On définit une instance de ce problème par un ensemble de p agents, p objets et n scénarios. L'ensemble des affectations réalisables $\mathcal{X}$ est constitué de matrices binaires X de taille $p \times p$ dont chaque composante x_{ij} est une variable de décision binaire qui prend la valeur 1 si et seulement si l'objet j est attribué à l'agent i. La contrainte $\sum_{i=1}^{p} x_{ij} = 1$ traduit le fait que l'objet j doit être attribué à exactement un agent, tandis que la contrainte $\sum_{j=1}^{p} x_{ij} = 1$ traduit le fait qu'exactement un objet est attribué à l'agent i. L'ensemble des solutions réalisables est donc défini par : $\mathcal{X} = \{x = (x_{ij})_{i\in\{1,\dots,p\},j\in\{1,\dots,p\}} | x_{ij} \in \{0,1\}, \sum_{i=1}^{p} x_{ij} = 1, \sum_{j=1}^{p} x_{ij} = 1, i \in \{1,\dots,p\}, j \in \{1,\dots,p\}\}$. L'utilité de l'agent i pour une solution $x \in \mathcal{X}$ est donnée par $u^i(x) = (\sum_{j=1}^{p} u_{ij}^1 x_{ij}, \dots, \sum_{j=1}^{p} u_{ij}^n x_{ij})$ où u_{ij}^k est l'utilité que l'agent i associe à l'objet j dans le scénario k. Finalement, l'utilité globale de la solution x est donnée par $u(x) = (\sum_{i=1}^{p} u_1^i(x), \dots, \sum_{i=1}^{p} u_n^i(x))$. On se place ici dans le cas où les préférences du décideur sont représentées par l'opérateur OWA, la qualité de la solution x est donc évaluée par $\text{OWA}_w(x) = \sum_{i=1}^{n} w_i u_{(i)}(x)$ où $w \in W$ représente le vecteur de pondération à éliciter et $u_{(i)}$ est la i^e plus petite composante de $u(x)$.

Exemple 2.4. *On considère une instance du problème d'affectation robuste à* 4 *agents,* 4 *objets et* 3 *scénarios. Chaque matrice* U_k *donne en ligne* i *colonne* j *l'utilité* u_{ij}^k *qu'attribue l'agent* i *à l'objet* j *dans le scénario* k*.*

$$U_1 = \begin{pmatrix} 10 & 3 & 6 & 4 \\ 10 & 1 & 0 & 1 \\ 9 & 9 & 4 & 0 \\ 4 & 2 & 3 & 4 \end{pmatrix} \quad U_2 = \begin{pmatrix} 10 & 5 & 0 & 2 \\ 6 & 7 & 1 & 4 \\ 7 & 0 & 7 & 3 \\ 1 & 9 & 10 & 2 \end{pmatrix} \quad U_3 = \begin{pmatrix} 0 & 2 & 8 & 0 \\ 2 & 0 & 6 & 5 \\ 4 & 9 & 4 & 2 \\ 7 & 8 & 3 & 9 \end{pmatrix}$$

Par exemple, la solution x*, définie par* $x_{ij} = 1$ *pour* $(i,j) \in \{((1,3),(2,1),(3,2),(4,4)\}$ *et* $x_{ij} = 0$ *sinon, est une affectation réalisable de cette instance et mène au vecteur d'utilités* $u(x) = (6+10+9+4, 0+6+0+2, 8+2+9+9) = (29, 8, 28)$*. Pour tout vecteur* $w \in W$*, cette solution est évaluée par* $\text{OWA}_w(x) = 8w_1 + 28w_2 + 29w_3$*.*

Notons que ce problème est NP-difficile même dans le cas où le paramètre w est connu de manière précise : l'opérateur OWA contient en particulier l'opérateur min (en prenant

$w = (1, 0, \ldots, 0)$), or il a été démontré par Kouvelis et Yu [2013] que la maximisation du pire scénario pour le problème d'affection robuste est un problème NP-difficile même lorsque l'on ne considère que 2 scénarios.

Énumération ordonnée des solutions du problème d'affectation robuste Afin d'instancier la fonction *énumérer*$(\mathcal{X}, k)$ de l'algorithme 2.4 nous avons utilisé une méthode introduite par Murty [1968]. Elle consiste à effectuer plusieurs résolutions successives de sous-problèmes d'affectation obtenus par des partitionnements de l'espace des solutions.

Pour énumérer les solutions de ce problème dans l'ordre décroissant des moyennes d'utilité on procède de la manière suivante : on calcule tout d'abord la solution moyenne-optimale en utilisant par exemple la méthode Hongroise. On trouve alors une solution que l'on note x^1. Soit $\text{Ind}(x^1)$ l'ensemble des couples d'indices (i, j) tels que $x_{ij} = 1$:

$$\text{Ind}(x^1) = \{(1, o^1), (2, o^2), .., (p, o^p)\}$$

L'ensemble des solutions réalisables $\mathcal{X}$ est partitionné à l'aide de la solution x^1 de la manière suivante :

$$\mathcal{X} = \{x^1\} \cup \mathcal{P}^1 \cup \mathcal{P}^2 \cup .. \cup \mathcal{P}^{p-1}$$

où $\mathcal{P}^i = \{x \in \mathcal{X} | x_{jo^j} = 1, \forall j \in \{1, \ldots, i-1\} \text{ et } x_{io^i} = 0\}$, i.e., les $i-1$ objets attribués aux $i-1$ premiers agents sont les mêmes que ceux de la solution x^1, l'objet attribué au i^e agent est différent de celui de x^1 et le reste de la solution peut être construit librement.

Une fois cette partition obtenue, on résout le sous-problème induit par chaque $\mathcal{P}^i$ obtenant ainsi des solutions $y^i \in \arg\max_{x \in \mathcal{P}^i} \sum_{i=1}^{n} u_i(x)$ pour tout i. On peut alors déterminer la deuxième meilleure solution par :

$$x^2 = y^r \quad \text{avec} \quad r = \arg\max_{i \in \{1,\ldots,p-1\}} \frac{1}{n} \sum_{j=1}^{n} u_j(y^i)$$

On partitionne maintenant l'ensemble $\mathcal{P}^r$ en utilisant x^2 de la même manière que pour $\mathcal{X}$. On résout ensuite le sous-problème induit par chaque ensemble de la partition obtenue ajoutant ainsi les solutions optimales associées aux solutions y^i, $i \in \{1, \ldots, p-1\}, i \neq r$, calculées à l'étape précédente, celle de plus grande moyenne nous donne alors x^3. On procède de la même manière pour énumérer les autres solutions du problème, alternant ainsi partitionnement et résolution de sous-problèmes d'affectation. L'énumération se termine lorsque tout ensemble de la partition est réduit à un seul élément et ne peut être partitionné d'avantage.

La complexité de cet algorithme est polynomiale en p, chaque résolution de sous-problème est d'une complexité d'au plus $\text{O}(p^4)$ si elle est réalisée par l'algorithme Hongrois, l'énumération de k solutions par cette approche est donc d'une complexité de $\text{O}(kp^4)$.

Pour instancier la fonction *énumérer*$(\mathcal{X}, k)$ de l'algorithme 2.4, on effectuera une étape de la procédure d'énumération de Murty : en utilisant la dernière solution énumérée x^{k-1}, on met à jour la partition de $\mathcal{X}$ en partitionnant l'ensemble $\mathcal{P}^i$ qui a mené à la détermination de x^{k-1}, puis on calcule les solutions moyenne-optimales y^j selon chaque sous-problème induit par les nouveaux ensembles $\mathcal{P}^j$ de la partition de $\mathcal{P}^i$, et on détermine la solution de plus grande moyenne qui nous donne x^k.

Avant de présenter nos résultats numériques, on donne ici un exemple du déroulement de l'algorithme 2.4 :

Exemple 2.4 (suite). *On illustre ici le déroulement de l'algorithme 2.4 sur l'instance d'affectation robuste donnée dans l'exemple 2.4. Dans cet exemple, on simule les réponses du décideur en utilisant un OWA de paramètre* $\hat{w} = (1/2, 1/3, 1/6)$*, on effectue une phase d'élicitation dès que* $|PO| = 5$ *et on prend un seuil de tolérance* $\delta = 0$*. La meilleure solution au sens de la moyenne est donnée par* x^1 *définie par* $x^1_{ij} = 1$ *pour tout couple* $(i,j) \in \{(1,3),(2,1),(3,2),(4,4)\}$ *et* $x^1_{ij} = 0$ *sinon. Cette solution est associée au vecteur d'utilités* $u(x^1) = (29, 8, 28)$ *d'une moyenne de* 21.67*. On estime la valeur d'OWA de* x^1 *en résolvant* $\min_{w\in W} \mathrm{OWA}_w(x1) = 8w_1 + 28w_2 + 29w_3$*, on obtient* $\min_{w\in W} \mathrm{OWA}_w(x^1) = 8$*. Comme la distance entre* $\sum_{i=1}^{3} x^1_i = 21.67 > 8$*, alors une autre itération de l'algorithme est effectuée. L'énumération ordonnée continue jusqu'à l'étape 5 où l'algorithme calcule la valeur du MMR car toutes les solutions énumérées sont insérées dans PO, i.e., que l'on a atteint* $|PO| = 5$*. On trouve une valeur MMR strictement positive alors une question est posée au décideur : on lui demande de comparer la* 2^e *solution énumérée* x^2*, définie par* $x^2_{ij} = 1$ *pour tout couple* $(i,j) \in \{(1,1),(2,4),(3,2),(4,3)\}$ *et* $x^1_{ij} = 0$ *sinon et associée à* $u(x^2) = (23, 24, 17)$*, à la solution lui assurant un regret maximum. Cette solution est donnée par* x^4 *associée au vecteur d'utilité* $u(x^4) = (18, 20, 25)$*. Le décideur préfère* x^2 *car* $\mathrm{OWA}_{\hat{w}}(x^2) = 20.17 \geq \mathrm{OWA}_{\hat{w}}(x^4) = 19.83$*. On met alors à jour la définition de* W *par* $W = \{w \in W | \mathrm{OWA}_w(x^2) \geq \mathrm{OWA}_w(x^4)\}$*. Cette contrainte se traduit par* $17w_1 + 23w_2 + 24w_3 \geq 18w_1 + 20w_2 + 25w_3$ *ou encore* $-w_1 + 3w_2 - w_3 \geq 0$*. Après l'ajout de la contrainte et le filtrage de PO, la valeur du MMR est toujours positive, une autre question est alors posée. Le décideur doit cette fois comparer* x^2 *et son pire adversaire qui est maintenant* x^1*. Il préfère toujours* x^2*, on met alors à jour les ensembles* W *et* PO *à l'aide de cette nouvelle information de préférences de la même manière que pour la première question. Après ces mises à jour, la valeur du MMR est nulle et la phase d'élicitation peut s'arrêter. On reprend alors l'énumération avec* $PO = \{x^2\}$ *jusqu'à l'étape 15 sans insérer aucune solution à PO car toutes les solutions énumérées sont dominées par* x^2 *(par la dominance de Pareto ou de* $W-$*dominance). À cette étape de l'algorithme, le critère d'arrêt se déclenche et l'algorithme s'arrête en retournant la solution* x^2*. La figure 2.12 illustre l'évolution de la moyenne et de l'estimation de l'OWA.*

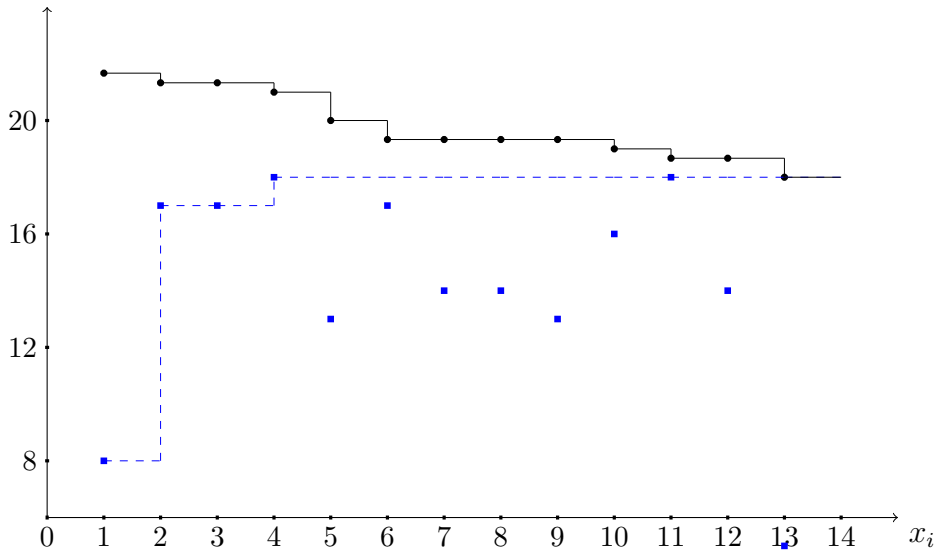


FIGURE 2.12 – Application de l'algorithme 2.4 à l'instance d'affectation robuste de l'exemple 2.4

On voit bien sur cette figure, que même si la solution optimale est énumérée très tôt, plusieurs itérations peuvent être nécessaires au déclenchement du critère d'arrêt. Le choix du seuil δ a alors une influence significative sur le nombre d'itérations exécutées. En prenant, par exemple, un seuil $\delta = 10\%$ de l'erreur initiale, le critère d'arrêt se déclencherait après avoir énuméré seulement 6 solutions.

Résultats numériques Nous avons appliqué l'algorithme 2.4 à différentes instances du problème d'affectation robuste. Pour cela, nous avons généré aléatoirement des instances de différentes tailles : $p \in \{20, 60, 100\}$ agents et $n \in \{3, 5, 10\}$ scénarios. La procédure pouvant être relativement longue pour certaines tailles d'instances, nous fixons le seuil d'erreur δ à 10% de l'erreur initiale (mesurée par la distance entre la moyenne de la première solution énumérée et son estimation pire cas de l'OWA) et nous imposons une limite de temps d'exécution de 20 minutes. Nous avons évalué l'efficacité de cette approche en termes de temps de calcul (en secondes), du nombre de solutions énumérées, du nombre de questions posées ainsi que du *gap* à la valeur optimale. Dans le cas où l'algorithme est interrompu avant que le critère d'arrêt ne se déclenche, la valeur du *gap* représente l'erreur maximum commise en recommandant la solution retournée par l'algorithme (proposition 2.10) exprimée en pourcentage de la borne supérieure de l'échelle de valeurs. Une phase d'élicitation (ligne 9 de l'algorithme) est effectuée à chaque fois que l'ensemble PO atteint une taille de $K = 5$. Pour finir, afin de simuler les réponses du décideur, on génère un poids caché aléatoirement dans W.

Pour ces expérimentations numériques [1], nous utilisons la librairie gurobipy de Python pour la résolution des différents programmes linéaires de l'algorithme (calculs de regrets, vérifications de relations de W-dominances et estimations pire cas de l'OWA). Les tables 2.1, 2.2 et 2.3 résument les résultats de ces expérimentations. Pour chaque couple de valeurs (p, n) le résultat donné est une moyenne sur 20 instances.

p	n = 3			
	temps(s)	solutions énumérées	questions	gap(%)
20	274	13607	1.3	0
60	1200	7236	4	1.1
100	1200	2075	5.1	2

Table 2.1 – Algorithme 2.4 pour le problème d'affectation robuste - 3 scénarios.

p	n = 5			
	temps(s)	solutions énumérées	questions	gap(%)
20	719	34009	2.3	0.5
60	1200	6945	6.7	5
100	1200	2550	11.3	6

Table 2.2 – Algorithme 2.4 pour le problème d'affectation robuste - 5 scénarios.

1. L'algorithme a été implémenté en Python. Les expérimentations ont été effectuées sur une machine Intel Core i7-4770 CPU avec 11GB de RAM

p	n = 10			
	temps(s)	solutions énumérées	questions	gap(%)
20	940	37739	2	1.4
60	1200	6382	11.6	4.7
100	1200	2216	17.5	7.5

TABLE 2.3 – Algorithme 2.4 pour le problème d'affectation robuste - 10 scénarios.

Les 3 tableaux montrent que l'algorithme 2.4 est relativement efficace. Il permet d'obtenir de bons résultats même dans le cas où l'algorithme est interrompu sans que la condition d'arrêt ne soit vérifiée : on voit bien que la valeur du *gap* est relativement faible. Le deuxième point positif que l'on peut observer est le faible nombre de questions posées. Il est, par exemple, d'au plus 5 questions en moyenne dans le cas de 3 scénarios quelle que soit la taille de l'instance traitée. Ce nombre augmente lorsque le nombre de scénarios augmente, ce qui s'explique par le fait que le modèle est plus complexe et requiert plus de questions afin d'être suffisamment précis pour pouvoir comparer les solutions de compromis énumérées. Il ne dépasse les 12 questions en moyenne que pour les plus grandes tailles d'instances à 100 agents et 10 scénarios.

Finalement, on peut observer que l'exécution de l'algorithme est relativement coûteuse en temps. La première colonne des tableaux montre que les temps d'exécution dépassent les 20 minutes à partir de $p = 60$ agents pour toutes les valeurs n considérées. Notons que ce temps est essentiellement dû à l'énumération des solutions. En effet, le nombre de solutions énumérées baisse considérablement lorsque p augmente ; on passe, par exemple pour $n = 10$, d'une vitesse d'énumération d'une solution toutes les 0.02 secondes pour $p = 20$ à une vitesse d'une solution toutes les 0.54 secondes. De plus, le nombre de questions indique que les phases d'élicitation sont peu nombreuses et la détermination des questions est très rapide car les calculs de regrets ne concernent que des ensembles de solutions de taille 5. Le temps d'exécution est donc essentiellement consommé par l'énumération des solutions qui engendre un temps moyen entre 2 questions relativement grand (entre 2 et 7 minutes). On peut tout de même voir, grâce à la valeur du *gap*, que l'interruption prématurée de l'algorithme ne détériore que très peu la qualité de la recommandation faite.

b. Problème du plus court chemin robuste

Afin de tester la généralité de l'algorithme 2.4 (*Ranking & Élicitation d'un OWA*), on s'intéresse maintenant au problème du plus court chemin robuste. Étant donné un graphe orienté où les temps de parcours des arcs sont évalués selon différents scénarios, on cherche à trouver le chemin dont la somme des durées des arcs est la plus petite possible quel que soit le scénario considéré.

On définit une instance de ce problème par un ensemble de sommets S dont un sommet de départ s et un sommet destination t. Les sommets de S sont reliés par un ensemble d'arcs noté A, où le temps de parcours de chaque arc est évalué selon n scénarios. L'ensemble des solutions réalisables $\mathcal{X}$ est alors constitué de matrices binaires X dont chaque composante x_{ij} est une variable de décision binaire qui prend la valeur 1 si et seulement si l'arc allant du sommet i vers le sommet j est emprunté. Ces variables sont soumises

aux contraintes

$$\sum_{j \in succ(i)} x_{ij} - \sum_{j \in pred(i)} x_{ij} = \begin{cases} 1 & i = s \\ -1 & i = t \\ 0 & sinon \end{cases}$$

où $succ(i)$ (resp. $pred(i)$) donne la liste des successeurs (resp. prédécesseurs) de i dans le graphe. Cette famille de contraintes exprime le fait que toute solution est telle que s (resp. t) possède un arc sortant (resp. entrant) et pour tout sommet autre que s et t, le nombre d'arcs sortants doit être égal au nombre d'arcs entrants. Ainsi, les variables x_{ij} forment un chemin entre s et t. Le temps de parcours d'un chemin réalisable $x \in \mathcal{X}$ est donné par $u(x) = (\sum_{i \in S} \sum_{j \in succ(i)} x_{ij} u^1_{ij}, \ldots, \sum_{i \in S} \sum_{j \in succ(i)} x_{ij} u^n_{ij})$ où u^k_{ij} est le temps de parcours de l'arc (i, j) dans le scénario k. Dans le cas où les préférences du décideur sont représentées par l'opérateur OWA, la qualité de la solution x est évaluée par $\text{OWA}_w(x) = \sum_{i=1}^n u_{(i)}(x)$ où $w \in W$ représente le vecteur de pondération à éliciter et $u_{(i)}$ est la i^e plus grande composante de $u(x)$. On cherche ici à déterminer une solution dans $\arg\min_{x \in \mathcal{X}} \text{OWA}_{\hat{w}}(x)$ où $\hat{w}$ représente le poids réel du décideur.

Exemple 2.5. *On définit une instance du problème du plus court chemin robuste à* 3 *scénarios dont le graphe est représenté par la figure 2.13.*

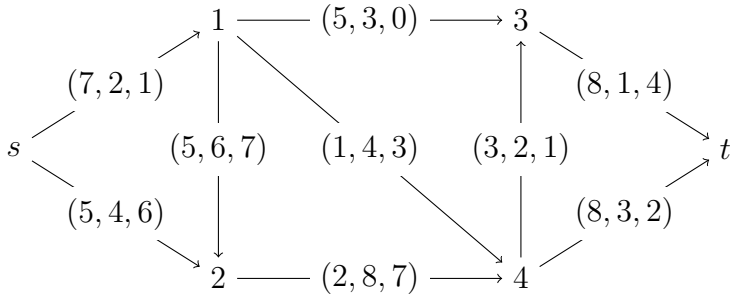


FIGURE 2.13 – Exemple de graphe pour le problème du plus court chemin robuste à 3 scénarios.

Le chemin réalisable c constitué des arcs $(s, 1), (1, 3)$ et $(3, t)$ est associé au vecteur $u(c) = (20, 6, 5)$. Pour un vecteur de pondération $w \in W$ donné, la qualité de c est évaluée par $\text{OWA}_w(c) = 20w_1 + 6w_2 + 5w_3$.

Ce problème est NP-difficile : Kouvelis et Yu [2013] ont démontré que la minimisation du pire scénario pour le problème du plus plus court chemin robuste est un problème NP-difficile même lorsque l'on ne considère que 2 scénarios. Or l'opérateur OWA contient en particulier l'opérateur max (en prenant $w = (1, 0, \ldots, 0)$).

Algorithme de ranking et d'élicitation pour la minimisation d'un OWA L'objectif ici étant de minimiser la valeur d'un OWA, l'application de l'algorithme 2.4 à ce problème nécessite l'adaptation de sa définition à ce contexte : les propositions 2.7 (borne de la valeur de l'OWA) et 2.9 (critère d'arrêt du ranking) ainsi que l'ordre d'énumération, et le critère d'arrêt.

Les résultats des propositions 2.7 et 2.9 s'adaptent très facilement lorsque l'on change le sens de l'optimisation. En effet, lorsque l'on considère un problème de minimisation, le

calcul d'$\text{OWA}_w(x)$ pour toute solution $x \in \mathcal{X}$ se fait en triant le vecteur de performances de x dans le sens inverse, i.e., dans l'ordre décroissant des composantes du vecteur de performances afin de toujours associer les poids les plus élevés aux composantes les moins performantes. Dans ce cas, la borne de la valeur de l'OWA devient :

Proposition 2.11. *Soit $w \in W$ un vecteur de pondération à composantes décroissantes et soit $x \in \mathcal{X}$ une solution associée au vecteur de performances $u(x) \in \mathcal{Y}$. On a :*

$$\text{OWA}_w(x) \geq \frac{1}{n} \sum_{i=1}^{n} u_i(x)$$

Démonstration. La preuve de cette proposition est similaire à celle de la proposition 2.7. On utilisera ici le fait que, dans un cadre de minimisation, $\text{OWA}_w(x) = \max\{w_\pi u(x) : \pi \in \Pi\}$, où Π est l'ensemble de toutes les permutations de $\{1, \ldots, n\}$. □

On peut alors concevoir un algorithme d'énumération ordonnée des solutions du problème dans l'ordre croissant des moyennes des vecteurs de performances. Le critère d'arrêt de l'algorithme sera alors fondé sur la proposition suivante :

Proposition 2.12. *Soit $\mathcal{X}^k = \{x^1, \ldots, x^k\}$ la liste des k meilleurs éléments de $\mathcal{X}$ au sens de la moyenne croissante des vecteurs de performances. On a :*

$$\min_{x \in PO_W(\mathcal{X}^k)} \max_{w \in W} \text{OWA}_w(x) < \frac{1}{n} \sum_{i=1}^{n} u_i(x^k) \;\Rightarrow\; PO_W(\mathcal{X}) \subseteq PO_W(\mathcal{X}^k)$$

Démonstration. En utilisant la proposition 2.11 pour borner la valeur d'un OWA, on peut construire cette preuve de la même manière que celle de la proposition 2.9. □

On pourra ainsi adapter l'algorithme 2.4 au contexte de minimisation en utilisant ces deux dernières propositions. De manière évidente, l'algorithme conservera les propriétés de terminaison, de validité ainsi que la garantie de performance de la proposition 2.10 (que l'on démontrera de manière similaire).

Avant de présenter l'algorithme d'énumération de chemins pour finaliser la définition de l'algorithme de ranking, notons que d'autres notions seront à adapter au contexte de minimisation : définitions des solutions potentiellement et nécessairement optimales, relations de W-dominance et de W-dominance d'ensembles. De plus, le regret maximum par paire de deux solutions $x \in \mathcal{X}$ et $y \in \mathcal{X}$ se définit maintenant par $\text{PMR}(x, y, W) = \max_{w \in W}\{\text{OWA}_w(x) - \text{OWA}_w(y)\}$. Et finalement, vérifier si une solution x est dominée par l'ensemble $\mathcal{X}$, i.e., si $\mathcal{X} \prec_W \{x\}$, revient maintenant à vérifier que la valeur obtenue en optimisant $\max_{w \in W} \min_{y \in \mathcal{X}} f_w(y) - f_w(x)$ est strictement négative.

Énumération ordonnée des solutions du problème du plus court chemin robuste Afin d'énumérer les chemins réalisables d'une instance de ce problème dans l'ordre croissant des moyennes des vecteurs de performances, on utilise l'algorithme de Jiménez et Marzal [2003]. Il consiste à construire progressivement une structure de données complexe permettant l'énumération ordonnée des chemins par un parcours d'arbre.

L'idée de base de cet algorithme consiste à sauvegarder les déviations possibles du plus court chemin dans une structure de tas. Cette structure nous permet ensuite de construire les alternatives possibles au plus court chemin dans l'ordre croissant des temps

de la manière suivante : on commence tout d'abord par calculer l'arborescence des plus courts chemins de n'importe quel sommet v du graphe G vers le sommet final t (en utilisant Dijkstra par exemple). À l'aide de cette arborescence, on calcule le graphe des regrets qui contient les sommets et arcs de G et est évalué de la manière suivante : le regret de tout arc (u, v) de G est donné par la perte de temps engendrée par le choix de (u, v) pour aller de u à t au lieu du plus court chemin allant de u à t. On en arrive alors à la construction de la structure contenant les déviations au plus court chemin : pour chaque sommet v du graphe, on construit un tas $H_{out}(v)$ contenant tous les arcs sortants de v, à l'exception de l'arc appartenant au plus court chemin de v à t, rangés dans l'ordre croissant de leurs regrets. $H_{out}(v)$ contient alors un sous ensemble de déviations du plus court chemin allant de v à t. Afin de stocker la totalité de ces déviations, on construit le tas général $H_G(v)$ contenant les tas $H_{out}(w)$ pour tout w appartenant au plus court chemin de v à t, rangés dans l'ordre croissant des regrets des racines des $H_{out}(w)$. On note $D(G)$ la structure globale contenant $H_G(v)$ pour tout sommet v de G. La structure D nous permet alors d'accéder à toutes les déviations d'un chemin, dans l'ordre croissant des regrets.

Le détail de la méthode d'énumération est donné par l'algorithme 2.5 :

Algorithme 2.5 :

Entrées : G : graphe orienté ; $l(e)$: longueur de l'arc e ; s : source du graphe ; t : puits du graphe

Sorties : $\mathcal{X}^k$: ensemble des k premiers chemins dans l'ordre croissant des coûts

1 Calculer l'arborescence des plus court chemin T ;
2 Calculer le graphe des regrets R ;
3 $\mathcal{X}^1 \leftarrow$ {plus court chemin de s à t} ;
4 $H_G(s) \leftarrow$ *Calcul_*$H_G(s)$;
5 $Q \leftarrow racine(H_G(s))$; `/* plus petite déviation du plus court chemin */`
6 $k \leftarrow 2$;
7 **tant que** Q *est non vide* **faire**
8 Extraire l'ensemble de déviation S_k de Q de plus petit regret ;
9 Construire un chemin de déviation x^k à partir de S_k ;
10 $\mathcal{X}^k \leftarrow \mathcal{X}^{k-1} \cup \{x^k\}$;
11 $(u, v) \leftarrow$ dernière déviation de S_k ;
12 $H_G(u) \leftarrow$ *Calcul_*$H_G(u)$;
13 $(u', v') \leftarrow racine(H_G(u))$ `/* plus petite déviation de` x^k `à partir de` u `*/` ;
14 $Q \leftarrow S_k \cup \{(u', v')\}$ avec le score $\sum_{e \in S_k} l(e) + R(u', v')$;
15 **pour toute déviation** (u, v) *tel que* (u, v) *est un arc dans* $D(G)$ **faire**
16 Insérer $Q \leftarrow S_k \setminus \{(u, v)\} \cup \{(u', v')\}$ avec le score $\sum_{e \in S_k} l(e) - R(u, v) + R(u', v')$;
17 **fin**
18 $k \leftarrow k + 1$;
19 **fin**
20 **retourner** $\mathcal{X}^{k-1}$

Dans cet algorithme, le calcul des tas H_G se fait à l'aide la fonction *Calcul_*$H_G(v)$ donnée par l'algorithme 2.6 :

Algorithme 2.6 : *Calcul_*$H_G(v)$

Entrées : v : sommet du graphe
Sorties : $H_G(v)$

```
1  si H_G(v) n'a pas encore été construit alors
2  |  Construire H_out(v)
3  |  si v = t alors
4  |  |  H_G(v) = H_out(v)
5  |  sinon
6  |  |  Appeler Calcul_H_G(suivant_T(v))
7  |  |  Insérer H_out(v) dans H_G(suivant_T(v)) pour former H_G(v)
8  |  fin
9  fin
10 retourner H_G(v)
```

Dans cet algorithme, $suivant_T(v)$ désigne le sommet de G suivant v dans l'arborescence T. Et Q désigne une liste de priorité contenant les déviations possibles des plus courts chemins énumérés triées par valeurs croissantes de regrets.

La complexité pire cas de l'énumération de k solutions en utilisant cet algorithme est en $O(|A| + |S| \log(|S|) + klog(k))$ [Jiménez et Marzal, 2003].

Résultats numériques Nous avons appliqué l'algorithme 2.4 à différentes instances du problème du plus court chemin robuste. Pour cela nous avons généré aléatoirement des instances contenant $p \in \{500, 1000, 1500\}$ nœuds de degré au plus $\frac{p}{2}$. Les évaluations des arcs selon les $n \in \{3, 5, 10\}$ scénarios sont également générées aléatoirement. Nous avons utilisé les mêmes paramètres que pour les expérimentations précédentes : le seuil d'erreur δ est fixé à 10% de l'erreur initiale et la limite de temps imposée est de 20 minutes. Les phases d'élicitation (ligne 9 de l'algorithme) sont effectuées dès que PO atteint une taille de $K = 5$. Et le poids caché représentant les préférences du décideur est généré aléatoirement dans W. On évalue l'efficacité de l'algorithme selon les mêmes critères : temps de calcul (en secondes), nombre de solutions énumérées, nombre de questions posées et *gap* (distance maximum à la valeur optimale exprimée en pourcentage de la borne supérieure de l'échelle de valeurs).

Les tables 2.4, 2.5 et 2.6 résument les résultats de ces expérimentations. Pour chaque couple de valeurs p, n donnée, on effectue une moyenne sur 20 instances aléatoires :

p	n = 3			
	temps(s)	solutions énumérées	questions	gap(%)
500	183.7	9595	2.8	0.0
1000	337.1	9830	4.7	0.0
1500	365.5	8591	5.1	0.0

TABLE 2.4 – Algorithme 2.4 pour le problème du plus court chemin robuste - 3 scénarios.

p	n = 5			
	temps(s)	solutions énumérées	questions	gap(%)
500	548.0	28461	5.0	0.1
1000	612.3	28524	5.9	0.1
1500	802.9	22894	7.6	0.0

TABLE 2.5 – Algorithme 2.4 pour le problème du plus court chemin robuste - 5 scénarios.

p	n = 10			
	temps(s)	solutions énumérées	questions	gap(%)
500	681.2	41797	6.3	0.1
1000	852.7	43971	6.7	0.1
1500	892.2	29864	10.5	0.1

TABLE 2.6 – Algorithme 2.4 pour le problème du plus court chemin robuste - 10 scénarios.

Les 3 tableaux montrent que l'algorithme 2.4 est encore plus efficace sur les instances étudiées de ce problème que sur les instances du problème précédent. Les solutions recommandées sont systématiquement optimales ou très proches de la solution optimale : la valeur de la recommandation est à une distance d'au plus 0.1% de la valeur optimale. On observe ici aussi un nombre de questions posées raisonnable. Ce nombre augmente lorsque le nombre de scénarios augmente mais est inférieur à 11 questions en moyenne dans tous les cas.

Finalement, les temps d'exécution observés sont relativement bons, les instances sont en moyenne résolues dans la limite de temps imposée (en moins de 20 minutes). En comparaison avec les expérimentations précédentes, on peut tout d'abord remarquer que l'augmentation du temps d'exécution lorsque la taille de l'instance augmente n'est pas aussi importante que précédemment. De plus, la vitesse d'énumération est d'en moyenne une solution toutes les 0.03 secondes, le temps moyen entre deux questions est alors réduit à 1 minute 30 secondes, ce qui est plus rapide que pour les plus petites instances d'affectation robuste considérées.

2.3.3 Algorithmes de ranking pour la recherche d'une solution WOWA-optimale

Lorsque les préférences du décideur sont représentées par l'opérateur WOWA, l'approche de ranking présentée précédemment n'est plus valide car la moyenne ne constitue pas une borne pour cet opérateur :

Exemple 2.6. *Prenons par exemple une solution x associée au vecteur de performances $u(x) = (29, 8, 28)$. Supposons que les probabilités des scénarios soient données par $p = (1/2, 1/6, 1/3)$ et que les préférences du décideur soient représentées par la fonction de déformation $\varphi(r) = r^2, \forall r \in [0, 1]$. On voit bien ici que la moyenne de $u(x)$ ne constitue pas une borne supérieure de la valeur de WOWA puisque* $\text{WOWA}_\varphi(x) = 8 + (28 - 8)\varphi(5/6) + (29 - 28)\varphi(1/2) = 22.14 > \frac{1}{3}\sum_{i=1}^{3} u_i(x) = 21.67$.

Afin d'adapter l'approche, on doit trouver une borne linéaire de la valeur de WOWA afin de déterminer une règle d'énumération ainsi qu'un critère d'arrêt efficaces. On utilisera ici la somme pondérée par les probabilités $p_1, \ldots, p_n$:

Proposition 2.13. *Soient $p_1, \ldots, p_n$ les probabilités des n scénarios et soit $x \in \mathcal{X}$ une solution associée au vecteur de performances $u(x)$. Pour toute fonction $\varphi \in \Phi$, on a :*

$$\text{WOWA}_\varphi(x) \leq \sum_{i=1}^{n} p_i u_i(x)$$

Démonstration. La fonction φ est convexe, on a alors $\varphi(ta+(1-t)b) \leq t\varphi(a)+(1-t)\varphi(b)$ pour tout $a, b, t \in [0,1]$. En fixant $a = 1$ et $b = 0$ on obtient : $\varphi(t) \leq t$ pour tout $t \in [0,1]$. De ce fait, $\varphi(\sum_{k=i}^{n} p_{(k)}) \leq \sum_{k=i}^{n} p_{(k)}$ pour $i = 1, \ldots, n$. Ce qui nous donne :

$$\text{WOWA}_\varphi(x) = \sum_{i=1}^{n} \left[x_{(i)} - x_{(i-1)}\right] \varphi(\sum_{k=i}^{n} p_{(k)}) \leq \sum_{i=1}^{n} \left[x_{(i)} - x_{(i-1)}\right] \sum_{k=i}^{n} p_{(k)}$$

puisque $x_{(i)} - x_{(i-1)} \geq 0$ pour tout $i = 1, \ldots, n$. Finalement, on obtient :

$$\text{WOWA}_\varphi(x) \leq \sum_{i=1}^{n} \left[\sum_{k=i}^{n} p_{(k)} - \sum_{k=i+1}^{n} p_{(k)}\right] x_{(i)} = \sum_{i=1}^{n} p_{(i)} x_{(i)} = \sum_{i=1}^{n} p_i x_i$$

□

La somme pondérée par les probabilités $p_1, \ldots, p_n$ nous offre alors une borne de la valeur de WOWA d'une solution. Notons tout d'abord que cette borne est atteinte puisqu'en prenant $\varphi(r) = r, \forall r \in [0,1]$, on obtient $\text{WOWA}_\varphi(x) = \sum_{i=1}^{n} \left[x_{(i)} - x_{(i-1)}\right] \sum_{k=i}^{n} p_{(k)} = \sum_{i=1}^{n} [\sum_{k=i}^{n} p_{(k)} - \sum_{k=i+1}^{n} p_{(k)}] x_{(i)} = \sum_{i=1}^{n} p_{(i)} x_{(i)} = \sum_{i=1}^{n} p_i x_i$.

On utilisera cette somme pondérée pour effectuer une énumération ordonnée des solutions de $\mathcal{X}$ ainsi que pour définir un critère d'arrêt de la procédure d'énumération (de manière similaire à ce que l'on a fait pour OWA). Cet opérateur étant linéaire, il est facile de passer de l'espace des vecteurs de performances à l'espace scalarisé des sommes pondérées de ces vecteurs. De plus, il respecte le principe d'optimalité de Bellman, ce qui nous permet d'utiliser des méthodes d'énumérations constructives ou par opérations d'échanges de sous-solutions comme nous avons pu le faire précédemment.

Afin de définir un critère d'arrêt pour le calcul des solutions de l'ensemble $PO_\Phi(\mathcal{X})$, nous établissons la proposition suivante :

Proposition 2.14. *Soient $p_1, \ldots, p_n$ les probabilités des n scénarios, et soit $\mathcal{X}^k 0$ la liste des k meilleurs éléments de $\mathcal{X}$ notée $\mathcal{X}^k = \{x^1, \ldots, x^k\}$ au sens de la somme pondérée par $p_1, \ldots, p_n$. On a :*

$$\max_{x \in PO_\Phi(\mathcal{X}^k)} \min_{\varphi \in \Phi} \text{WOWA}_\varphi(x) > \sum_{i=1}^{n} p_i u_i(x^k) \Rightarrow PO_\Phi(\mathcal{X}) \subseteq PO_\Phi(\mathcal{X}^k)$$

Notons que, pour toute solution x fixée, la valeur $\min_{\varphi \in \Phi} \text{WOWA}_\varphi(x)$ peut être calculée par simple programmation linéaire si on utilise la formulation spline donnée en sous-section 2.2.2 puisque $\text{WOWA}_\varphi = \text{WOWA}_\alpha$ est linéaire en α. La démonstration de ce résultat est similaire à la démonstration de la proposition 2.9 :

Démonstration. On montre que lorsque l'hypothèse de la proposition est vérifiée à l'étape k, alors tout élément n'appartenant pas à $\mathcal{X}_k$ ne peut être optimal : si la condition est vérifiée à l'étape k, i.e., $\max_{x \in PO_\Phi(\mathcal{X}^k)} \min_{\varphi \in \Phi} \text{WOWA}_\varphi(x) > \sum_{i=1}^n p_i x_i^k$, alors il existe une solution $y \in \mathcal{X}^k$ telle que :

$$\min_{\varphi \in \Phi} \text{WOWA}_\varphi(y) > \sum_{i=1}^n p_i x_i^k \tag{2.10}$$

Considérons maintenant une solution non encore énumérée $x' \in \mathcal{X} \setminus \mathcal{X}^k$. Puisque x' arrive après x^k dans l'ordre d'énumération, on a :

$$\sum_{i=1}^n p_i x_i^k \geq \sum_{i=1}^n p_i x'_i \geq \text{WOWA}_\varphi(x') \tag{2.11}$$

pour toute fonction $\varphi \in \Phi$. À partir des équations (2.10) et (2.11) on obtient, pour toute fonction $\varphi \in \Phi$, $\text{WOWA}_\varphi(y) > \text{WOWA}_\varphi(x')$ et donc $x' \notin PO_\Phi(\mathcal{X})$. Ceci montre que $PO_\Phi(\mathcal{X}) \subseteq \mathcal{X}^k$. De plus, un élément de $PO_\Phi(\mathcal{X})$ ne peut être $\succ_\Phi$-dominé dans $\mathcal{X}^k$ puisque $\mathcal{X}^k \subseteq \mathcal{X}$. Alors $PO_\Phi(\mathcal{X}) \subseteq PO_\Phi(\mathcal{X}^k)$. □

Cette proposition nous permet de définir un critère d'arrêt qui, lorsqu'il est vérifié, nous garantit que toute solution potentiellement WOWA-optimale a déjà été énumérée.

Algorithme de ranking et d'élicitation de WOWA

On appliquera ici, en utilisant notre nouveau critère d'énumération et notre nouvelle condition d'arrêt, la même procédure que l'algorithme 2.4 consistant à alterner énumération des solutions et phases d'élicitation par la stratégie CSS. L'approche est détaillée par l'algorithme 2.7.

Dans cet algorithme, la fonction *énumérer*$(\mathcal{X}, k, p)$ retourne la k^e meilleure solution de $\mathcal{X}$ au sens de la somme pondérée par les probabilités p. En utilisant une base de fonctions splines pour représenter φ, on élicite un vecteur de poids α que l'on sait appartenir au simplexe. De ce fait, en dehors de la définition de la fonction *énumérer*$(\mathcal{X}, k, p)$ et du critère d'arrêt, l'algorithme est similaire à l'algorithme 2.4 (*Ranking & élicitation d'un OWA*). Afin d'accélérer la procédure, on peut ici aussi relâcher le critère d'arrêt en fixant un seuil de tolérance δ strictement positif. On peut alors démontrer la proposition suivante :

Proposition 2.15. *L'algorithme 2.7 termine et renvoie une solution δ-optimale.*

Notons que si l'on souhaite résoudre le problème de manière exacte, on fixera la valeur de δ à 0. Ainsi, cette proposition traduit le fait que l'algorithme 2.7 retourne une solution nécessairement optimale. La preuve de cette proposition est identique à la preuve de la proposition 2.10 à la différence qu'elle utilise la proposition 2.14 pour la définition du critère d'arrêt.

Algorithme 2.7 : Ranking & élicitation d'un WOWA

Entrées : $\mathcal{X}$: ensemble de solutions ; $\mathcal{A}$: ensemble des paramètres possibles ; p : vecteur de probabilités des scénarios ; K : taille des sous-ensembles de solutions ;
δ : seuil de tolérance.

Sorties : x^* : recommandation δ optimale

$PO \leftarrow \emptyset$; $k \leftarrow 1$;
Critère-d'arrêt $\leftarrow$ faux ;
tant que *Critère-d'arrêt = faux* **faire**
  $x^k \leftarrow$ énumérer$(\mathcal{X}, k, p)$;
  si $x^k \neq \emptyset$ **alors**
    si *non(*$PO \succ_{\mathcal{A}} \{x^k\}$*)* **alors**
      $PO \leftarrow PO \cup \{x^k\}$;
      si $|PO| = K$ ***et*** $\mathrm{MMR}(PO, \mathcal{A}) > 0$ **alors**
        $\mathcal{A} \leftarrow$ Élicitation$(PO, \mathcal{A})$;
        $PO \leftarrow \{x \in PO : \mathrm{MR}(x, PO, \mathcal{A}) = 0\}$;
      fin
    fin
  sinon
    Critère-d'arrêt $\leftarrow$ vraie ;
    $\mathcal{A} \leftarrow$ Élicitation$(PO, \mathcal{A})$;
    $PO \leftarrow \{x \in PO : \mathrm{MR}(x, PO, \mathcal{A}) = 0\}$;
  fin
  si $\max_{x \in PO} \min_{\alpha \in \mathcal{A}} \mathrm{WOWA}_\alpha(x) > \sum_{i=1}^n p_i x_i - \delta$ **alors**
    Critère-d'arrêt $\leftarrow$ vraie ;
  fin
  $k \leftarrow k + 1$;
fin
retourner $x^* \in PO$;

Expérimentations numériques

Afin de tester l'efficacité de l'approche de ranking pour l'élicitation de la fonction de déformation φ de l'opérateur WOWA, nous avons implémenté l'algorithme 2.7 et nous l'avons testé sur des instances aléatoires des deux problèmes précédents : l'affectation robuste et le plus court chemin robuste. La fonction *énumérer*$(\mathcal{X}, k, p)$ est implémentée tel que présenté dans la sous-section précédente, en remplaçant simplement la moyenne des vecteurs de performances par leur somme pondérée par les probabilités p. Les paramètres de l'algorithme ainsi que des instances générées sont les mêmes que pour les expérimentations précédentes : $\delta = 10\%$ de la distance initiale, une phase d'élicitation est effectuée dès que PO atteint une taille de $K = 5$. Les instances du problème d'affectation sont générées pour $p \in \{20, 60, 100\}$ et $n \in \{3, 5, 10\}$, celles du problème du plus court chemin pour des instances contenant $p \in \{500, 1000, 1500\}$ sommets de degrés au plus $\frac{p}{2}$, avec des arcs évalués selon $n \in \{3, 5, 10\}$ scénarios. Finalement, les réponses du décideur sont simulées par une fonction de déformation $\varphi = \sum_{i=1}^5 \alpha_i s_i$ où les fonctions s_i sont des

fonctions C-Splines définies par l'équation (2.5) et où α est un vecteur de pondération généré aléatoirement dans le simplexe de dimension 5. Les tables 2.7 et 2.8 résument les résultats obtenus.

p	n = 3			
	temps(s)	solutions énumérées	questions	gap(%)
20	270	5338	10.3	1
60	1200	5446	8.95	8.1
100	1200	2978	6.15	10.3

p	n = 5			
	temps(s)	solutions énumérées	questions	gap(%)
20	475	4657	9.45	1.1
60	1200	3192	5.3	7.6
100	1200	2302	5	15.9

p	n = 10			
	temps(s)	solutions énumérées	questions	gap(%)
20	865	6743	9.2	1.7
60	1200	4381	5.4	6.5
100	1200	1901	4.9	11.3

TABLE 2.7 – Algorithme 2.7 pour le problème d'affectation robuste.

p	n = 3			
	temps(s)	solutions énumérées	questions	gap(%)
500	181.0	5822	3.3	0.0
1000	190.7	1317	5.4	0.0
1500	632.0	2327	6.6	0.0

p	n = 5			
	temps(s)	solutions énumérées	questions	gap(%)
500	400.7	17540	5.5	0.1
1000	544.0	16340	7.2	0.1
1500	785.9	13886	8.9	0.1

p	n = 10			
	temps(s)	solutions énumérées	questions	gap(%)
500	1187.5	38677	8.5	1.0
1000	1200	34788	8.4	0.8
1500	1200	27447	9.7	1.2

TABLE 2.8 – Algorithme 2.7 pour le problème du plus court chemin robuste.

On peut observer une efficacité relativement bonne sur l'ensemble des instances considérées, avec globalement de meilleures performances moyennes pour les instances du problème du plus court chemin robuste pour tous les critères d'évaluation considérés, et ce malgré les tailles d'instances relativement grandes.

En conclusion des expérimentations effectuées pour les algorithmes de ranking (tables 2.1 à 2.8), on peut dire que les résultats obtenus sont relativement satisfaisants en termes de temps de calcul et de qualité de la solution recommandée. Notons que l'application de cet algorithme nécessite l'existence d'une borne efficace sur la valeur de l'opérateur d'agrégation ainsi que l'existence d'une procédure d'énumération efficace. Il est également à noter qu'il peut exister des instances pathologiques pour lesquelles un très grand nombre de solutions (voire toutes les solutions) devront être énumérées avant que le critère d'arrêt ne soit vérifié. Par exemple, une instance pour laquelle tous les vecteurs de performances associés aux différentes solutions de l'instance ont la même moyenne (ou la même somme pondérée pour WOWA). Un tel algorithme ne convient pas à la recherche d'une solution OWA ou WOWA optimale pour de telles instances.

2.3.4 Algorithme de Branch and Bound interactif

La seconde méthode que nous proposons pour explorer efficacement l'espace des solutions d'un problème est une approche interactive par séparation et évaluation, que l'on appelle plus communément *Branch and Bound*. Une telle méthode repose sur une énumération implicite des solutions par l'exploration d'un arbre de recherche binaire dont chaque nœud représente un sous-problème du problème associé à son nœud parent. Plus précisément, l'algorithme de Branch and Bound consiste à alterner successivement deux phases : une phase d'évaluation qui consiste à estimer la qualité d'un nœud en déterminant des bornes (supérieure et inférieure) sur la qualité des solutions qu'il contient, et une phase de séparation (ou de branchement) qui construit progressivement l'arbre d'exploration en développant un nœud considéré comme intéressant (selon les phases d'évaluation précédentes) en divisant le problème associé en deux sous-problèmes distincts. L'énumération s'arrête lorsque tous les nœuds intéressants ont été explorés.

Lorsque les préférences du décideur ne sont pas précisément connues, on définit un algorithme de *Branch and Bound interactif* qui alterne ces deux phases avec des phases d'élicitation permettant de préciser les préférences du décideur. On présente ici ces 3 phases plus en détail dans le cadre de l'élicitation d'un OWA pour la résolution d'un problème à variables entières. Notons qu'une approche similaire peut être définie pour WOWA en utilisant simplement les résultats équivalents de cet opérateur comme nous l'avons précédemment fait pour l'algorithme de ranking.

Phase d'évaluation - Bounding

L'évaluation d'un nœud η de l'arbre d'exploration a pour objectif d'évaluer la qualité des solutions du sous-problème associé sans avoir à les énumérer explicitement, on saura alors si η peut mener ou non à une solution prometteuse afin de déterminer s'il est intéressant de l'explorer. Cette évaluation se fait par le calcul de deux bornes : une borne supérieure nous donnant la meilleure valeur d'OWA que l'on peut espérer atteindre en explorant les solutions du sous-problème associé à η, on note ces solutions $\mathcal{X}^\eta$. Et une borne inférieure nous donnant une estimation pire cas, selon la connaissance du paramètre w, de la valeur d'OWA d'une solution de $\mathcal{X}^\eta$. Nous verrons que la pertinence de cette borne dépend fortement de nos connaissances des préférences du décideur, d'où la nécessité d'avoir recours à une procédure d'élicitation.

L'efficacité de l'algorithme de branch and bound dépend fortement de la pertinence de ces bornes qui guident la recherche d'une solution optimale. Celles-ci nous permettront d'estimer la localisation, dans l'arbre d'exploration, des solutions intéressantes pour le décideur. En effet, ces bornes sont utilisées d'une part pour le choix du prochain nœud à explorer (phase de séparation), et d'autre part pour établir une règle d'élagage, i.e., une règle nous permettant d'éliminer un nœud si on sait qu'il ne mènera pas à une solution optimale. De ce fait, des bornes trop larges peuvent mener à très peu d'élagages, et donc à un nombre de solutions énumérées trop important. On donne ici des définitions de bornes ainsi que deux règles d'élagages :

1. Borne supérieure : la borne supérieure du noeud η, notée $BS(\eta)$, est une estimation optimiste de la qualité d'une bonne solution (au sens de l'OWA) dans $\mathcal{X}^\eta$. On propose de définir cette borne en utilisant la fonction moyenne :

$$BS(\eta) = \max_{x \in \mathcal{X}_r^\eta} \frac{1}{n} \sum_{i=1}^{n} u_i(x)$$

où $\mathcal{X}_r^\eta$ est la relaxation continue de $\mathcal{X}^\eta$. La validité de cette borne est donnée par la proposition suivante :

Proposition 2.16. *Soient $\mathcal{X}^\eta$ l'ensemble des solutions réalisables du sous-problème associé au nœud η et $\mathcal{X}_r^\eta$ sa relaxation continue. Soit $x \in \mathcal{X}^\eta$ une solution réalisable. Pour tout vecteur de pondération $w \in W$, on a :*

$$\mathrm{OWA}_w(x) \leq \max_{y \in \mathcal{X}_r^\eta} \frac{1}{n} \sum_{i=1}^{n} u_i(y)$$

Démonstration. Cette proposition découle directement de la proposition 2.7. On sait que $\forall x \in \mathcal{X}^\eta, \forall w \in W, \mathrm{OWA}_w(x) \leq \frac{1}{n}\sum_{i=1}^{n} u_i(x) \leq \max_{y \in \mathcal{X}^\eta} \frac{1}{n}\sum_{i=1}^{n} u_i(y)$ pour toute solution $x \in \mathcal{X}$. On sait de plus que $\max_{y \in \mathcal{X}^\eta} \frac{1}{n}\sum_{i=1}^{n} u_i(y) \leq \max_{y \in \mathcal{X}_r^\eta} \frac{1}{n}\sum_{i=1}^{n} u_i(y)$ par définition de X_r^η, ce qui conclut la démonstration. □

Cette borne peut être obtenue par programmation linéaire si le problème étudié possède une formulation en programme linéaire, ou par une méthode plus efficace qui est spécifique au problème comme nous le verrons dans la partie expérimentations numériques.

2. Borne inférieure : cette borne, notée $BI(\eta)$, donne une estimation pessimiste de la valeur OWA-optimale dans $\mathcal{X}^\eta$. Lorsque le paramètre w est précisément connu, on peut simplement définir BI comme la valeur d'OWA d'une solution dans $\mathcal{X}^\eta$ obtenue heuristiquement (voir [Galand et Spanjaard, 2012] par exemple). Lorsque w n'est pas précisément connu, on propose d'utiliser l'estimation pire cas $\min_{w \in W} \mathrm{OWA}_w(x)$ pour une solution donnée $x \in \mathcal{X}^\eta$. Afin de définir une borne non triviale, et que $BI(\eta)$ soit la plus informative possible, on propose d'utiliser la projection entière de la solution moyenne-optimale relaxée calculée pour la détermination de $BS(\eta)$: soit x_r^* une solution telle que $x_r^* \in \arg\max_{y \in \mathcal{X}_r^\eta} \frac{1}{n}\sum_{i=1}^{n} u_i(y)$, on définit une bonne solution entière $x^* \in \mathcal{X}^\eta$ à partir de la solution continue x_r^* en projetant x_r^* dans l'espace $\mathcal{X}^\eta$. L'opération de projection dépend du problème traité et, plus particulièrement, de la nature des variables de décision associées, mais consiste généralement à arrondir les composantes continues à

l'entier (supérieur ou inférieur) le plus proche de manière à respecter les contraintes de $\mathcal{X}^\eta$. Une fois cette solution obtenue, on définit notre borne inférieure par :

$$BI(\eta) = \min_{w \in W} \text{OWA}_w(x^*)$$

L'idée derrière l'utilisation de la moyenne pour BS et BI est la même que pour l'algorithme de ranking présenté dans la sous-section précédente, on s'attend à ce qu'une solution performante au sens de la moyenne le soit aussi au sens de l'opérateur OWA. La borne BI nous permet également d'avoir une borne sur le regret réel donné par la distance entre la valeur de la recommandation x^* (par rapport aux préférences cachées du décideur) et la valeur OWA-optimale (selon les préférences cachées du décideur) :

Proposition 2.17. *Soient $\hat{w}$ le vecteur poids (caché) représentant les préférences du décideur et $\hat{x}$ la solution $\text{OWA}_{\hat{w}}$-optimale associée. Soit $x^* \in \mathcal{X}^\eta$ la projection de la solution moyenne-optimale relaxée x_r^* dans $\mathcal{X}^\eta$, alors :*

$$\text{OWA}_{\hat{w}}(\hat{x}) \geq \min_{w \in W} \text{OWA}_w(x^*)$$

Démonstration. Par définition de $\hat{w}$ et de $\hat{x}$ on sait que $\text{OWA}_{\hat{w}}(\hat{x}) \geq \text{OWA}_{\hat{w}}(x), \forall x \in \mathcal{X}$, donc en particulier $\text{OWA}_{\hat{w}}(\hat{x}) \geq \text{OWA}_{\hat{w}}(x^*)$. On en déduit alors la validité de l'inégalité puisque $\text{OWA}_{\hat{w}}(x^*) \geq \min_{w \in W} \text{OWA}_w(x^*)$. □

Au vu des informations acquises sur les préférences du décideur, la meilleure borne inférieure possible de la valeur $\text{OWA}_{\hat{w}}(\hat{x})$ est donnée par $\max_\eta BI(\eta)$. Les bornes $BS(\eta), BI(\eta)$ et $\max_\eta BI(\eta)$ donnent une évaluation de l'intérêt de chaque nœud de l'arbre d'exploration et permettent de déterminer les nœuds à explorer et les nœuds à élaguer.

3. Règles d'élagage : une règle d'élagage permet d'écarter de l'exploration toute solution ou tout nœud ne menant pas à une solution potentiellement ou nécessairement optimale. On propose dans un premier temps d'utiliser nos bornes pour identifier les nœuds qui ne seront pas développés dans la suite. On note $\mathcal{N}^k$ l'ensemble des nœuds non élagués ni développés à l'étape k de l'algorithme et $\eta^* \in \mathcal{N}^k$ un nœud de l'arbre tel que $BI(\eta^*) = \max_{\eta \in \mathcal{N}^k} BI(\eta)$ (la meilleure évaluation pire cas de la valeur OWA-optimale). On définit alors la règle suivante :

Règle d'élagage R1 : tout nœud $\eta \in \mathcal{N}^k$ tel que $BS(\eta) < BI(\eta^*)$ est élagué de l'arbre.

La pertinence de cette borne est donnée par la proposition suivante :

Proposition 2.18.

$$\forall \eta \in \mathcal{N}^k, BS(\eta) < BI(\eta^*) \Rightarrow PO_W(\mathcal{X}^\eta) \cap PO_W(\mathcal{X}) = \emptyset$$

Démonstration. Notons η' un nœud vérifiant la condition $BS(\eta') < BI(\eta^*)$ et montrons que toute solution de $\mathcal{X}^{\eta'}$ ne peut être potentiellement optimale. Par définition de la borne supérieure on sait que toute solution $x' \in \mathcal{X}^{\eta'}$ est telle que, pour tout $w \in W$, $\text{OWA}_w(x') \leq BS(\eta') < BI(\eta^*)$. Or, $BI(\eta^*) = \min_{w \in W} \text{OWA}_w(x^*)$, alors pour tout vecteur poids $w \in W$ et pour toute solution $x' \in \mathcal{X}^{\eta'}$ on a $\text{OWA}_w(x') < \text{OWA}_w(x^*)$. On en déduit que $x^* \succ_W x', \forall x' \in \mathcal{X}^{\eta'}$. Par conséquent, aucune solution dans $\mathcal{X}^{\eta'}$ ne peut être potentiellement optimale. □

Lorsque l'on explore les nœuds de l'arbre et que l'on calcule leur borne inférieure, on énumère un certain nombre de solutions réalisables pouvant être intéressantes pour le décideur. Notons $\mathcal{X}^k$ l'ensemble des solutions distinctes énumérées à l'étape k de l'algorithme, et PO l'ensemble où on conserve toutes les solutions potentiellement optimales de $\mathcal{X}^k$. On peut définir une règle d'élagage sur l'ensemble PO.

Règle d'élagage R2 : toute solution $x \in \mathcal{X}^k$ telle que $x \notin PO_W(\mathcal{X}^k)$ n'est pas insérée dans PO.

L'ensemble $\mathcal{X}^k$ étant un sous-ensemble de $\mathcal{X}$, toute solution n'étant pas potentiellement optimale dans $\mathcal{X}^k$ ne l'est pas non plus dans $\mathcal{X}$. Appliquer cette règle d'élagage revient à filtrer l'ensemble $\mathcal{X}^k$ en n'ajoutant à PO que les solutions x pour lesquelles la condition $\mathcal{X}^k \succ_W \{x\}$ n'est pas vérifiée.

Phase de séparation - Branching

Cette phase vise à explorer l'espace des solutions et consiste à choisir un nœud intéressant η non encore développé et à créer deux nœuds enfants η' et η'' en partitionnant le sous-problème associé à η en deux sous-problèmes distincts. L'exploration de l'espace des solutions associé à η a pour but d'affiner les bornes $BS(\eta)$ et $BI(\eta)$ afin de mieux orienter la recherche d'une solution optimale.

Soit η le nœud à développer, et soit $\mathcal{X}^\eta$ l'ensemble des solutions du sous-problème associé. Durant la phase d'évaluation, une solution moyenne-optimale continue x_r^* a été calculée pour définir la borne supérieure $BS(\eta)$. Cette solution possède au moins une composante continue, notons la $x_{r_i}^*$. S'il n'existe pas une telle composante, on cherche un autre nœud à développer, s'il en existe plusieurs, on en choisit une arbitrairement. On crée deux nœuds enfants η' et η'' à partir du nœud η en utilisant $x_{r_i}^*$ de la manière suivante : η' représente le sous-problème caractérisé par $\mathcal{X}^{\eta'} = \{x \in \mathcal{X}^\eta | x_i \leq \lfloor x_{r_i} \rfloor\}$ tandis que η'' représente le sous problème caractérisé par $\mathcal{X}^{\eta''} = \{x \in \mathcal{X}^\eta | x_i \geq \lceil x_{r_i} \rceil\}$. L'espace des solutions associé à η est donc partitionné en deux sous-espaces $\mathcal{X}^{\eta'}$ et $\mathcal{X}^{\eta''}$ et de nouvelles bornes doivent être calculées. Notons que les nouvelles bornes supérieures sont telles que $BS(\eta') \leq BS(\eta)$ et $BS(\eta'') \leq BS(\eta)$ par définition de $\mathcal{X}^{\eta'}$ et $\mathcal{X}^{\eta''}$. De plus, la meilleure estimation pire cas définie par $\max_{\eta \in \mathcal{N}^k} BI(\eta)$ ne décroit pas (par définition) durant l'exploration de l'espace des solutions. De ce fait, plus on développe des nœuds en profondeur de l'arbre d'exploration, plus les bornes BS seront petites et plus on aura de chances d'élaguer des nœuds par la règle **R1**. De plus, la projection d'une solution d'un espace continu $\mathcal{X}_r^\eta$ à un espace entier $\mathcal{X}^\eta$ peut parfois mener à des solutions sous-optimales. Le développement du nœud η peut alors permettre de trouver une meilleure solution entière et d'améliorer la borne $BI(\eta)$ et donc, par conséquent, d'améliorer $\max_{\eta \in \mathcal{N}^k} BI(\eta)$. Pour ces deux raisons, on propose ici d'effectuer une exploration en profondeur.

Phase d'élicitation incrémentale

La procédure de branch and bound constituée des deux phases de séparation et d'évaluation décrites plus haut énumère progressivement l'ensemble des solutions potentiellement optimales de $\mathcal{X}$. Comme proposé dans l'algorithme général 2.2, on alternera l'énumération avec des phases d'élicitation des préférences pour mieux discriminer entre les

solutions trouvées. Pour cela, on appliquera la stratégie CSS aux solutions énumérées (non élaguées par **R2**) en appliquant *Élicitation*$(\mathcal{X}^k, W)$.

Algorithme de Branch and Bound Interactif

L'idée de l'algorithme est donc d'alterner les phases d'évaluation, de séparation et d'élicitation présentées plus haut jusqu'à ce qu'il n'y ait plus de nœuds à développer dans l'arbre. L'approche globale est détaillée dans l'algorithme 2.8 qui fait appel à la fonction récursive de l'algorithme 2.9.

Algorithme 2.8 : *Branch & Bound interactif pour l'élicitation d'un OWA*

Entrées : $\mathcal{X}$: ensemble des solutions réalisables défini implicitement ; W : espace des paramètres ; K : taille des sous-ensembles de solutions.
Sorties : x^* : recommandation optimale

1 $PO \leftarrow \emptyset$; `/* solutions PO parmi les solutions énumérées */`
2 $maxBI \leftarrow 0$; `/* meilleure borne pire cas */`
3 BB($\mathcal{X}$) ;
4 **répéter**
5 | $\mathcal{W} \leftarrow$ Élicitation(PO, W) ;
6 | $PO \leftarrow \{x \in PO : \text{MR}(x, PO, W) = 0\}$;
7 **jusqu'à** MMR(PO, W) $= 0$;
8 **retourner** $x \in PO$;

Les variables PO, W et $maxBI$ sont ici des variables globales de l'algorithme 2.8 qui peuvent être modifiées par tout appel de la fonction récursive $BB(\mathcal{X}^\eta)$. On distingue ici les trois phases principales de l'approche : évaluation de la ligne 3 à la ligne 5, élicitation de la ligne 9 à la ligne 14 et finalement séparation de la ligne 16 à la ligne 23. Lorsque la procédure d'énumération est terminée et que la valeur du MMR est strictement positive, une dernière phase d'élicitation est effectuée afin de ne garder dans l'ensemble PO que les solutions nécessairement optimales. On démontre la pertinence de cette approche par la proposition suivante :

Proposition 2.19. *L'algorithme 2.8 termine et retourne une solution nécessairement optimale.*

Démonstration. Remarquons tout d'abord que, sans les règles **R1** et **R2**, l'algorithme consiste simplement en une énumération exhaustive de l'ensemble des solutions de $\mathcal{X}$, l'algorithme est donc fini. Appliquer la règle **R1** nous permet d'écarter de l'exploration tout nœud contenant uniquement des solutions dominées (proposition 2.18). L'élagage d'un nœud par la règle **R1** ne peut alors pas mener à l'exclusion du nœud contenant la solution optimale pour le décideur. De plus, la phase d'élicitation réduit peu à peu l'ensemble des paramètres possibles W. Si on note W^i l'ensemble des poids possibles après la question i, on peut voir facilement que toute opération d'élagage par **R1** ou **R2** à l'issue de la question i reste valide à l'issue de la question $i+1$ car $W^i \subset W^{i+1}$. Finalement, toute opération d'élagage faite par **R1** ou **R2** reste valide pour l'espace des paramètres final. Lorsque la procédure s'arrête, la valeur du MMR est nulle, les solutions restantes dans PO sont donc toutes nécessairement optimales. □

Algorithme 2.9 : BB($\mathcal{X}^\eta$)

Entrées : $\mathcal{X}^\eta$: ensemble de solutions du sous-problème associé au nœud η.

Déterminer une solution $x_r^* \in \arg\max_{x \in \mathcal{X}_r^\eta} \frac{1}{n} \sum_{i=1}^n u_i(x)$;
Projeter x_r^* dans $\mathcal{X}^\eta$ pour obtenir x^* ;
$BS(\eta) = \frac{1}{n} \sum_{i=1}^n u_i(x^* r)$;
$BI(\eta) = \min_{w \in W} \mathrm{OWA}_w(x^*)$;
$maxBI \leftarrow \max\{maxBI, BI(\eta)\}$;
si $BS(\eta) > maxBI^*$ **alors**
  si *non*$(PO \succ_W \{x\})$ **alors**
    $PO \leftarrow PO \cup \{x\}$;
    si $|PO| = K$ **alors**
      répéter
        $\mathcal{W} \leftarrow$ Élicitation(PO, W) ;
        $PO \leftarrow \{x \in PO : \mathrm{MR}(x, PO, W) = 0\}$;
      jusqu'à $\mathrm{MMR}(PO, W) = 0$;
    fin
  fin
  $\mathcal{X}^{\eta'} = \{x \in \mathcal{X}^\eta | x_i < x_{r_i}^*\}$;
  $\mathcal{X}^{\eta''} = \{x \in \mathcal{X}^\eta | x_i \geq x_{r_i}^*\}$;
  si $\mathcal{X}^{\eta'} \neq \emptyset$ **alors**
    BB($\mathcal{X}^{\eta'}$) ;
  fin
  si $\mathcal{X}^{\eta''} \neq \emptyset$ **alors**
    BB($\mathcal{X}^{\eta''}$) ;
  fin
fin

L'algorithme 2.8 est *anytime*, il peut être interrompu à tout moment de l'exploration. Dans ce cas, une dernière phase d'élicitation est effectuée et une solution x^* telle que $\mathrm{MR}(x^*, PO, W) = 0$ est retournée. On peut alors donner une garantie sur la qualité de cette recommandation :

Proposition 2.20. *Soit η^0 le premier nœud de l'arbre, et soit $\hat{w}$ le poids (caché) représentant les préférences du décideur et $\hat{x}$ la solution OWA-optimale associée. Soit x^* une solution telle que* $\mathrm{MR}(x^*, PO, W) = 0$*, alors :*

$$\mathrm{OWA}_{\hat{w}}(\hat{x}) - \mathrm{OWA}_{\hat{w}}(x^*) \leq BS(\eta^0) - \max_\eta BI(\eta)$$

Démonstration. On sait tout d'abord que $\mathrm{OWA}_{\hat{w}}(\hat{x}) \leq BS(\eta^0)$ par définition de la borne supérieure. Nous allons maintenant montrer par l'absurde que $\mathrm{OWA}_{\hat{w}}(x^*) \geq \max_\eta BI(\eta)$. Supposons que $\mathrm{OWA}_{\hat{w}}(x^*) < \max_\eta BI(\eta)$, il existe alors un nœud η' dans l'arbre tel que $\mathrm{OWA}_{\hat{w}}(x^*) < BI(\eta')$. Notons x' la solution au sous-problème associé à η' telle que $BI(\eta') = \min_{w \in W} \mathrm{OWA}_w(x')$. Par conséquent, il existe un poids $w \in W$ et une solution x' telle que $\mathrm{OWA}_w(x^*) < \mathrm{OWA}_w(x')$ ce qui contredit le fait que $\mathrm{MR}(x^*, PO, W) = 0$ par définition du regret maximum. On obtient alors le résultat de la proposition en sommant les deux inégalités $\mathrm{OWA}_{\hat{w}}(\hat{x}) \leq BS(\eta^0)$ et $-\mathrm{OWA}_{\hat{w}}(x^*) \leq -\max_\eta BI(\eta)$. □

Expérimentations numériques

Afin de tester l'efficacité de l'algorithme 2.8 (*Branch and Bound & Élicitation d'un OWA*), nous l'avons implémenté et testé sur des instances du problème du *sac à dos multi-agents*. Avant de présenter les résultats obtenus, nous commençons par définir ce problème et par donner quelques détails de définition de l'algorithme 2.8 propres à ce problème.

Problème du sac à dos multi-agents On considère ici un problème de sac à dos multi-agents où la décision collective est supervisée par un décideur ayant une certaine exigence d'équité. Ce problème se définit de la manière suivante : on dispose d'un sac à dos de capacité C et de p objets évalués par n agents. On cherche à trouver un sous-ensemble d'objets dont la somme des poids ne dépasse pas la capacité C et qui maximise la satisfaction des agents de manière équitable. Plus formellement, chaque objet k est caractérisé par un poids β_k et un vecteur d'utilités $(u_k^1, \ldots, u_k^n)$ où u_k^i représente l'utilité de l'objet k du point de vue de l'agent i. L'ensemble des solutions réalisables de ce problème est donné par $\mathcal{X} = \{x \in \{0,1\}^p | \sum_{i=1}^p x_i\beta_i \leq C\}$, où x est un vecteur dont chaque composante x_k est une variable de décision binaire telle que $x_k = 1$ si et seulement si l'objet k est choisi dans la solution, et la contrainte $\sum_{i=1}^p x_i\beta_i \leq C$ traduit le fait que le poids total des objets de la solution ne doit pas dépasser la capacité C du sac à dos. Toute solution réalisable x est associée à un vecteur d'utilité $u(x) = (\sum_{k=1}^p u_k^1 x_k, \ldots, \sum_{k=1}^p u_k^n x_k)$ dont chaque composante $u_i(x)$ représente la satisfaction de l'agent i pour la solution x. On suppose ici que les préférences du décideur sont représentées par le paramètre w de l'opérateur OWA, l'objectif est alors de trouver une solution x telle que la valeur $\text{OWA}_w(x)$ est la plus grande possible.

Exemple 2.7. *On considère une instance du problème du sac à dos avec* 3 *agents, une capacité* $C = 48$ *et* 7 *objets dont les poids et utilités sont données par le tableau suivant :*

k	1	2	3	4	5	6	7
β_k	6	5	6	11	13	15	12
u_k^1	5	20	17	16	13	1	4
u_k^2	6	18	3	3	20	12	17
u_k^3	11	0	13	17	4	10	3

La solution $x = (1, 0, 1, 1, 1, 0, 1)$ *représente une solution réalisable associée au vecteur d'utilités* $u(x) = (55, 49, 48)$.

Notons que ce problème est NP-difficile car l'opérateur OWA contient en particulier l'opérateur de moyenne. Or, la recherche d'une solution moyenne-optimale revient à résoudre le problème du sac à dos *standard* qui est bien connu comme étant NP-Difficile Kouvelis et Yu [2013].

Calcul des bornes d'un nœud Afin de calculer la borne supérieure d'un nœud $BS(\eta)$, on utilise un algorithme glouton bien connu pour la résolution efficace du problème du sac à dos relaxé. L'algorithme consiste à classer les objets dans l'ordre décroissant des valeurs $\overline{u}_k/w_k$, où $\overline{u}_k$ désigne la performance moyenne de l'objet k, i.e., $\overline{u}_k = \frac{1}{n}\sum_{i=1}^n u_k^i$. On ajoute alors les objets à la solution dans cet ordre en veillant à respecter les contraintes de $\mathcal{X}^\eta$.

L'ajout d'objets se fait jusqu'à ce que la place restante dans le sac ne suffise pas à l'ajout de l'objet suivant. On tronque alors cet objet de manière à remplir l'espace restant dans le sac à dos, ce qui mène à une solution x_r^* possédant au plus une composante non entière. On pose $BS(\eta) = \frac{1}{n}\sum_{i=1}^{n} u_i(x_r^*)$. Pour construire la solution entière x^* permettant de calculer la borne inférieure de ce même nœud, on prend simplement tous les objets de la solution x_r^* à l'exception de l'objet fractionné, i.e., $x_i^* = x_{ri}^*$ pour tout $i \in \{1, \ldots, p\}$ tel que $x_{ri}^* = 1$, $x_i^* = 0$ sinon. On calcule alors $BI(\eta) = \min_{w \in W} \mathrm{OWA}_w(x^*)$.

Exemple 2.7 (suite). *On donne ici les étapes clés du déroulement de l'algorithme 2.8 sur l'instance du problème de sac à dos donnée dans l'exemple 2.7. On simule ici les réponses du décideur en utilisant le vecteur de pondération $w \in (1/2, 1/3, 1/6)$. On pose une question dès que $|PO| = 2$. L'algorithme commence par calculer les ratios $\overline{u}_k/w_k$ pour tout objet k afin de déterminer l'ordre de sélection des objets dans l'algorithme glouton. On obtient :*

k	1	2	3	4	5	6	7
$\overline{u}_k/w_k$	3.67	7.6	5.5	3.27	2.85	1.53	2
ordre	3	1	2	4	5	7	6

*L'algorithme évalue ensuite le premier nœud η^0 associé au problème initial définit par $\mathcal{X} = \{x \in \{0,1\}^7 | \sum_{i=1}^{7} \beta_i x_i \leq 48\}$. On trouve alors la solution moyenne-optimale relaxée $x_r^* = (1, 1, 1, 1, 1, 0, 7/12)$ qui nous permet d'obtenir la borne $BS(\eta^0) = 60$. La projection de cette solution dans $\mathcal{X}$ est donnée par la solution $x^1 = (1, 1, 1, 1, 1, 0, 0)$ associée au vecteur d'utilités $u(x^1) = (71, 50, 45)$. On obtient alors la borne inférieure $BI(\eta^0) = 45$. Ces calculs permettent d'initialiser $PO \leftarrow \{x^1\}$ et $maxBI \leftarrow BI(\eta^0)$. On peut maintenant créer deux nœuds η^1 et η^2 en partitionnant $\mathcal{X}$ de la manière suivante : η^1 correspond au sous-problème défini par $\mathcal{X}^{\eta^1} = \{x \in \mathcal{X} | x_7 = 0\}$ tandis que η^2 correspond au sous-problème défini par $\mathcal{X}^{\eta^2} = \{x \in \mathcal{X} | x_7 = 1\}$. Les bornes des nœuds η^1 et η^2 sont calculées de la même manière que les bornes de η^0. L'arbre d'exploration est développé jusqu'à ce qu'une solution non dominée par PO (règle **R2**) soit trouvée, cette solution est donnée par $x^2 = (0, 1, 1, 1, 1, 0, 1)$ qui est associée au vecteur d'utilités $u(x^2) = (70, 61, 37)$. Comme $|PO| = 2$, la valeur du MMR est calculée. On trouve $\mathrm{MMR}(PO, W) = 1.5 > 0$, on demande alors au décideur de comparer x^1 et x^2. Le décideur déclare préférer x^1 alors W est mis à jour par $W \leftarrow \{w \in W | \mathrm{OWA}_w(x^1) \geq \mathrm{OWA}_w(x^2)\}$ et x^2 est supprimée de PO. L'algorithme reprend l'exploration jusqu'à l'énumération de la solution $x^3 = (1, 0, 1, 1, 1, 0, 1)$ associée à $u(x^3) = (55, 49, 48)$ et que l'on ajoute à PO. Comme le regret $\mathrm{MMR}(PO, W) = 3$ est positif, on demande au décideur de comparer les deux solutions x^1 et x^3, il déclare préférer x^1. On pose alors $W \leftarrow \{w \in W | \mathrm{OWA}_w(x^1) \geq \mathrm{OWA}_w(x^2)$ & $\mathrm{OWA}_w(x^1) \geq \mathrm{OWA}_w(x^3)\}$. L'algorithme continue ainsi l'exploration jusqu'à ce que tous les nœuds de l'arbre soient explorés ou élagués. Finalement, la solution x^1 est retournée.*

Résultats numériques Nous avons implémenté et testé [2] l'algorithme 2.8 (*Branch & Bound interactif pour l'élicitation d'un OWA*) sur différentes instances aléatoires du problème du sac à dos multi-agents. Pour cela, nous avons généré des instances de sac à dos avec $p \in \{20, 50, 100\}$ objets et $n \in \{3, 5, 10\}$ agents. Pour chaque objet, un poids et des valeurs d'utilités sont générés uniformément dans $[0, 20]$ et la capacité du sac est

2. L'algorithme a été implémenté en Python. Caractéristiques de la machine : Intel(R) Core(TM) i7-4790 CPU avec 15 GB de RAM.

générée aléatoirement autour d'une valeur de référence définie par la somme des poids des p objets. Cela nous permet de générer des instances difficiles à résoudre. Le poids caché représentant les préférences du décideur est généré aléatoirement dans W. Les phases d'élicitation sont effectuées dès lors que $|PO| = 5$.

Nous avons évalué l'efficacité de cette approche en termes de temps de calcul (en secondes), du nombre de questions posées ainsi que de la valeur du *gap* indiquant la distance maximum de la valeur de la recommandation à la valeur optimale (proposition 2.20). Le gap est donné en pourcentage de l'échelle de valeurs. La table 2.9 résume les résultats de ces expérimentations. Pour chaque couple de valeur p, n donné, on effectue une moyenne sur 30 instances avec une limite de temps de 20 minutes. La table 2.10 donne le pourcentage d'instances résolues dans cette limite de temps.

p	n = 3			n = 5			n = 10		
	temps(s)	questions	gap	temps(s)	questions	gap	temps(s)	questions	gap
20	10.45	3.3	5.55	25.8	3.9	6.15	131.06	6.87	9.9
50	563.54	6.17	6	513.91	7.33	5.25	924.44	10.17	8.52
100	832.84	6.47	5.94	984.03	10	6.53	1062.34	14.63	8.64

TABLE 2.9 – Branch and Bound & Élicitation pour le problème du sac à dos multi-agents.

p	n=3	n=5	n=10
20	100%	100%	97%
50	60%	67%	27%
100	37%	23%	17%

TABLE 2.10 – Pourcentage d'instances résolues en moins de 20 minutes.

L'algorithme offre de bonnes performances sur l'ensemble des instances considérées. Les temps de calcul sont, en moyenne, inférieurs à 20 minutes mais on peut voir que le pourcentage d'instances résolues (à l'exact) en moins de 20 minutes baisse considérablement lorsque la taille des instances augmente. Néanmoins, la qualité de la recommandation est relativement bonne car la valeur du gap se situe entre 5 et 6% de l'échelle des valeurs d'OWA possibles dans le cas de 3 et 5 agents pour toutes les tailles d'instances considérées, et autour de 9% dans le cas de 10 agents. Le nombre de questions posées est raisonnable et ne dépasse pas 15 même dans le cas des instances les plus grandes. Finalement, l'énumération de solutions par un algorithme de branch and bound offre une bonne alternative à l'algorithme de ranking. On pourra alors opter pour un tel algorithme lorsque l'on traite un problème pour lequel il n'existe pas de procédure d'énumération ordonnée efficace.

2.4 Conclusion du chapitre

Dans ce chapitre, nous nous sommes intéressés à la conception de différentes méthodes d'élicitation des préférences pour la résolution de problèmes mêlant simultanément trois difficultés : la nature combinatoire de l'espace des solutions, le fait de considérer que plusieurs points de vue/critères/scénarios (potentiellement conflictuels) peuvent exister

concernant l'évaluation d'une solution, et finalement la représentation des préférences du décideur par un modèle non-linéaire qui ne respecte pas le principe d'optimalité de Bellman. Nous avons introduit dans ce chapitre deux méthodes fondées sur une énumération méthodique de l'espace des solutions. Elles énumèrent des solutions réalisables du problème et les filtrent ensuite pour ne garder que les solutions potentiellement optimales. Ces deux méthodes ont pour avantage de générer rapidement des solutions réalisables du problème qui peuvent être recommandées au décideur si l'exécution est interrompue prématurément (avec garantie de performance).

De plus, les algorithmes que nous avons introduits dans ce chapitre sont tous génériques et peuvent être adaptés à la résolution de différents problèmes et à l'élicitation de différents modèles de décision sous certaines conditions. L'algorithme de ranking nécessite : tout d'abord que le problème traité possède un algorithme d'énumération ordonnée efficace, et ensuite que l'opérateur d'agrégation utilisé possède une borne linéaire proche afin de définir une règle d'énumération et un critère d'arrêt efficaces. Lorsqu'aucune méthode d'énumération ordonnée efficace n'est connue pour le problème traité, une alternative à l'algorithme de ranking peut être l'algorithme de branch and bound interactif. Cet algorithme peut être utilisé pour la résolution de tout problème défini par des variables entières et peut être utilisé pour l'élicitation de tout opérateur possédant une borne linéaire efficace. Finalement, ces deux algorithmes nécessitent que l'opérateur choisi soit linéaire en son paramètre, c'est le cas de la fonction OWA. Même si WOWA n'est initialement pas linéaire en son paramètre (la fonction φ), nous avons pu adapter l'utilisation de ces méthodes à l'élicitation de φ en introduisant une nouvelle formulation de ce paramètre utilisant une nouvelle famille de fonctions splines que nous définissons comme les fonctions *C-Splines*. De cette manière, on obtient une formulation de WOWA qui est linéaire en son (nouveau) paramètre.

Une extension intéressante de ces deux approches serait de les appliquer à l'élicitation de la capacité d'une intégrale de Choquet. La difficulté de cette extension réside dans le fait que la seule borne linéaire (efficace) connue de l'intégrale de Choquet dépend de sa capacité, or celle-ci ne peut être fixée lorsque les préférences ne sont pas précisément connues. L'extension de la méthode à cet opérateur n'est donc pas évidente.

Les algorithmes introduits dans ce chapitre sont très efficaces en pratique puisqu'ils permettent de déterminer des solutions optimales ou presque optimales avec une connaissance imprécise des préférences du décideur. Néanmoins, il peut arriver que certaines instances nécessitent qu'un très grand nombre de solutions soient énumérées avant que le critère d'arrêt ne se déclenche. L'efficacité de ce critère dépend alors essentiellement de la structure de l'instance traitée ainsi que de l'efficacité de la borne choisie. Alors même que la procédure d'élicitation est fondée sur des calculs de regrets, ceux-ci ne peuvent être utilisés pour définir un critère d'arrêt plus efficace puisqu'ils sont calculés sur un sous-ensemble de solutions énumérées. Or, du fait de certaines propriétés intéressantes concernant les regrets et la stratégie CSS (cf. sous-section 2.2.3), il serait intéressant de pouvoir calculer les regrets sur la totalité de l'espace des solutions afin de pouvoir définir un critère d'arrêt centré sur la procédure d'élicitation et sur l'évolution des regrets. Nous présentons dans la suite une méthode fondée sur une minimisation directe des regrets nous permettant de bénéficier de tous les avantages de la stratégie CSS.

Chapitre 3

Élicitation des préférences fondée sur la minimisation directe des regrets

Résumé du chapitre

Ce chapitre est consacré à la conception d'algorithmes d'élicitation incrémentale fondés sur la minimisation directe des regrets. On se place ici dans un cadre similaire à celui du chapitre précédent où la prise de décision collective ou dans l'incertain se fait sur domaine combinatoire. On suppose également que les préférences du décideur sont représentables par une fonction d'agrégation paramétrée linéaire en son paramètre. On se place ici dans le cas particulier où les préférences du décideur sont représentées par une somme pondérée ordonnée (OWA pour Ordered Weighted Averages). On élicitera alors les paramètres d'un OWA modélisant les préférences du décideur afin de lui proposer la meilleure recommandation possible. Pour cela, on introduit une nouvelle méthode de calcul des regrets par programmation linéaire rendant possible leur détermination sur domaine combinatoire. On proposera ensuite une adaptation de cet algorithme à la résolution de problèmes multi-agents impliquant un très grand nombre d'agents. Ce chapitre est essentiellement issu de travaux publiés dans [Bourdache et Perny, 2019] pour l'élicitation des paramètres d'un OWA pour la résolution du problème du sac à dos multi-agents.

3.1 Introduction

On s'intéresse dans ce chapitre à la même problématique que le chapitre précédent, i.e., la conception d'algorithmes d'élicitation incrémentale des préférences traitant plusieurs difficultés simultanément : 1. la nature combinatoire de l'espace des solutions réalisables, 2. l'incertitude concernant les préférences du décideur impliquant que l'optimisation d'un opérateur d'agrégation relève, au mieux, de la programmation quadratique, et 3. l'utilisation d'opérateurs non-linéaires, qui ne respectent pas le principe d'optimalité de Bellman, pour modéliser les préférences du décideur. Nous avons pu voir que ces trois difficultés constituaient un challenge pour la conception de méthodes de résolution, notamment pour le calcul des solutions potentiellement optimales ainsi que pour le calcul des différents regrets qui nécessitent alors, d'une part, de définir des programmes linéaires contenant un nombre exponentiel de contraintes, et d'autre part, d'effecteur des opérations de comparaisons de regrets en nombre exponentiel (cf. sous-section 2.2.4).

Nous introduisons dans ce chapitre une méthode qui aborde le problème de résolution et d'élicitation sous un angle différent (par rapport au chapitre précédent), on souhaite toujours combiner élicitation des préférences et optimisation combinatoire, mais l'exploration de l'espace des solutions se fait ici de manière différente puisqu'elle est cette fois guidée par une exploration de l'espace des paramètres. On propose une approche d'énumération de solutions exploitant les points extrêmes du polyèdre représentant l'ensemble d'incertitude. Le fait de travailler sur cet ensemble de paramètres particulier se justifie par la nature convexe de l'ensemble d'incertitude et par la linéarité des fonctions objectifs définies lors du calcul des différents regrets. En exploitant ces deux propriétés, nous définissons une méthode permettant l'optimisation directe du minimax regret sur la totalité de l'espace des solutions. À notre connaissance, la seule approche permettant de calculer les valeurs de regrets sur domaine combinatoire est une méthode de génération de contraintes présentée par Boutilier *et al.* [2006a] pour l'élicitation des *valeurs d'utilités* dans le cadre de la résolution d'un problème de satisfaction de contraintes. Dans un cadre similaire au nôtre, où l'incertitude porte sur les *valeurs des paramètres* du modèle de décision, notre méthode est la seule méthode, à notre connaissance, permettant d'effectuer un calcul de regret sur domaine combinatoire.

Contexte et notations. On utilisera ici les mêmes notations que le chapitre précédent : $\mathcal{X}$ désigne l'ensemble des solutions réalisables défini par un ensemble de contraintes linéaires. Chaque solution de cet ensemble est évaluée selon n fonctions objectifs : point de vue de n agents ou évaluation selon n scénarios. L'image de $\mathcal{X}$ dans l'espace des critères est donnée par $\mathcal{Y} = \{u(x) \in \mathbb{R}^n, \forall x \in \mathcal{X}\}$ où $u(x)$ est le vecteur de performance associé à la solution x et dont la $i^{\text{ème}}$ composante représente la performance de x selon la fonction objectif i. On se place dans un cadre où aucune information concernant les préférences du décideur n'est connue à l'exception d'une exigence plus ou moins forte d'équité ou de robustesse. On modélise les préférences du décideur par un OWA à poids décroissants, on définit alors l'ensemble d'incertitude par l'ensemble de vecteurs de pondération $W = \{w \in \mathbb{R}^n_+ | \max_{i=1}^n w_i = 1, w_1 \geq \cdots \geq w_n\}$ (la contrainte $\max_{i=1}^n w_i = 1$ remplace la contrainte de normalisation habituelle, elle est non-restrictive mais sera utile par la suite). Les questions que l'on pose au décideur pour éliciter w ou φ consistent à lui demander de comparer des paires de solutions. Ces comparaisons permettent de réduire l'incerti-

tude sur les préférences par le biais de contraintes linéaires réduisant systématiquement et irréversiblement l'espace des paramètres. Les algorithmes présentés sont décrits dans le cadre d'une maximisation de l'OWA.

Organisation du chapitre. Ce chapitre est organisé de la manière suivante : la section 3.2 est consacrée à la présentation de notre algorithme d'élicitation fondé sur une nouvelle méthode de calcul des regrets sur domaine combinatoire. On introduit, dans un premier temps, les différents programmes linéaires permettant de calculer les regrets par paires (PMR), les regrets maximum (MR) ainsi que le minimax regret (MMR). On présente ensuite deux stratégies d'élicitation ainsi que l'algorithme alternant questions préférentielles et mises à jour de l'ensemble d'incertitude jusqu'à annulation des regrets. La section 3.3 donne ensuite une extension de la méthode permettant de résoudre des problèmes multi-agents impliquant un très grand nombre d'agents. Finalement, on conclut le chapitre par la section 3.4.

3.2 Une approche exploitant les sommets de l'espace des paramètres

Lorsque les préférences du décideur sont modélisées par une fonction d'agrégation paramétrée linéaire en son paramètre, l'espace des paramètres possibles est défini par un polyèdre convexe. En effet, prenons l'exemple de l'opérateur OWA auquel on s'intéresse dans ce chapitre. Lorsque l'on modélise une préférence pour les solutions aux vecteurs de performances (plus ou moins) équilibrés, l'espace des paramètres est défini par tous les vecteurs de pondérations du simplexe dont les composantes sont triées dans l'ordre décroissant. On a donc $W = \{w \in \mathbb{R}_+^n | \max_{i=1}^n w_i = 1, w_1 \geq \cdots \geq w_n\}$. De plus, comme nous avons pu le préciser précédemment, toute contrainte induite par une information préférentielles du type *"je préfère la solution x à la solution y"* est linéaire, son ajout conserve donc la convexité de W. On surmonte ici les difficultés liées aux calculs des différents regrets sur domaine combinatoire (sous-section 2.2.4) grâce à cette convexité et à la linéarité d'OWA en w. Ces deux propriétés nous permettent de montrer que le calcul des regrets MR et MMR (qui effectue habituellement un nombre exponentiel de comparaisons par paires) peut être réduit à la comparaison d'un ensemble de solutions bien précis (de taille polynomiale) obtenu à l'aide des sommets de l'espace des paramètres.

3.2.1 Programmation linéaire pour le calcul des regrets sur domaine combinatoire

Afin de déterminer une question ou une recommandation en suivant la stratégie *CSS*, un grand nombre de calculs de regrets par paires est effectué. On rappelle que le regret d'une paire de solutions $x, y \in \mathcal{X}$ prend la forme suivante :

$$\text{PMR}(x, y, W) = \max_{w \in W} \{\text{OWA}_w(y) - \text{OWA}_w(x)\} \tag{3.1}$$

La méthode que nous proposons est fondée sur l'observation suivante : le calcul de $\text{PMR}(x, y, W)$ pour toute paire de solutions $x, y \in \mathcal{X}$ revient à optimiser une fonction

linéaire sur le polyèdre convexe W, la valeur optimale est alors vérifiée par au moins un point extrême du polyèdre W. Si on note W^{ext} l'ensemble des points extrêmes de W, l'équation 3.1 est alors équivalente à :

$$\mathrm{PMR}(x, y, W) = \max_{w \in W^{ext}} \{\mathrm{OWA}_w(y) - \mathrm{OWA}_w(x)\} \tag{3.2}$$

On en déduit alors la proposition suivante :

Proposition 3.1. *Soit $x \in \mathcal{X}$ une solution réalisable et soit W^{ext} l'ensemble des points extrêmes du polyèdre des paramètres possibles W. Le regret maximum de x est donné par :*

$$\mathrm{MR}(x, \mathcal{X}, W) = \max_{w \in W^{ext}} \{\mathrm{OWA}_w(x_w^*) - \mathrm{OWA}_w(x)\}$$

où x_w^* est la solution OWA-optimale pour le point extrême w, i.e., $x_w^* \in arg \max_{y \in \mathcal{X}} \mathrm{OWA}_w(y)$.

Avant de démontrer la validité de cette proposition, notons que la solution x_w^* peut être obtenue par programmation linéaire puisqu'ici le poids w est fixé. On utilisera alors une linéarisation d'OWA telle que celle proposée par Ogryczak et Śliwiński [2003].

Démonstration. On rappelle tout d'abord que le regret maximum est défini par :

$$\mathrm{MR}(x, \mathcal{X}, W) = \max_{y \in \mathcal{X}} \mathrm{PMR}(x, y, W) = \max_{y \in \mathcal{X}} \max_{w \in W} \{\mathrm{OWA}_w(y) - \mathrm{OWA}_w(x)\}$$

or, par l'équation (3.2) on a :

$$\begin{aligned} \mathrm{MR}(x, \mathcal{X}, W) &= \max_{y \in \mathcal{X}} \max_{w \in W^{ext}} \left\{\mathrm{OWA}_w(y) - \mathrm{OWA}_w(x)\right\} \\ &= \max_{w \in W^{ext}} \max_{y \in \mathcal{X}} \left\{\mathrm{OWA}_w(y) - \mathrm{OWA}_w(x)\right\} \\ &= \max_{w \in W^{ext}} \max_{y \in \mathcal{X}} \left\{\mathrm{OWA}_w(y)\right\} - \mathrm{OWA}_w(x) \end{aligned}$$

la dernière égalité nous mène directement à notre résultat puisque, pour un point extrême w fixé, on a $\max_{y \in \mathcal{X}} \{\mathrm{OWA}_w(y)\} = \mathrm{OWA}_w(x_w^*)$, ce qui nous donne :

$$\mathrm{MR}(x, \mathcal{X}, W) = \max_{w \in W^{ext}} \mathrm{OWA}_w(x_w^*) - \mathrm{OWA}_w(x)$$

□

De ce fait, le regret maximum d'une solution correspond à une solution optimisant un point extrême de W, son calcul ne se définit plus par l'optimisation d'une fonction quadratique mais par la comparaison d'un nombre polynomial (en $|W^{ext}|$) de calculs de regrets par paires. Pour pouvoir effectuer ce calcul, on doit d'abord résoudre $|W^{ext}|$ programmes linéaires pour obtenir la solution OWA_w-optimale pour chaque $w \in W^{ext}$, puis comparer les différences d'OWA $(\mathrm{OWA}_w(x_w^*) - \mathrm{OWA}_w(x))$ pour tout $w \in W^{ext}$.

En utilisant la proposition 3.1 on établit la proposition suivante pour le calcul du minimax regret :

Proposition 3.2. *Soit W^{ext} l'ensemble des m points extrêmes de W. La valeur du minimax regret $\mathrm{MMR}(\mathcal{X}, W)$ est obtenue par la résolution de $m + 1$ programmes linéaires à variables mixtes.*

Démonstration. Par définition du minimax regret $\text{MMR}(\mathcal{X}, W) = \min_{x \in \mathcal{X}} \text{MR}(x, \mathcal{X}, W)$. Par la proposition 3.1 on obtient :

$$\text{MMR}(\mathcal{X}, W) = \min_{x \in \mathcal{X}} \max_{w \in W^{ext}} \{\text{OWA}_w(x^*_w) - \text{OWA}_w(x)\}$$

On utilise alors un résultat bien connu de la linéarisation de la fonction min max (où le max porte sur un ensemble discret d'éléments) pour obtenir :

$$\text{MMR}(x, \mathcal{X}, W) = \begin{cases} \min t \\ t \geq \text{OWA}_w(x^*_w) - \text{OWA}_w(x) \qquad \forall w \in W^{ext} \\ x \in \mathcal{X} \end{cases} \tag{3.3}$$

où $x^*_w \in arg\max_{y \in \mathcal{X}} \text{OWA}_w(y)$ pour le point extrême w. Le calcul des m solutions x^*_w nécessite la résolution de m programmes linéaires. Un dernier programme linéaire, défini par le programme 3.3, est ensuite résolu afin d'obtenir la valeur du minimax regret. Ainsi, on obtient $\text{MMR}(\mathcal{X}, W)$ par la résolution de $m + 1$ programmes linéaires. □

Grâce à ces deux propositions, nous avons réduit la charge de calcul nécessaire à la détermination des valeurs MR et MMR en passant de formulations quadratiques (et nécessitant un nombre exponentiel, en la taille du problème, de comparaisons par paires) à des formulations linéaires. On peut alors utiliser cette méthode de calcul pour définir un algorithme d'élicitation des préférences pour la résolution de tout problème possédant une formulation linéaire efficace.

3.2.2 Algorithme d'exploration et d'élicitation

L'idée de l'algorithme est simple, elle consiste à alterner calcul de regrets et questions préférentielles jusqu'à ce que la valeur du minimax regret soit nulle (pour une résolution exacte) ou en dessous d'un seuil δ prédéfini (pour la recherche d'une solution δ-optimale). L'approche est détaillée dans l'algorithme 3.1.

Algorithme 3.1 :

Entrées : $\mathcal{X}$: ensemble des solutions réalisables ; δ : seuil de tolérance
Sorties : x^* : recommandation δ-optimale

1 $W \leftarrow \{w \in \mathbb{R}^n_+ : w_1 = 1, w_1 \geq \cdots \geq w_n\}$;
2 $W^{ext} \leftarrow \{w^1, \ldots, w^n\}$ avec pour tout i, $w^i_j = 1$ si $j \leq i$ et $w^i_j = 0$ sinon ;
3 **répéter**
4 | Déterminer x^*_w pour tout $w \in W^{ext}$;
5 | Poser une question au décideur ;
6 | Restreindre W en fonction de la réponse ;
7 | Mettre à jour W^{ext} ;
8 **jusqu'à** $\text{MMR}(\mathcal{X}, W) \leq \delta$;
9 **retourner** $x^* \in \arg\min_{x \in \mathcal{X}} \text{MR}(x, \mathcal{X}, W)$

Notons qu'avec la définition initiale de W (ligne 1), l'ensemble W^{ext} contient exactement $m = n$ points extrêmes : les i premières composantes du i^e vecteur de W^{ext} valent 1 et les $n - i$ composantes restantes sont nulles. Lorsque W évolue, W^{ext} peut être calculé

par une méthode d'énumération des sommets comme, par exemple, la méthode Primale-Duale d'énumération des sommets et facettes d'un polyèdre [Bremner *et al.*, 1998]. Afin d'obtenir des calculs plus rapides, on calcule uniquement les points devenus extrêmes par l'ajout de la dernière contrainte : si on note $\sum_{i=1}^{n} w_i d_i \geq 0$ cette contrainte, les points extrêmes engendrés par $\sum_{i=1}^{n} w_i d_i \geq 0$ sont obtenus en appliquant l'algorithme d'énumération des points extrêmes au polyèdre $W \cap \{w \in \mathbb{R}_+^n | \sum_{i=1}^{n} w_i d_i = 0\}$. Pour obtenir tous les points de W^{ext}, il suffira alors d'ajouter les points extrêmes de l'étape précédente qui vérifient la nouvelle contrainte, i.e. les précédents points extrêmes w tel que $\sum_{i=1}^{n} w_i d_i \geq 0$.

Cette méthode étant fondée sur une exploration de l'espace des paramètres guidée par le calcul des différents regrets, son efficacité dépend entièrement de l'efficacité de la procédure d'élicitation utilisée. Les questions que l'on pose au décideur devront être suffisamment informatives pour réduire la valeur du MMR jusqu'à ce qu'il s'annule (ou atteigne une valeur $\leq \delta$) avec le moins de questions possibles.

Stratégies du choix de la question

On propose ici deux stratégies du choix de la question suivante : la première a pour objectif de réduire la valeur du MMR à l'issue de chaque question, tandis que la seconde stratégie est conçue de manière à contrôler l'évolution de l'espace des paramètres afin de borner le nombre de questions posées. Avec cette seconde stratégie, la réduction de la valeur du MMR n'est qu'une conséquence de la réduction de l'espace des paramètres et peut être moins fréquente que pour la première stratégie.

Stratégie S1 : La première stratégie proposée est la stratégie *CSS*, elle constitue un choix naturel de questions à poser lorsque l'on a pour objectif de réduire rapidement les valeurs de regrets. Pour rappel, cette stratégie consiste à demander au décideur de comparer la meilleure recommandation courante au sens du MMR, i.e., $x^* \in \arg\min_{x \in \mathcal{X}} \mathrm{MR}(x, \mathcal{X}, W)$ obtenue en résolvant le programme linéaire donné par le système (3.3), à son pire adversaire au sens du regret maximum, i.e., $y^* \in \arg\max_{y \in \mathcal{X}} \mathrm{PMR}(x^*, y, W)$. En pratique, l'utilisation de cette stratégie nous permet de réduire efficacement l'espace des paramètres et de déterminer une solution nécessairement optimale (ou δ-optimale) rapidement avec une connaissance du paramètre w très imprécise. De plus, utiliser cette stratégie du choix de la question dans l'algorithme 3.1 a pour avantage de ne pas nécessiter de calculs supplémentaires : les regrets sont calculés à la fois pour déterminer la question à poser et pour vérifier la condition d'arrêt.

Exemple 3.1. *Supposons que l'on applique l'algorithme 3.1 avec la stratégie S1 à l'instance de sac à dos de l'exemple 2.7. Pour rappel, cet exemple considère une instance avec* 3 *agents, une capacité* $C = 48$ *et* 7 *objets dont les poids et utilités sont données par le tableau suivant :*

k	1	2	3	4	5	6	7
β_k	6	5	6	11	13	15	12
u_k^1	5	20	17	16	13	1	4
u_k^2	6	18	3	3	20	12	17
u_k^3	11	0	13	17	4	10	3

Dans cet exemple, on simule les réponses du décideur par le vecteur de pondération

$(1, 2/3, 1/3)$ *(représenté par le point bleu sur les polyèdres des 3 prochaines figures). Initialement, le polyèdre des paramètres possibles* W *est donné par la figure 3.1 :*

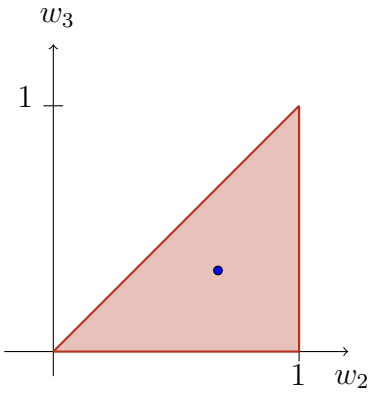


FIGURE 3.1 – Ensemble d'incertitude initial.

La valeur du minimax regret initial est strictement positive : $\mathrm{MMR}(\mathcal{X}, W) = 3$. *On pose alors une question au décideur en utilisant la stratégie* $S1$. *Pour cela, on lui demande de comparer la solution ayant le meilleur max regret* $x^* = (1, 1, 1, 1, 1, 0, 0)$, *associée au vecteur d'utilités* $u(x^*) = (71, 50, 45)$, *à la solution qui constitue son meilleur adversaire* $y^* = (1, 0, 1, 1, 1, 0, 1) \in \arg\max_{y \in \mathcal{X}} \mathrm{PMR}(x^*, y, \mathcal{X})$ *associée au vecteur* $u(y^*) = (55, 49, 48)$. *Le décideur déclare préférer* x^* *alors on ajoute la contrainte* $\mathrm{OWA}_w(x^*) \geq \mathrm{OWA}_w(y^*)$ *à la définition de* W. *L'ajout de cette contrainte mène à un espace des paramètres plus petit illustré par la figure 3.2 où la zone hachurée correspond au sous-ensemble de paramètres que l'on exclut de* W *:*

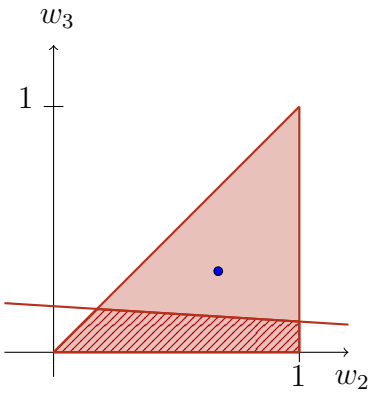


FIGURE 3.2 – Ensemble d'incertitude après la première question (choisie avec S1).

La valeur du MMR est de nouveau calculée et on trouve $\mathrm{MMR}(\mathcal{X}, W) = 2 > 0$, *le décideur compare alors* x^* *à son nouvel adversaire donné par la solution* $y^* = (0, 1, 1, 1, 1, 0, 1)$ *associée au vecteur de performance* $u(y^*) = (70, 61, 37)$. *Le décideur préfère toujours la solution* x^*, *on ajoute alors la contrainte* $\mathrm{OWA}_w(x^*) \geq \mathrm{OWA}_w(y^*)$ *à la définition de* W *pour obtenir le polyèdre de la figure 3.3 où la zone hachurée correspond toujours au sous-ensemble de paramètres que l'on exclut de* W *:*

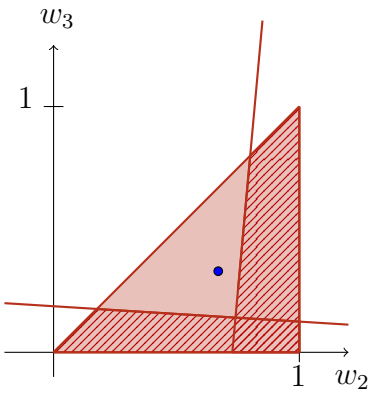


FIGURE 3.3 – Ensemble d'incertitude après la seconde question (choisie avec S1).

Cette fois le MMR vaut 0 *et l'algorithme s'arrête en retournant la solution* x^* *qui est nécessairement optimale.*

Proposition 3.3. *L'algorithme 3.1 appliqué avec la stratégie du choix de la question S1 termine et retourne une solution* δ*-optimale.*

Avant de démontrer cette proposition, notons que cette proposition signifie qu'une solution nécessairement optimale peut être obtenue en fixant la valeur de δ à 0.

Démonstration. Notons tout d'abord que de manière évidente l'algorithme posera dans le pire cas un nombre exponentiel de questions : à chaque réponse du décideur "$x \succsim y$", l'algorithme écarte de W tous les vecteurs de pondérations pour lesquels la solution y est meilleure que la solution x. Au bout d'au plus $|\mathcal{X}|(|\mathcal{X}|-1)$ questions, W ne contiendra plus que les vecteurs de pondération pour lesquels la solution préférée du décideur est optimale. Ce qui montre la terminaison de l'algorithme.

Concernant la qualité de la recommandation, la solution retournée par l'algorithme en utilisant la stratégie $S1$ est δ-optimale du fait de la proposition 2.6. □

La stratégie CSS est très efficace en pratique. Néanmoins, aucun résultat sur le nombre de questions qu'elle pose n'est connu. Lorsque l'on souhaite pouvoir définir une borne sur le nombre de questions posées, on propose l'alternative suivante :

Stratégie S2 : La seconde stratégie que l'on propose est fondée sur une division méthodique de l'espace des paramètres. À une étape donnée de l'algorithme, chaque composante $w_i, i \in N$, se situe dans un intervalle de valeurs possibles, notons le $I_i = [l_i, u_i]$ (initialement $[0,1]$). La stratégie que nous proposons consiste à poser des questions visant à diviser par deux le plus grand intervalle parmi $I_i, i \in N$. Pour cela, une fois que le plus grand intervalle I_i a été déterminé, on construit deux vecteurs de performances fictifs x et y comme suit :

$$x = (0, \underbrace{\frac{\lambda c}{1+\lambda}, ..., \frac{\lambda c}{1+\lambda}}_{\times(i-2)}, \underbrace{c, ..., c}_{\times(n-i+1)})$$

$$y = (\underbrace{\frac{\lambda c}{1+\lambda}, ..., \frac{\lambda c}{1+\lambda}}_{\times i}, \underbrace{c, ..., c}_{\times(n-i)})$$

où $\lambda = (l_i + u_i)/2$ représente le milieu de l'intervalle I_i et la constante c est définie dans $(0, M]$ où M est une constante. Le résultat de la comparaison de ces deux vecteurs nous mène à l'élimination de la moitié de l'intervalle I_i. En effet, supposons que le décideur déclare préférer x à y, on en déduit la nouvelle contrainte $\text{OWA}_w(x) \geq \text{OWA}_w(y)$ sur le paramètre w. On a alors :

$$\begin{array}{rcl}
\text{OWA}_w(x) \geq \text{OWA}_w(y) & \Rightarrow & \frac{\lambda c}{1+\lambda}(\sum_{j=2}^{i-1} w_j) + c\sum_{j=i}^{n} w_j \geq \frac{\lambda c}{1+\lambda}(\sum_{j=1}^{i} w_j) + c\sum_{j=i+1}^{n} w_j \\
& \Rightarrow & \frac{\lambda c}{1+\lambda}(\sum_{j=2}^{i-1} w_j - \sum_{j=1}^{i} w_j) + c(\sum_{j=i}^{n} w_j - \sum_{j=i+1}^{n} w_j) \geq 0 \\
& \Rightarrow & -\frac{\lambda c}{1+\lambda}(w_1 + w_i) + cw_i \geq 0 \\
& \Rightarrow & -\frac{\lambda}{1+\lambda} w_1 + \frac{1}{1+\lambda} w_i \geq 0 \\
& \Rightarrow & w_i \geq \lambda w_1
\end{array}$$

Sous la contrainte (non restrictive) $w_1 = 1$ on obtient $w_i \geq \lambda = (l_i + u_i)/2$. La nouvelle borne inférieure l_i est donc mise à jour par $l_i = \lambda$ et l'intervalle I_i est réduit de moitié. De la même manière, on obtient $w_i \leq \lambda = (l_i + u_i)/2$ si le décideur déclare préférer y à x.

Notons que la réduction progressive des intervalles I_i peut également mener à la réduction des valeurs de regrets PMR, MR et MMR. Ainsi, des questions seront posées au décideur jusqu'à ce que $\text{MMR}(\mathcal{X}, W) \leq \delta$.

Exemple 3.1 (suite). *Appliquons maintenant l'algorithme 3.1 à l'exemple 2.7 en utilisant cette fois la stratégie S2. L'ensemble des paramètres possible est initialement le même que celui de la figure 3.1 et le MMR vaut initialement* $3 > 0$. *Deux vecteurs x et y sont alors construits comme décrit plus haut : initialement,* $w_1 = 1$, $I_2 = I_3 = [0, 1]$ *alors on commence par réduire (arbitrairement) I_2 en demandant au décideur de comparer* $x = (0, c, c)$ *et* $y = (c/3, c/3, c)$ *pour une valeur quelconque de* $c \in (0, M]$. *Le décideur déclare préférer la solution x, alors on ajoute la contrainte* $w_2 \geq 1/2$ *à la définition de W. On obtient alors le polyèdre de la figure 3.4 où la zone hachurée correspond au sous-ensemble de paramètres que l'on exclut de W :*

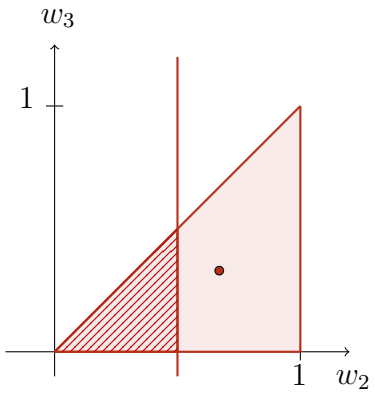


FIGURE 3.4 – Ensemble d'incertitude après la première question (choisie avec S2).

La valeur du MMR est toujours $3 > 0$ *et les intervalles sont maintenant* $I_2 = [1/2, 1]$ *et* $I_3 = [0, 1]$, *alors on demande au décideur de comparer* $x = (0, c/3, c)$ *et* $y = (c/3, c/3, c/3)$ *afin de réduire I_3 qui est l'intervalle le plus large. Le décideur préfère la solution x, alors la contrainte* $w_3 \geq 1/2$ *est ajoutée à la définition de W (figure 3.5).*

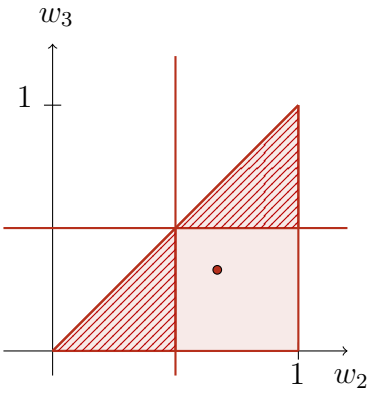


FIGURE 3.5 – Ensemble d'incertitude après la deuxième question (choisie avec S2).

La valeur du MMR n'a toujours pas changé et $I_2 = I_3 = [1/2, 1]$. *Deux autres questions (figures 3.6(a) et 3.6(b)) sont posées avant d'obtenir* $\text{MMR} = 0$. *L'algorithme s'arrête et retourne la solution* $x^* = (1, 1, 1, 1, 1, 0, 0) \in \arg\min_{x \in \mathcal{X}} \text{MR}(x, W, \mathcal{X})$ *associée au vecteur d'utilités* $u(x^*) = (71, 50, 45)$.

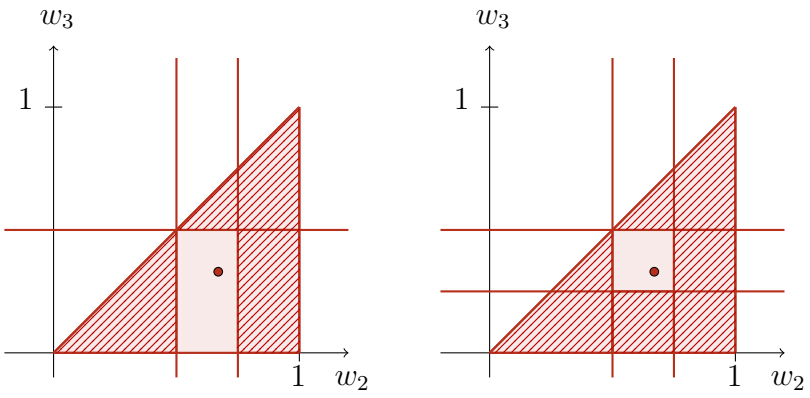


FIGURE 3.6 – Ensemble d'incertitude après la troisième (à gauche) et la quatrième (à droite) question (choisie avec S2).

On voit bien à travers cet exemple que la stratégie S2 peut poser plus de questions que la stratégie S1 car la réduction du MMR est moins rapide. Néanmoins, cette stratégie nous permet de borner le nombre de questions nécessaires pour déclencher la condition d'arrêt de l'algorithme 3.1. Si les évaluations individuelles des critères sont bornées par un nombre M, alors la proposition suivante est vraie :

Proposition 3.4. *L'algorithme 3.1 appliqué avec la stratégie S2 et un seuil de tolérance* $\delta = \varepsilon npM, \forall \varepsilon \in (0, 1]$, *termine et retourne une solution* δ*-optimale en moins de* $n\lceil \log_2(1/\varepsilon) \rceil$ *questions.*

Démonstration. Nous allons d'abord prouver que si $u_i - l_i \leq \varepsilon, \forall i \in N$, alors $\text{MMR}(\mathcal{X}, W) \leq \delta$. Soit w' un vecteur quelconque de l'espace des paramètres W à une étape donnée de l'algorithme et soit x' la solution OWA-optimale associée. Pour toute solution $y \in \mathcal{X}$ et pour tout vecteur poids $w \in W$ on a $\text{OWA}_w(y) - \text{OWA}_w(x') \leq \text{OWA}_w(y) - \text{OWA}_w(x') +$

$\text{OWA}_{w'}(x') - \text{OWA}_{w'}(y)$. Cette inégalité est vraie car $\text{OWA}_{w'}(x') - \text{OWA}_{w'}(y) \geq 0$ pour toute solution y du fait que x' soit optimale pour le poids w'. On a alors :

$$\begin{aligned}
\text{PMR}(x', y, W) &= \max_{w \in W} \Big\{ \text{OWA}_w(y) - \text{OWA}_w(x') \Big\} \\
&\leq \max_{w \in W} \Big\{ \text{OWA}_w(y) - \text{OWA}_w(x') \Big\} + \text{OWA}_{w'}(x') - \text{OWA}_{w'}(y) \\
&\leq \max_{w \in W} \Big\{ (\text{OWA}_w(y) - \text{OWA}_{w'}(y)) - (\text{OWA}_w(x') - \text{OWA}_{w'}(x')) \Big\} \\
&\leq \max_{w \in W} \Big\{ \textstyle\sum_{i=1}^{n} (w_i - w'_i) u_{(i)}(y) - \sum_{i=1}^{n} (w_i - w'_i) u_{(i)}(x') \Big\} \\
&\leq \max_{w \in W} \Big\{ \textstyle\sum_{i=1}^{n} \underbrace{(w_i - w'_i)}_{\leq \varepsilon} (u_{(i)}(y) - u_{(i)})(x') \Big\} \\
&\leq \varepsilon \textstyle\sum_{i: w_i > w'_i} u_{(i)}(y) + \varepsilon \sum_{i: w_i < w'_i} u_{(i)}(x') \leq \varepsilon npM = \delta
\end{aligned}$$

Cette inégalité étant vraie pour toute solution $y \in \mathcal{X}$, on a alors $\text{MR}(x', \mathcal{X}, W) \leq \delta$. De plus, par définition du minimax regret $\text{MMR}(\mathcal{W}, W) \leq \text{MR}(x', \mathcal{X}, W)$. On a alors $\text{MMR}(\mathcal{X}, W) \leq \delta$ et par la proposition 2.4 on obtient $\text{OWA}_{\hat{w}}(\hat{x}) - \text{OWA}_{\hat{w}}(x^*) \leq \text{MMR}(\mathcal{X}, W) \leq \delta$ où x^* est la solution retournée par l'algorithme. Donc x^* est (au pire) δ-optimale.

Montrons maintenant que ce résultat est obtenu en moins de $n\lceil \log_2(1/\varepsilon) \rceil$ questions. Du fait de l'initialisation de l'espace des paramètres (ligne 1 de l'algorithme) on a initialement $I_i = [0, 1]$ pour tout $i \in \{2, \ldots, n\}$. Afin de réduire un intervalle I_i jusqu'à obtenir $u_i - l_i \leq \varepsilon$, on a besoin d'au plus $\lceil \log_2(1/\varepsilon) \rceil$ questions. La réduction de tous les intervalles $I_i, i \in \{2, \ldots, n\}$ nécessite alors au total $(n-1)\lceil \log_2(1/\varepsilon) \rceil$ questions. Ce qui prouve également la terminaison de l'algorithme. □

Notons que la borne sur le nombre de questions est une borne pire cas. En pratique, il arrive très souvent que la condition $\text{MMR}(\mathcal{X}, W) \leq \delta$ soit vérifiée bien avant que l'amplitude des intervalles soit inférieure à ε :

Exemple 3.1 (suite). *Nous avons vu dans l'exemple 3.1 que seulement* 4 *questions étaient nécessaires à la détermination d'une solution nécessairement optimale lorsque l'on applique l'algorithme 3.1 avec la stratégie S2. Alors qu'en prenant* $\varepsilon = 0.05$*, la proposition 3.4 nous donne la borne* $\delta = \varepsilon npM = 0.05 \times 3 \times 7 \times 20 = 21$ *avec au plus* $3\lceil \log_2(1/0.05) \rceil = 15$ *questions.*

3.2.3 Expérimentations numériques

L'algorithme 3.1 a été implémenté et testé sur le problème d'affectation robuste et multi-agents ainsi que sur le problème du sac à dos multi-agents (définis dans la section précédente). Les différents programmes linéaires sont résolus en utilisant la librairie gurobipy de Python. L'ensemble des points extrêmes d'un polyèdre donné est calculé en utilisant la librairie cdd de Python qui implémente la *double description method* introduite par Fukuda et Prodon [1996].

1. Problème du sac à dos multi-agents

Les premiers tests [1] que nous présentons ici visent à montrer l'efficacité de l'algorithme 3.1 avec la stratégie S1. Nous avons appliqué l'algorithme à des instances du problème

1. Implémentation en Python. Les tests sont effectués sur une machine Intel(R) Core(TM) i7-4790 CPU avec 15 GB de RAM

du sac à dos de différentes tailles : $p \in \{20, 50, 100\}$ objets évalués par $n \in \{3, 5, 10\}$ agents. Ces instances sont les mêmes que les instances utilisées pour les expérimentations numériques de l'algorithme de Branch and Bound interactif. Pour ces expérimentations on fixe $\delta = 0$ et on évalue l'efficacité en termes de temps (secondes), du nombre de questions posées ainsi que du *gap* qui nous donne une borne supérieure sur le regret réel (cette borne est donnée par la valeur du MMR à la fin de l'exécution de l'algorithme). Le gap est exprimé en pourcentage de la borne supérieure de l'échelle de valeurs d'un OWA. Les résultats sont donnés dans le tableau 3.1. Chaque case représente le résultat moyen obtenu sur l'application de l'algorithme à 30 instances avec une limite de temps de 20 minutes.

p	n = 3			n = 5			n = 10		
	temps(s)	questions	gap	temps(s)	questions	gap	temps(s)	questions	gap
20	0,2	2,7	0.0	0,5	2,8	0.0	8,3	4,7	0.0
50	0,9	3,9	0.0	1,9	6,8	0.0	81,8	11,2	0.0
100	0,6	5,3	0.0	5,8	11,2	0.0	411,7	23,1	0.0

TABLE 3.1 – Algorithme 3.1 pour l'élicitation d'un OWA - stratégie S1.

Ces résultats montrent que l'algorithme 3.1 est très performant pour la résolution de ce problème. Toutes les instances sont résolues de manière exacte dans le temps imparti. La plupart des instances sont résolues en moins de 10 secondes et les plus grandes instances ne dépassent pas les 7 minutes en moyenne. On peut remarquer que ces résultats sont largement meilleurs que pour l'algorithme de branch and bound interactif qui, pour la plupart des instances, ne renvoyait qu'une solution approchée car l'algorithme était interrompu du fait de la limite de temps imposée. Le nombre de questions est également relativement faible, seules les plus grandes instances (10 agents et 100 objets) posent plus de 11 questions en moyenne.

2. Comparaison des stratégies S1 et S2

On présente maintenant des résultats visant à comparer les deux stratégies de choix des questions S1 et S2. Pour cela on observe l'évolution de la valeur du MMR (qui définit le critère d'arrêt) au cours de l'exécution de l'algorithme. Le résultat est donné par la figure 3.7 qui nous donne pour chaque stratégie la valeur moyenne, sur 30 instances, du MMR après chaque question (en pourcentage du MMR initial). Sur cette figure, on compare également les stratégies S1 et S2 à une stratégie aléatoire S3 consistant à choisir la question à poser en prenant deux solutions aléatoirement parmi les solutions correspondant aux solutions optimales pour les points extrêmes du polyèdre.

On peut voir que les deux stratégies S1 et S2 sont bien plus performantes que la stratégie S3, l'évolution de la valeur du MMR étant bien plus importante. Par exemple, pour obtenir une valeur de MMR à moins de 10% du regret initial, moins de 5 questions sont nécessaires pour S1, 17 pour S2 et 30 ne suffisent pas à atteindre ce seuil pour S3. Ce qui montre l'intérêt de nos stratégies de choix des questions. On peut également voir que la stratégie S1 permet de réduire la valeur du MMR plus rapidement que S2. Cette différence provient du fait que S1 est spécialement conçue pour réduire la valeur du MMR d'une question à l'autre, tandis que S2 a pour objectif de contrôler la réduction de l'espace des paramètres afin de pouvoir borner le nombre de questions posées.

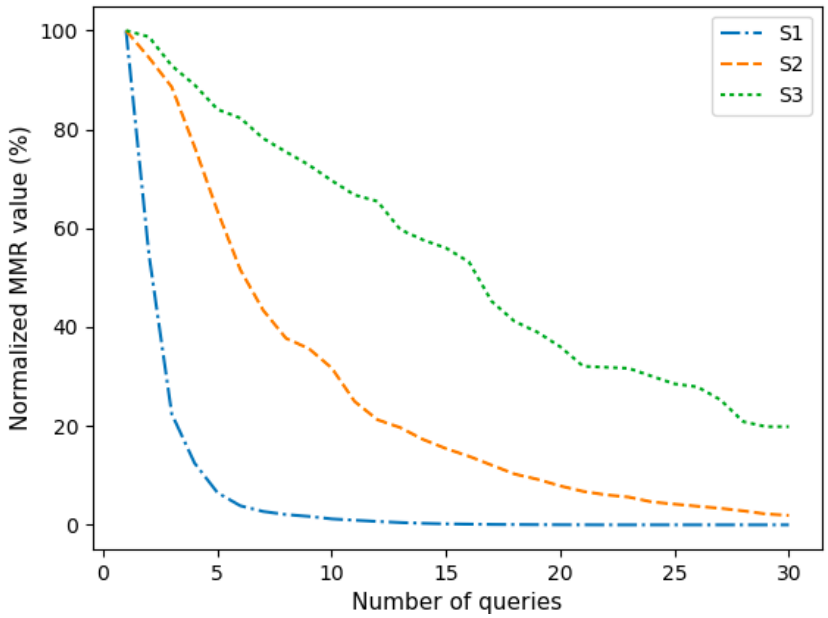


FIGURE 3.7 – Algorithme 3.1 avec les stratégies S1, S2 et S3 (10 agents, 50 items)

3. Problème d'affectation robuste

Nous avons également testé[2] l'algorithme 3.1 sur des instances du problème d'affectation robuste. Pour cela, nous avons généré aléatoirement des instances de tailles $p = 20, 50, 100$ et 150 évaluées selon $n = 3, 5$ et 10 scénarios. Les tableaux 3.2 et 3.3 résument les résultats obtenus en utilisant les stratégies S1 et S2. Les résultats donnés représentent une moyenne sur 20 instances avec une limite de temps de 20 minutes. On évalue l'algorithme en termes de temps (secondes), du nombre de questions posées ainsi que du *gap* donné par la valeur du MMR à la fin de l'algorithme (en pourcentage de l'échelle de valeurs). Les cases contenant des "-" indiquent qu'aucun résultat n'a été obtenu dans la limite de temps imposée.

p	n = 3			n = 5			n = 10		
	temps(s)	questions	gap	temps(s)	questions	gap	temps(s)	questions	gap
20	1.3	2.0	0.0	14.4	3.8	0.0	1200	6.75	$< 10^{-4}$
50	11.9	2.9	0.0	360.2	5.6	0.0	1200	-	-
100	28.2	2.7	0.0	966.9	5.9	0.0	1200	-	-
150	49.8	1.9	0.0	1200	4.5	$< 10^{-5}$	1200	-	-

TABLE 3.2 – Algorithme 3.1 pour l'élicitation d'un OWA - stratégie S1.

Ce premier tableau montre l'efficacité de l'algorithme pour les instances évaluées selon 3 ou 5 scénarios. En comparaison avec l'algorithme de ranking, on voit une meilleure efficacité de l'approche qui retourne une solution optimale (à 10^{-4}% près) pour toutes les instances telles que $n = 3$ ou 5 : les temps donnés sont largement inférieurs et le nombre de questions posées est plus petit que pour le ranking.

2. Mêmes caractéristiques de machine que pour le ranking : Intel Core i7-4770 CPU avec 11GB de RAM

Notons que l'on observe des résultats équivalents lorsque l'on utilise la stratégie S2. L'algorithme retourne en effet une solution optimale (à $10^{-3}\%$ près) pour toutes les instances telles que $n = 3$ ou 5. On peut également constater, comme nous l'avons fait sur la figure 3.7, que le nombre de questions posées est supérieur au nombre de questions posées pour la stratégie S1.

p	n = 3			n = 5			n = 10		
	temps(s)	questions	gap	temps(s)	questions	gap	temps(s)	questions	gap
20	2.3	5.0	0.0	28.9	8.8	0.0	1200	13.7	$< 10^{-3}$
50	12.8	3.2	0.0	626.7	11.6	0.0	1200	-	-
100	55.5	6.1	0.0	1200	10.1	$< 10^{-4}$	1200	-	-
150	144.3	5.9	0.0	1200	7.7	10^{-4}	1200	-	-

TABLE 3.3 – Algorithme 3.1 pour l'élicitation d'un OWA - stratégie S2.

Néanmoins, la qualité de l'algorithme se voit détériorée pour les instances évaluées sur $n = 10$ scénarios, et ce, quelle que soit la stratégie utilisée (S1 ou S2). On voit en effet qu'à partir de $p = 50$, l'algorithme ne permet même par de déterminer une recommandation en un temps inférieur à 20 minutes. Ceci est dû au manque d'efficacité du programme linéaire associé au problème d'affectation. Dans ce cas, le nombre de programmes linéaires à résoudre en une seule étape devient rédhibitoire. Cette faiblesse a participé à motiver l'adaptation de cette méthode à une approche utilisant des *classes de satisfactions* afin de résoudre des problèmes où les solutions sont évaluées sur un grand nombre de critères. Le reste de la section est consacré à la présentation de cette approche.

3.3 Résolution de grandes instances par classes de satisfaction

Lorsque le problème traité implique un très grand nombre d'objectifs, notamment pour les problèmes multi-agents avec un très grand nombre d'agents, l'espace des paramètres devient également très grand (une dimension par agent). Dans ce cas, la méthode proposée (algorithme 3.1) devient inefficace car : d'une part, comparer des paires de solutions évaluées selon un grand nombre d'objectifs peut s'avérer être une tâche difficile à accomplir pour le décideur. Et d'autre part, la performance de l'algorithme 3.1 peut se dégrader considérablement car le nombre de points extrêmes devient très grand, ce qui implique un grand nombre de programmes linéaires à résoudre. De plus, exiger une équité parfaite de la satisfaction individuelle des agents n'a pas d'intérêt pratique lorsque le nombre d'agents est très grand. On propose alors de se ramener à une évaluation plus simple en regroupant les agents (ou critères) par classes selon leur satisfaction. Une classe contiendra un pourcentage donné des agents ayant une satisfaction similaire. La satisfaction globale d'une classe est ensuite définie comme étant la somme des satisfactions des agents y appartenant. On contrôle alors l'équité des classes d'agents en pondérant la satisfaction des classes d'agents par l'importance que l'on accorde à chaque classe. L'objectif est ici de déterminer une solution au problème telle que les classes d'agents sont satisfaites de manière équitable. Ainsi, la dimension de l'espace des paramètres est réduite puisque le nombre de classes est largement inférieur au nombre total d'agents. Notons que ces

classes ne peuvent être définies a priori puisque la satisfaction de chaque agent change en fonction de la solution considérée. Ainsi, les classes d'agents changent avec chaque solution en fonction de la manière dont les satisfactions des agents sont distribuées. Il est alors nécessaire de concevoir une méthode de résolution adaptée à cet aspect.

Plus précisément, supposons que l'on traite un problème multi-agents avec n agents. Étant donnés une solution x et son vecteur de performances $u(x)$, dont les composantes représentent les satisfactions individuelles des agents, on partitionne les agents en c classes de la manière suivante : on note $u_{()}(x)$ le vecteur de performances de x trié dans l'ordre croissant de ses composantes, la première classe contient les $m = n/c$ premiers éléments de $u_{()}(x)$, ce qui correspond aux m agents les moins satisfaits de la solution. La seconde classe contient les m éléments suivants de $u_{()}(x)$, ... etc. On définit ainsi le vecteur de satisfaction de classes C de la manière suivante [3] :

$$C = (C_1, .., C_c) = (\sum_{j=1}^{m} u_{(j)}(x), \sum_{j=m+1}^{2m} u_{(j)}(x), \ldots, \sum_{j=(c-1)m+1}^{cm} u_{(j)}(x))$$

On donne alors la définition de l'OWA^c de la solution x par :

Définition 3.1. *Soit x une solution de $\mathcal{X}$ et $C \in \mathbb{R}_+^c$ le vecteur de satisfaction de classes associé et soit $w \in W$ un vecteur de pondération de dimension c, l'*OWA^C *du vecteur C est donnée par :*

$$\text{OWA}_w^c(x) = \sum_{i=1}^{c} w_i C_i$$

De cette manière, l'espace des critères est réduit ; on passe de n agents à un nombre de classes $c < n$ largement inférieur. Les paires de solutions sont alors plus faciles à comparer pour le décideur, et la charge d'élicitation est réduite puisqu'il y a moins de paramètres à éliciter.

Afin de pouvoir utiliser l'algorithme 3.1, nous avons besoin d'une méthode de calcul d'une solution dans $\arg\max_{x \in \mathcal{X}} \text{OWA}_w^c(x)$ (ligne 4 de l'algorithme). Ce calcul n'est pas évident car la détermination des classes de satisfaction dépend de la solution envisagée et non directement des agents. Il n'existe, à notre connaissance, aucune méthode permettant d'effectuer ce calcul. On propose ici une adaptation de la linéarisation de l'OWA d'Ogryczak et Śliwiński [2003] au cas où le souci d'équité porte sur des classes d'agents définies par leurs satisfactions relatives (quantiles).

Programmation linéaire pour le problème de satisfaction de classes équitables

La linéarisation d'Ogryczak et Śliwiński [2003] utilise la formulation de l'OWA d'une solution x par les composantes du vecteur de Lorenz associé. L'adaptation que l'on propose utilise uniquement un sous-ensemble des composantes du vecteur de Lorenz. On rappelle d'abord que, étant donnés une solution x et son vecteur de performances $u(x)$, on définit le vecteur de Lorenz de la solution x par :

$$L_i(x) = \sum_{j=1}^{i} u_{(j)}(x), \forall i \in \{1, \ldots, n\}$$

3. On présente ici des classes de satisfaction de mêmes tailles et avec $n\%c = 0$ pour simplifier mais on peut utiliser une définition similaire des classes lorsque les classes ne sont pas toutes de même taille.

On remarque alors qu'une composante C_i du vecteur de satisfaction de classes peut être définie par deux composantes de Lorenz particulières :

$$C_i = \sum_{j=(i-1)m+1}^{im} u_{(j)}(x) = L_{im}(x) - L_{(i-1)m}(x)$$

On en déduit la formulation de $\mathrm{OWA}_w^c(x)$ par les composantes de Lorenz :

$$\mathrm{OWA}_w^c(x) = \sum_{i=1}^{c} w_i(L_{im}(x) - L_{(i-1)m}(x))$$

ou encore

$$\mathrm{OWA}_w^c(x) = \sum_{i=1}^{c-1} L_{im}(x)(w_i - w_{i+1}) + L_{cm}(x)w_c \tag{3.4}$$

L'OWA du vecteur de satisfaction de classes s'exprime donc comme une somme pondérée d'un sous-ensemble de composantes de Lorenz. On peut alors utiliser la formulation d'Ogryczak en omettant les variables et contraintes correspondant aux composantes de Lorenz que l'on n'utilise pas dans cette formulation. La linéarisation consiste à écrire les composantes de Lorenz $L_i(x), i \in \{1, \ldots, c\}$, sous forme de programmes linéaires :

$$L_i(x) = \begin{cases} \min & \sum_{j=1}^{n} \alpha_j^i u_j(x) \\ & \sum_{j=1}^{n} \alpha_j^i = i \\ & 0 \leq \alpha_j^i \leq 1 \qquad \forall j \in N \end{cases}$$

Afin de se ramener à un problème de maximisation, on remplacera les composantes de Lorenz par le programme dual de ce programme linéaire :

$$L_i(u) = \begin{cases} \max & ir^i - \sum_{j=1}^{n} b_j^i \\ & r^i - b_j^i \leq u_j \qquad \forall j \in N \\ & b_j^i \geq 0 \qquad \forall j \in N \end{cases} \tag{3.5}$$

On intègre alors les contraintes de ce programme linéaire au programme correspondant au problème traité en remplaçant chaque composante L_i de la fonction objectif (la maximisation de l'équation (3.4)) par l'objectif du système (3.5) correspondant. On obtient :

$$\begin{cases} \max & \sum_{i=1}^{c}(w_i - w_{i+1})(im\ r^{im} - \sum_{j=1}^{n} b_j^{im}) \\ & r^{im} - b_j^{im} \leq u_j(x) & \forall i \in \{1, \ldots, c\}, \forall j \in N \\ & b_j^{im} \geq 0 & \forall i \in \{1, \ldots, c\}, \forall j \in N \\ & x \in \mathcal{X} \\ & w_{c+1} = 0 \end{cases}$$

Ce programme linéaire sera alors utilisé pour déterminer la solution OWA-optimale correspondant à chaque point extrême calculé dans l'algorithme 3.1. Notons que ce PL contient moins de variables et de contraintes que la linéarisation classique (qui prend en compte la satisfaction individuelle des agents et non des classes d'agents), il est donc plus rapide à résoudre. De plus, l'espace des paramètres étant réduit, le nombre de programmes linéaires à résoudre est également réduit, ce qui accélère également la procédure.

Expérimentations numériques

On présente dans le tableau 3.4 quelques résultats obtenus en appliquant l'algorithme 3.1 avec la stratégie S1 pour la résolution du problème d'affectation multi-agents. Le problème d'affectation à n agents et n objets se définit de manière similaire au problème robuste (voir la définition de $\mathcal{X}$ dans la partie expérimentale de l'algorithme de ranking) à la différence que l'utilité globale d'une solution $x \in \mathcal{X}$ est donnée par $u(x) = (u_1(x), \ldots, u_n(x))$ où $u_i(x)$ représente la satisfaction de l'*agent* i pour la solution x et est donnée par $u_i(x) = (\sum_{j=1}^{n} u_{ij}x_{ij}, \ldots, \sum_{j=1} u_{ij}^{n}x_{ij})$ avec u_{ij} l'utilité que l'agent i associe à l'objet j. Étant donné un vecteur de pondération w, la qualité d'une solution x sera donnée par $\mathrm{OWA}_w^c(x)$.

Pour ces tests [4], nous avons généré des instances de taille $n = 20, 50, 100$ et 150. Nous considérons ici que les agents sont regroupés selon $c = 5$ ou $c = 10$ classes. La table 3.4 résume les résultats obtenus. Ces résultats sont évalués selon les mêmes critères que précédemment en effectuant, pour tout couple n, c, des moyennes sur 20 instances avec une limite de temps de 20 minutes :

n	c=5			c=10		
	temps(s)	questions	gap	temps(s)	questions	gap
20	2.4	1.5	0.0	6.4	2.45	0.0
50	14.8	1.7	0.0	34.1	2.85	0.0
100	249.9	1.45	0.0	1200	1.95	$<10^{-3}$
150	355.7	1.1	0.0	1200	-	-

TABLE 3.4 – Algorithme 3.1 pour le problème d'affectation multi-agents

Ce dernier tableau montre de très bonnes performances de l'algorithme 3.1 pour la résolution des instances traitées. L'algorithme retourne une solution optimale (à 10^{-3} près) en un temps relativement court et avec très peu de questions pour la plupart des instances. Seules les plus grandes instances qui considèrent 10 classes de satisfaction ne peuvent être résolues dans le temps imparti. On remarque tout de même que l'on peut résoudre des instances de plus grande taille en utilisant les classes de satisfaction. Effectivement, en comparaison avec le tableau 3.2, on voit bien que les performances données ici sont bien meilleures pour tous les critères d'évaluation considérés : temps, nombre de questions et qualité de la recommandation.

3.4 Conclusion du chapitre

Dans ce chapitre, nous nous sommes intéressés à la conception d'une méthode d'élicitation des préférences fondée sur une minimisation directe des regrets. Cet algorithme permet la résolution de problèmes mêlant simultanément trois difficultés : la nature combinatoire de l'espace des solutions, le fait de considérer que plusieurs points de vue, critères, ou scénarios (potentiellement conflictuels) peuvent exister concernant l'évaluation d'une

4. Mêmes caractéristiques de machine que les expérimentations précédentes concernant l'affectation robuste : Intel Core i7-4770 CPU avec 11GB de RAM

solution, et finalement la représentation des préférences du décideur par un modèle non-linéaire qui ne respecte pas le principe d'optimalité de Bellman. La méthode que nous introduisons dans ce chapitre permet de surmonter ces difficultés en effectuant une exploration de l'espace des solutions guidée par une exploration de l'espace des paramètres, et plus particulièrement, une exploration des points extrêmes du polyèdre représentant l'espace des paramètres. Pour cela, nous avons exploité la convexité de l'ensemble d'incertitude et la linéarité de la fonction d'agrégation en son paramètre pour réduire le calcul des regrets à la résolution d'un nombre polynomial (en le nombre de points extrêmes) de programmes linéaires. Cette méthode est, à notre connaissance, la première méthode permettant d'optimiser les regrets sur domaine combinatoire.

Une extension intéressante de ce travail serait de déterminer une stratégie du choix de la question efficace (en termes d'informativité) et qui puisse permettre de contrôler le nombre de points extrêmes du polyèdre des paramètres possibles. En effet, le nombre de sommets de ce polyèdre est un élément déterminant de cette méthode puisque, pour déterminer la recommandation courante ainsi que la question à poser, on doit résoudre autant de programmes linéaires qu'il n'y a de points extrêmes afin de calculer les différentes valeurs de regrets. Or, l'ajout de nouvelles contraintes à ce polyèdre peut faire croître le nombre de points extrêmes de manière relativement significative, et peut donc ralentir la procédure d'élicitation.

Cet algorithme est *anytime* et *générique* et peut être adapté à la résolution de différents problèmes s'ils possèdent une formulation en programme linéaire efficace. Elle peut également être adaptée à l'élicitation de différents modèles de décision s'ils sont linéaires en leurs paramètres et s'ils possèdent une linéarisation efficace permettant d'adapter les programmes linéaires associés aux différents regrets.

La méthode présentée ici est très efficace en pratique (plus efficace que celles du chapitre précédent) puisqu'elle permet de déterminer des solutions optimales ou presque optimales rapidement et avec une connaissance très imprécise des préférences du décideur. Néanmoins, elle nécessite de faire une hypothèse relativement forte, celle que le décideur répond toujours de manière exacte et cohérente et qu'il ne revient jamais sur une réponse précédente. Par ailleurs, les stratégies de choix des questions présentées sont conçues de manière à ne jamais donner au décideur l'occasion de se contredire. Notons que c'est également le cas des méthodes du chapitre 2. De ce fait, ces méthodes sont très sensibles aux potentielles incohérences du décideur. Nous introduisons dans les chapitres suivants différents algorithmes permettant de résoudre ce problème.

Chapitre 4

Élicitation incrémentale par mise à jour Bayésienne d'une densité de probabilité

Résumé du chapitre

Ce chapitre est consacré à la conception d'algorithmes d'élicitation incrémentale des préférences, tolérants aux incohérences dans les informations préférentielles, pour la prise de décision multicritère. On suppose ici que les préférences du décideur peuvent être représentées par un modèle de décision paramétré tel qu'une somme pondérée (WS pour *Weighted Sum*), une somme pondérée ordonnée (OWA pour *Ordered Weighted Average*) ou une intégrale de Choquet 2-additive. Nous présentons une méthode d'élicitation fondée sur une représentation de l'incertitude sur les préférences du décideur par une distribution de probabilité continue. Cette méthode alterne analyse de l'espace des solutions et mise à jour de la densité de probabilité par régression linéaire Bayésienne. Nous présentons tout d'abord un algorithme d'élicitation des préférences pour la détermination d'une recommandation de bonne qualité sur domaine explicite, puis nous étendons la méthode à la résolution de problèmes définis sur domaine combinatoire. Les travaux présentés dans ce chapitre ont fait l'objet d'une publication et d'une soumission en cours de review : [Bourdache *et al.*, 2019b] pour une méthode d'élicitation par régression linéaire Bayésienne pour la résolution de problèmes définis sur domaine explicite, et [Bourdache *et al.*, 2020] pour l'extension de cette méthode au domaine combinatoire.

4.1 Introduction

Les procédures d'élicitation incrémentale que nous avons vues dans le chapitre précédent permettent de réduire efficacement l'incertitude concernant les préférences du décideur et de déterminer une recommandation optimale en un nombre de questions relativement petit. L'efficacité de ces méthodes vient de la prise en compte des informations préférentielles par l'ajout de contraintes irréversibles permettant de restreindre rapidement l'espace des paramètres. Néanmoins, ces méthodes nécessitent des réponses précises et parfaitement cohérentes de la part du décideur. Il peut pourtant arriver qu'un décideur se trompe ou présente des incohérences dans ses réponses. Par exemple, lorsque le modèle de décision utilisé ne convient pas parfaitement à la modélisation de ses préférences, lorsque deux solutions sont proches et que le décideur ne parvient pas à les comparer avec précision, ou tout simplement parce qu'il change d'avis au fur et à mesure de la découverte de nouvelles alternatives. Lorsque de telles incohérences se produisent, l'irréversibilité de l'ajout des contraintes constitue une faiblesse de ces méthodes car la qualité de la recommandation peut être fortement impactée. L'exemple suivant illustre cet impact :

Exemple 4.1. *On considère un problème de choix bicritère où l'ensemble des alternatives est défini par* $\mathcal{X} = \{x^0, \dots, x^{100}\}$ *avec* $x^i = (i, 100 - i), i = 0, \dots, 100$. *Les points de* $\mathcal{X}$ *se trouvent sur le segment en gras de la figure 4.1(a). On suppose ici que* $u_j(x^i) = x^i_j, i \in \{0, \dots, 100\}, j \in \{1, 2\}$. *On parlera alors indifféremment de* x^i *et de* $u(x^i)$ *pour simplifier. On suppose aussi que le système de valeurs du décideur est représenté par un* OWA *de la forme :* $\text{OWA}_{\hat{w}}(x^i) = \hat{w} \min\{x^i_1, x^i_2\} + (1 - \hat{w}) \max\{x^i_1, x^i_2\}$, *où* $\hat{w} = 0.6$ *est le paramètre modélisant les préférences du décideur. Par conséquent, pour tout* $i \in \{0, \dots, 50\}$, $\text{OWA}_{\hat{w}}(x^i) = \hat{w}i + (1 - \hat{w})(100 - i) = 40 + 0.2i$ *et pour tout* $i \in \{51, \dots, 100\}$, $\text{OWA}_{\hat{w}}(x^i) = \hat{w}(100 - i) + (1 - \hat{w})i = 60 - 0.2i$. *La solution optimale pour le décideur est donc la solution* x^{50} *de valeur* $\text{OWA}_{\hat{w}}(x^{50}) = 50$. *Cette solution est représentée par un point vert sur la figure 4.1(a). Notons que les solutions pour lesquelles le décideur est le moins satisfait sont les solutions* x^0 *et* x^{100} *(représentées par des points bleus sur la figure 4.1(a)) qui ont une valeur d'OWA de* 40.

Supposons maintenant que le poids réel du décideur $\hat{w}$ *n'est pas précisément connu et qu'il doit être élicité. L'ensemble des paramètres possibles est défini par* $W = [0, 1]$ *qui est représenté par le segment rouge de la figure 4.1 (b) où le poids réel* $\hat{w}$ *est représenté par un point vert.*

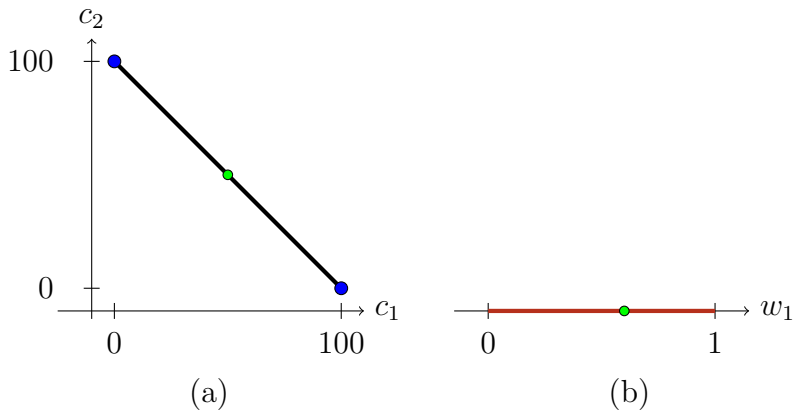


FIGURE 4.1 – Ensemble $\mathcal{X}$ à gauche et espace des paramètres W à droite.

*Supposons que le décideur ait à comparer deux solutions x^i et x^k telles que $0 \leq i < k \leq 50$. Lorsque les indices i et k sont proches, leurs valeurs d'*OWA *le sont également et le décideur peut ne pas être suffisamment sûr de ses préférences pour répondre correctement. Supposons donc qu'il se trompe en déclarant préférer (strictement) x^i à x^k. L'inégalité $\text{OWA}_w(x^i) > \text{OWA}_w(x^k)$ est alors ajoutée à la définition de l'ensemble d'incertitude. Cette inégalité mène à la contrainte $wi + (1-w)(100-i) > wk + (1-w)(100-k)$, c'est-à-dire $2w(i-k) - (i-k) > 0$ avec $i < k$. On obtient alors l'ensemble $W \leftarrow \{w \in [0,1] \mid w < 1/2\} = [0, 1/2)$ illustré par le segment rouge de la figure 4.2 (b). Tout poids $w \in W$ est maintenant tel que $\text{OWA}_w(x)$ favorise la composante maximum de x. Tout w mène alors aux solutions optimales x^1 et x^{100} (représentées en bleu sur la figure 4.2 (b)) qui sont en réalité les moins bonnes solutions selon les préférences du décideur.*

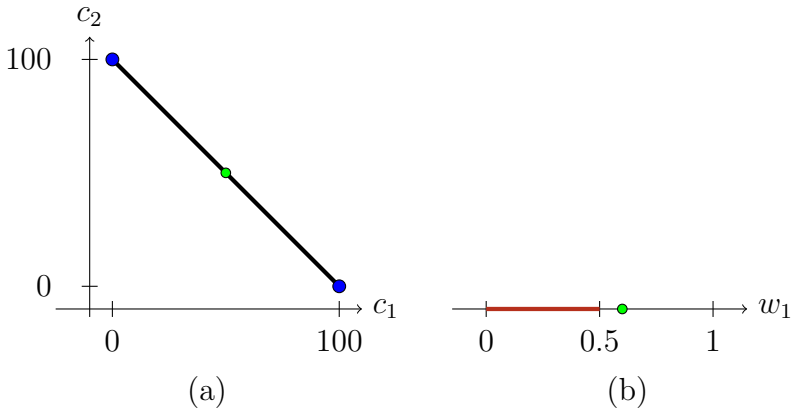


FIGURE 4.2 – Ensemble $\mathcal{X}$ à gauche et espace des paramètres W réduit à droite.

Cet exemple montre bien l'efficacité de cette approche qui détermine une solution optimale (lorsque les réponses du décideur sont correctes) malgré une connaissance de la valeur des paramètres encore très imprécise. Mais il montre également la faiblesse de l'approche qui, en cas de réponses erronées, peut recommander une solution de très mauvaise qualité. Cela vient du fait que les contraintes induites par les réponses du décideur restreignent l'espace des paramètres de manière irréversible et, lorsqu'une réponse est fausse, la contrainte induite exclut donc définitivement le poids réel du décideur de l'espace des paramètres, ce qui peut fortement réduire la pertinence de la recommandation.

Une manière d'obtenir une procédure moins sensible aux erreurs est d'adopter une approche probabiliste : une distribution de probabilité est associée à l'espace des paramètres pour représenter l'incertitude concernant la valeur précise des paramètres modélisant les préférences du décideur. À chaque étape de l'algorithme, les solutions du problème peuvent être évaluées en espérance pour déterminer une recommandation pertinente ou une question informative à poser. Lorsque de nouvelles informations préférentielles sont récoltées, l'espace des paramètres reste inchangé mais la distribution de probabilité est mise à jour, ce qui réduit progressivement le risque associé à une recommandation. Ici aussi, le but n'est pas d'éliciter le paramètre de manière précise mais de réduire suffisamment l'incertitude sur sa valeur pour pouvoir déterminer une recommandation avec un niveau de confiance suffisant. On propose dans ce chapitre une méthode *incrémentale* de *régression linéaire Bayésienne* qui alterne questions de comparaisons par paires et mises à jour d'une distribution Gaussienne de probabilité. Même si la méthode de mise à jour que l'on utilise

est initialement prévue pour des opérateurs d'agrégation linéaires dont les paramètres ne sont pas contraints autrement que par la positivité des composantes, notre algorithme ne se limite pas à de tels opérateurs puisque l'on l'adapte également à une représentation des préférences par un OWA ou par une intégrale de Choquet 2-additive. Pour cela, on utilise des bases de fonctions permettant d'exprimer ces opérateurs par des combinaisons linéaires de fonctions non-linéaires.

L'utilisation d'une distribution de probabilité pour représenter l'incertitude concernant les préférences du décideur a été exploitée dans différents travaux. Par exemple, dans le cadre de l'élicitation en décision dans le risque, Chajewska *et al.* [2000] proposent d'associer une distribution de probabilité aux valeurs d'utilité des conséquences possibles. Les différentes solutions du problème sont alors évaluées selon une fonction de perte espérée et une procédure d'élicitation est effectuée jusqu'à pouvoir déterminer une recommandation avec une valeur de perte espérée suffisamment petite. D'autres travaux utilisant des *Processus Gaussiens* ont été réalisés, comme par exemple ceux de [Chu et Ghahramani, 2005; Eric *et al.*, 2008; Zintgraf *et al.*, 2018]. Ces approches permettent d'approximer la valeur qu'associe le décideur aux solutions sans avoir à spécifier de modèle pour représenter ses préférences. Pour cela, le processus associe une distribution Gaussienne à la valeur de chaque solution. Ces distributions sont mises à jour à l'aide de questions préférentielles jusqu'à pouvoir aboutir à une bonne recommandation. Plus récemment, Sauré et Vielma ont proposé une méthode qui utilise une Gaussienne afin de définir une ellipsoïde de confiance [Sauré et Vielma, 2019]. Cette ellipsoïde définit une zone dans laquelle le poids réel du décideur se trouve avec une probabilité élevée. Ils utilisent une méthode de programmation linéaire à variables mixtes pour choisir des questions à poser au décideur de manière à réduire l'ellipsoïde de confiance le plus efficacement possible, jusqu'à ce qu'elle soit suffisamment petite pour pouvoir faire une recommandation pertinente. Dans un cadre différent, où on dispose uniquement d'un ensemble fixe de données récoltées a priori, quelques travaux ont été effectués pour la conception de méthodes d'estimation d'une distribution de probabilité a posteriori, par exemple [Tanner et Wong, 1987; Albert et Chib, 1993]. Cependant, ces méthodes ne sont pas incrémentales et ne permettent pas de déterminer un questionnaire adapté aux préférences spécifiques d'un décideur en interagissant (activement) avec ce dernier.

Les travaux présentés dans ce chapitre se différencient de l'existant par :

- l'utilisation d'un *modèle paramétré* pour représenter les préférences du décideur : on suppose que les valeurs des critères sont connues et que l'on élicite *incrémentalement* les *paramètres* du modèle d'agrégation considéré,
- alors que les travaux cités considèrent uniquement des modèles linéaires, nos algorithmes permettent également d'éliciter les paramètres de modèles *non-linéaires* tels qu'OWA et l'intégrale de Choquet,
- l'extension de notre algorithme aux domaines combinatoires. En effet, très peu de travaux de la littérature ont été proposés dans ce cadre pour la résolution de problèmes définis sur domaine combinatoire, on peut citer par exemple [Teso *et al.*, 2016] mais où l'approche proposée est différente car elle ne repose pas sur la mise à jour d'une distribution de probabilité mais utilise plutôt des modèles à vastes marges.

Contexte et notations. On considère dans ce chapitre que l'on résout un problème où l'espace des solutions $\mathcal{X}$ est défini de manière explicite (section 4.2) ou par un ensemble de contraintes linéaires à variables entières (section 4.3). Chaque solution de cet ensemble est évaluée selon n fonctions objectifs : évaluation selon n critères, point de vue de n agents ou évaluation selon n scénarios. L'image de $\mathcal{X}$ dans l'espace des critères est alors donnée par $\mathcal{Y} = \{u(x) \in \mathbb{R}^n, \forall x \in \mathcal{X}\}$ où $u(x)$ est le vecteur de performance associé à la solution x et dont la i^e composante représente la performance de x selon la fonction objectif i. On se place dans un cadre où aucune information concernant les préférences du décideur n'est connue mais on fait l'hypothèse qu'il convient de les modéliser par une *somme pondérée* (WS pour *Weighted Sum*), par une *somme pondérée ordonnée* (OWA pour *Ordered Weighted Average*) ou par une intégrale de Choquet 2-additive. L'algorithme de régression linéaire est présenté dans le cadre de l'élicitation d'une fonction linéaire f_ω, où ω est un vecteur de pondération défini dans $\mathcal{W} = \mathbb{R}^n_+$. On montre ensuite comment instancier f_ω par un opérateur non-linéaire dépendant du rang comme un OWA ou une intégrale de Choquet 2-additive qui ne s'écrivent (initialement) pas sous forme linéaire ou qui nécessitent des contraintes plus complexes que la positivité des composantes. Les questions que l'on pose au décideur pour éliciter ses préférences consistent à lui demander de comparer des paires de solutions que l'on choisit par une adaptation (que l'on donne plus loin) de la stratégie *CSS* au cadre probabiliste. Ces comparaisons permettent de réduire l'incertitude sur les préférences du décideur par le biais de mises à jour de la distribution de probabilité associée à l'espace des paramètres. On se place ici dans le cadre d'une maximisation de la valeur de l'opérateur considéré.

Organisation du chapitre. Le chapitre est organisé de la manière suivante : la section 4.2 présente l'algorithme d'élicitation incrémentale par régression linéaire Bayésienne pour la résolution d'un problème défini sur domaine explicite. Après avoir présenté une adaptation de la stratégie du choix de la question CSS au cadre probabiliste (où les solutions sont évaluées par des regrets *espérés*), on montre comment utiliser les questions préférentielles pour mettre à jour la distribution de probabilité courante lorsque l'on considère une fonction d'agrégation linéaire f_ω où la seule contrainte imposée sur la valeur du paramètre ω est la positivité de ses composantes. Puis on donne les formulations que l'on utilise afin d'étendre la méthode à des opérateurs non-linéaires plus complexes comme OWA et l'intégrale de Choquet 2-additive. On introduit ensuite, dans la section 4.3, une nouvelle méthode permettant d'étendre l'algorithme d'élicitation par régression linéaire Bayésienne à la résolution de problèmes définis sur domaines combinatoires. Pour cela, on introduit une méthode de calcul des regrets espérés par génération de contraintes et de variables.

4.2 Élicitation incrémentale par régression linéaire Bayésienne sur domaine explicite

On suppose dans cette section que l'on dispose d'un ensemble de solutions $\mathcal{X}$ défini explicitement. On se place dans un cadre où aucune information concernant les préférences du décideur n'est connue mais où l'on sait tout de même qu'il peut privilégier les solutions ayant un vecteur de performance équilibré et qu'il peut considérer qu'il existe des

synergies entre des paires de critères. Pour cela, on considère que l'on peut représenter ces préférences par un OWA (cf. définition 1.10) ou par une intégrale de Choquet à capacité 2-additive (cf. définition 1.13).

On associe à l'espace des paramètres $\mathcal{W}$ une densité de probabilité p qui représente notre incertitude concernant les préférences du décideur. Cette densité associe, à chaque valeur de paramètre possible, une vraisemblance relative de cette valeur. Elle permet également de calculer la probabilité que le paramètre ω se trouve dans un sous-ensemble donné $\mathcal{W}' \subseteq \mathcal{W}$. Lorsque l'on acquiert de nouvelles informations préférentielles, la densité p est mise à jour nous permettant ainsi d'obtenir une estimation plus précise de la valeur du paramètre représentant les préférences du décideur. Un exemple d'une telle mise à jour est donné figure 4.3 où l'on donne une densité Gaussienne p (à gauche) et la densité mise à jour après l'acquisition d'une information r :

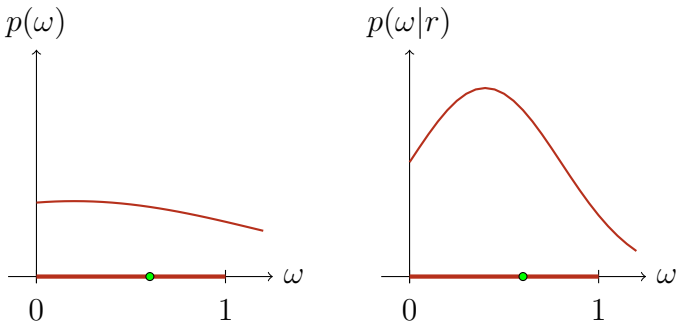


FIGURE 4.3 – Exemple de mise à jour d'une distribution de probabilité

En utilisant une telle modélisation de la connaissance des préférences, l'incertitude concernant la valeur du paramètre ω peut être réduite, par le biais de questions préférentielles, tout en veillant à ce que l'impact de réponses erronées ne soit pas trop important. En effet, contrairement à l'ajout de contraintes irréversibles sur l'espace des paramètres $\mathcal{W}$, la mise à jour d'une distribution de probabilité ne constitue pas une décision irréversible (à condition que la procédure de mise à jour soit correctement définie). Ainsi, l'acquisition d'une information préférentielle, cohérente ou non avec les informations précédentes, mène simplement à une nouvelle mise à jour de la fonction de densité et n'exclut pas systématiquement et irréversiblement une partie (potentiellement intéressante) de l'espace des paramètres. Avant de présenter notre méthode de mise à jour par régression linéaire Bayésienne, on introduit dans la sous-section suivante le principe général de notre méthode ainsi que la notion de regrets *espérés* qui nous permet d'établir une stratégie du choix de la question à poser.

4.2.1 Algorithme d'élicitation incrémentale

L'idée de l'algorithme est la suivante : on définit une densité de probabilité a priori de nature Gaussienne que l'on associera à l'espace des paramètres $\mathcal{W}$. À chaque étape de l'algorithme, une question est posée au décideur et sa réponse est utilisée afin de mettre à jour la densité de probabilité. Afin d'obtenir une procédure d'élicitation efficace, il est nécessaire de définir un choix judicieux de questions à poser afin d'obtenir le questionnaire

le plus informatif possible. On propose ici d'adapter la stratégie CSS [Wang et Boutilier, 2003] à l'utilisation d'une densité de probabilité pour modéliser l'incertitude concernant le système de valeurs du décideur.

Stratégie d'élicitation incrémentale

On considère ici un espace $\mathcal{W}_{norm}$ constitué des éléments *normalisés* de l'espace des paramètres $\mathcal{W}$, i.e., $\mathcal{W}_{norm} = \{\omega_{norm} = (\frac{\omega_1}{\sum_{j=1}^{n}\omega_j}, \ldots, \frac{\omega_n}{\sum_{j=1}^{n}\omega_j}), \forall \omega \in \mathcal{W}\}$. La normalisation des éléments de $\mathcal{W}$ est non-restrictive du point de vue de la représentativité des préférences mais est importante lorsque l'on calcule des regrets puisqu'il s'agit de comparer les valeurs f_ω des solutions de $\mathcal{X}$. Les valeurs des critères ainsi que celles des paramètres doivent donc être exprimées sur la même échelle afin que cette comparaison soit pertinente. Nous allons voir dans la suite que la définition de l'espace $\mathcal{W}_{norm}$ est indispensable car on utilise des regrets pour évaluer les différentes solutions. On utilise donc cette définition au lieu de prendre en compte la contrainte de normalisation dans la définition de la densité p.

Afin de sélectionner la prochaine question à poser pour mettre à jour la densité de probabilité p, on propose une stratégie du choix fondée sur la minimisation des regrets espérés. On définit ces regrets par :

Définition 4.1. *Soit f_ω une fonction d'agrégation de paramètre $\omega \in \mathcal{W}_{norm}$ et soit p_{norm} une fonction de densité définie sur $\mathcal{W}_{norm}$. On définit le regret espéré de la paire $(x, y) \in \mathcal{X}^2$ (Pairwise Expected Regret) par :*

$$\mathrm{PER}(x, y, p_{norm}) = \int \max\{0, f_\omega(y) - f_\omega(x)\} p_{norm}(\omega) d\omega$$

En d'autres termes, le PER de la paire (x, y) donne la perte d'utilité espérée engendrée par le choix de la solution x au lieu de la solution y. Ce regret constituant une intégrale difficile à calculer analytiquement, sa valeur est calculée approximativement en utilisant un échantillon représentatif S de paramètres générés dans $\mathcal{W}_{norm}$ suivant la densité p_{norm}. Ainsi, la discrétisation de $\mathcal{W}_{norm}$ par un échantillon nous permet de passer d'un calcul d'intégrale complexe au calcul d'une moyenne arithmétique :

$$\mathrm{PER}(x, y, S) = \frac{1}{|S|} \sum_{\omega \in S} \max\{0, f_\omega(y) - f_\omega(x)\} \tag{4.1}$$

Autrement dit, ce calcul consiste à sommer des intensités de préférences, $f_\omega(y) - f_\omega(x)$, pour tout paramètre ω tel que y est meilleure que x. Ce calcul ne peut avoir de sens que lorsque l'espace des paramètres est normalisé, d'où la nécessité d'obtenir un échantillon S d'éléments dans $\mathcal{W}_{norm}$. En pratique, notons que l'on ne dispose pas de la densité p_{norm} (associée à $\mathcal{W}_{norm}$) mais de la densité p (associée à $\mathcal{W}$). Le calcul de l'échantillon S selon p_{norm} peut néanmoins être réalisé facilement par la normalisation d'un échantillon tiré selon la distribution p (que l'on connaît). En effet, on voit bien que dans ce calcul (équation (4.1)), l'information importante contenue dans S est la proportion d'éléments ω de S tels que la solution y est meilleure que la solution x au sens de f_ω. Or, comme les éléments de S sont tous à composantes positives, l'ordre induit par f_ω est conservé lors de l'opération de normalisation de ω et, par conséquent, la proportion d'éléments de S qui favorisent la solution y (par rapport à la solution x) est également conservée. Il est donc

inutile de contraindre p à associer une densité nulle aux vecteurs de poids non-normalisés, car cette contrainte pourrait ralentir la procédure de mise à jour alors qu'elle n'est pas indispensable.

Une fois les valeurs $\text{PER}(x, y, S), \forall (x, y) \in \mathcal{X}^2$, obtenues on les utilise pour estimer le pire regret de chaque solution, i.e., le regret espéré maximum.

Définition 4.2. *Soit f_ω une fonction d'agrégation de paramètre $\omega \in \mathcal{W}_{norm}$ et soit p_{norm} une fonction de densité définie sur $\mathcal{W}_{norm}$. On définit le regret espéré maximum (Maximum Expected Regret) d'une solution $x \in \mathcal{X}$ par :*

$$\text{MER}(x, \mathcal{X}, p_{norm}) = \max_{y \in \mathcal{X}} \text{PER}(x, y, p_{norm})$$

Le MER d'une solution représente donc la perte d'utilité espérée maximum (pire cas) qu'engendre le choix de la solution x plutôt que n'importe quelle autre solution de $\mathcal{X}$. Tout comme pour le PER, on effectue en pratique un calcul approché du MER en utilisant un échantillon représentatif de $\mathcal{W}_{norm}$ tiré selon la distribution courante p puis normalisé. Pour un échantillon S donné, on note :

$$\text{MER}(x, \mathcal{X}, S) = \max_{y \in \mathcal{X}} \text{PER}(x, y, S)$$

Les valeurs de MER sont ensuite utilisées afin de comparer les solutions de $\mathcal{X}$ en terme de regret pire cas :

Définition 4.3. *Soit f_ω une fonction d'agrégation de paramètre $\omega \in \mathcal{W}_{norm}$ et soit p_{norm} une fonction de densité définie sur $\mathcal{W}_{norm}$. Le minimax regret espéré (MiniMax Expected Regret) de $\mathcal{X}$ est défini par :*

$$\text{MMER}(\mathcal{X}, p_{norm}) = \min_{x \in \mathcal{X}} \text{MER}(x, \mathcal{X}, p_{norm})$$

La valeur du MMER nous donne le plus petit regret espéré pire cas pouvant être engendré par le choix d'une solution de $\mathcal{X}$. Ici aussi, on effectue en pratique un calcul approché du MMER en utilisant un échantillon représentatif de $\mathcal{W}_{norm}$. Pour un échantillon S donné on note :

$$\text{MMER}(\mathcal{X}, S) = \min_{x \in \mathcal{X}} \text{MER}(x, \mathcal{X}, S)$$

On propose ici d'adapter le critère de choix fondé sur le minimax regret ainsi que la stratégie CSS [Boutilier *et al.*, 2006b] (décrite dans le chapitre précédent) à ces nouvelles définitions de regrets : à n'importe quelle étape de l'algorithme, on peut recommander au décideur la solution ayant le plus petit MER. Lorsque l'on souhaite préciser nos connaissances sur les préférences du décideur, les valeurs de regrets espérés sont utilisées comme critère pour évaluer l'intérêt des solutions de $\mathcal{X}$ et en extraire deux solutions intéressantes. On demandera alors au décideur de comparer la solution $x^* \in \arg\min_{x \in \mathcal{X}} \text{MER}(x, \mathcal{X}, S)$ (recommandation courante) à la solution qui constitue son meilleur adversaire au sens du MER, i.e., $y^* \in \arg\max_{y \in X} \text{PER}(x^*, y, S)$. La comparaison de ces deux solutions par le décideur apporte une nouvelle information préférentielle permettant de mettre à jour la densité p définie sur $\mathcal{W}$. Ces deux opérations sont répétées alternativement jusqu'à ce que la valeur du MMER soit inférieure à un seuil $\delta \geq 0$. La méthode est donnée par l'algorithme 4.1 :

Algorithme 4.1 :

Entrées : $\mathcal{X}$: ensemble de solutions défini explicitement ; δ : seuil de tolérance ; $p(\omega)$: fonction de densité a priori.

1 $i \leftarrow 1$;
2 **répéter**
3 $S \leftarrow$ échantillon selon $p(\omega)$;
4 $S \leftarrow$ échantillon S normalisé ;
5 $x^{(i)} \leftarrow \arg\min_{x \in \mathcal{X}} \mathrm{MER}(x, \mathcal{X}, S)$;
6 $y^{(i)} \leftarrow \arg\max_{y \in \mathcal{X}} \mathrm{PER}(x^{(i)}, y, S)$;
7 Demander au décideur s'il préfère $x^{(i)}$ ou $y^{(i)}$;
8 $r^{(i)} \leftarrow 1$ si la réponse est $x^{(i)} \succsim y^{(i)}$ et 0 sinon ;
9 Mettre à jour la densité $p(\omega)$ en utilisant $r^{(i)}$;
10 $i \leftarrow i + 1$;
11 **jusqu'à** $\mathrm{MMER}(\mathcal{X}, S) \leq \delta$;
12 **retourner** $x^* \in \arg\min_{x \in \mathcal{X}} \mathrm{MER}(x, \mathcal{X}, S)$

On montre dans la suite notre méthode de mise à jour de la fonction de densité $p(\omega)$ (ligne 9 de l'algorithme).

4.2.2 Mise à jour d'une distribution de probabilité par régression linéaire Bayésienne

Supposons qu'à l'étape i de l'algorithme 4.1 une distribution de probabilité $p(\omega)$ est associée à l'espace des paramètres défini par $\mathcal{W} = \mathbb{R}^n_+$. Pour mettre à jour cette densité, deux solutions $x^{(i)}$ et $y^{(i)}$ (choisies en utilisant une adaptation de la stratégie CSS) sont comparées par le décideur. Notons $r^{(i)}$ sa réponse que l'on représente par une variable binaire qui vaut 1 si et seulement si le décideur déclare qu'il préfère la solution $x^{(i)}$ à la solution $y^{(i)}$. En utilisant la formule de Bayes, la densité de probabilité a posteriori s'écrit :

$$p(\omega|r^{(i)}) = \frac{p(w)p(r^{(i)}|\omega)}{p(r^{(i)})}$$

ou encore pour simplifier :

$$p(\omega|r^{(i)}) \propto p(w)p(r^{(i)}|\omega)$$

Par définition de la variable $r^{(i)}$ la vraisemblance $p(r^{(i)}|\omega)$ peut être vue comme une distribution de Bernoulli [1]. Or, aucune distribution conjuguée [2] n'existe pour cette fonction de vraisemblance dans le cas multivarié. De ce fait, la densité de probabilité a posteriori est difficile à déterminer analytiquement en utilisant la formule de Bayes. On utilise alors une méthode consistant à réécrire la densité de probabilité $p(\omega|r^{(i)})$ en y introduisant une variable latente $z^{(i)}$ [Albert et Chib, 1993]. Pour cela, on suppose que la réponse du décideur dépend de cette variable latente qui représente l'intensité de préférence entre les

1. Distribution de probabilité discrète qui prend la valeur 1 avec une probabilité q et la valeur 0 avec une probabilité $1 - q$.

2. Lorsque la distribution de probabilité a priori appartient à la même famille que la distribution a posteriori, on dit qu'elles sont *conjuguées*. On dit également que la distribution a priori est conjuguée pour la vraisemblance.

deux solutions $x^{(i)}$ et $y^{(i)}$. Si on note $d^{(i)}$ le vecteur des différences de performances défini par $d_j^{(i)} = u_j(x^{(i)}) - u_j(y^{(i)}), \forall j \in \{1, \ldots, n\}$, alors, étant donné un paramètre $\omega \in \mathcal{W}$, la variable latente est définie par :

$$z^{(i)} = \omega d^{(i)} + \varepsilon^{(i)} \tag{4.2}$$

où $\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ est une variable aléatoire représentant un bruit Gaussien. Ainsi, la réponse $r^{(i)}$ vaut 1 si et seulement si $z^{(i)} \geq 0$. Le bruit $\varepsilon^{(i)}$ nous permet de modéliser de potentielles erreurs dans les réponses du décideur et/ou de prendre en compte les potentielles limitations descriptives de f_ω. En prenant un bruit *Gaussien*, on modélise le fait que les incohérences ont plus de chances de se produire lorsque la valeur $z^{(i)}$ est proche de 0, i.e., lorsque les valeurs des solutions comparées sont proches.

À l'aide de cette variable latente, nous reformulons la probabilité a posteriori $p(\omega|r^{(i)})$ comme une loi marginale de $w, z^{(i)}|r^{(i)}$. On écrit :

$$p(\omega|r^{(i)}) = \int p(\omega, z|r^{(i)})dz = \int p(\omega|z, r^{(i)})p(z|r^{(i)})dz \tag{4.3}$$

Cette intégrale n'est pas simple à calculer analytiquement. Tanner et Wong [1987] proposent une méthode itérative permettant d'approcher cette densité en résolvant l'équation de point fixe résultant de l'équation (4.3) où l'on remplace $p(z|r^{(i)})$ par l'expression :

$$p(z|r^{(i)}) = \int p(z, \theta|r^{(i)})d\theta = \int p(\theta|r^{(i)})p(z|\theta, r^{(i)})d\theta \tag{4.4}$$

La méthode consiste à alterner successivement des tirages d'échantillons $z_1, \ldots, z_m$ selon la distribution $p(z|r^{(i)})$ et des mises à jour de l'estimation de la densité $p(\omega|r^{(i)})$ en utilisant les valeurs $z_1, \ldots, z_m$. La méthode est détaillée dans l'algorithme 4.2 :

Algorithme 4.2 : Mise_à_jour($p(\omega|r^{(i)})$)

Entrées : $p(\omega)$: fonction de densité avant la question i ; m : taille des échantillon de z ; K : nombre maximum d'itérations.

```
1  k ← 0 ;
2  p_0(ω) ← p(ω) ;                    /* estimation courante de p(ω|r^(i)) */
3  répéter
4  |  pour j = 1 à m faire
5  |  |  Tirer θ selon p_k(ω) ;
6  |  |  Tirer z_j selon p(z|θ, r^(i)) ;
7  |  fin
8  |  p_{k+1}(ω) ← (1/m) Σ_{j=1}^{m} p_k(ω|z_j, r^(i)) ;
9  |  k ← k + 1 ;
10 jusqu'à stabilisation de p_k ou  k = K;
11 retourner p_k
```

Dans cet algorithme, la boucle *pour* permet d'obtenir un échantillon $z_1, \ldots, z_m$ selon la distribution $p(z|r^{(i)})$ [Tanner et Wong, 1987]. Pour cela, on alterne des tirages de valeurs de θ selon $p_k(\omega)$ (k^e estimation de la densité $p(\omega|r^{(i)})$ dans le processus itératif) et de valeurs z_j selon $p(z|\theta, r^{(i)})$. Pour que cet algorithme soit efficace, il doit être possible

de générer facilement et rapidement des échantillons selon les distributions $p(\omega|r^{(i)})$ et $p(z|\theta, r^{(i)})$, d'autant plus que l'on doit également générer des échantillons de valeurs de ω selon $p(\omega|r^{(i)})$ pour calculer les valeurs de regrets et appliquer la stratégie du choix de la question donnée plus haut. Nous allons voir dans la suite que l'on sait échantillonner efficacement selon ces deux distributions puisqu'elles sont Gaussiennes (propositions 4.1 et 4.2).

A l'issue de la boucle *pour*, l'algorithme permet de calculer une nouvelle estimation $p_{k+1}(\omega)$ de $p(\omega|r^{(i)})$ en utilisant le nouvel échantillon $z_1, \ldots, z_m$ (ligne 8 de l'algorithme). En effet, par l'équation (4.3), et puisque $z_1, \ldots, z_m$ est un échantillon représentatif de $p(z|r^{(i)})$ on écrit :

$$p(\omega|r^{(i)}) \approx \frac{1}{m} \sum_{j=1}^{m} p(\omega|z_j, r^{(i)}) \tag{4.5}$$

Ces deux étapes (boucle *pour* et mise à jour par l'équation 4.5) sont répétées jusqu'à convergence ou jusqu'à avoir atteint K itérations. Notons que *Tanner et Wong* [1987] montrent que l'algorithme 4.2 converge vers la distribution postérieure exacte.

On montre maintenant que les distributions de probabilité manipulées dans cet algorithme, ainsi que la distribution postérieure $p(\omega|r^{(i)})$ retournée, sont toutes de nature Gaussienne :

Proposition 4.1. *La densité de probabilité $p(z^{(i)}|\omega, r^{(i)})$ est définie par une Gaussienne tronquée :*

$$p(z^{(i)}|\omega, r^{(i)}) \propto \mathcal{N}(\omega d^{(i)}, \sigma^2)(\mathbb{1}_{\{z \geq 0 \ \& \ r^{(i)}=1\}} + \mathbb{1}_{\{z<0 \ \& \ r^{(i)}=0\}}) \tag{4.6}$$

où $\mathbb{1}_c$ est la fonction indicatrice qui vaut 1 si la condition c est vérifiée et 0 sinon.

Démonstration. Tout d'abord, notons que la densité de probabilité est tronquée à l'aide de la fonction indicatrice afin d'assurer que la variable latente $z^{(i)}$ soit cohérente avec la réponse donnée $r^{(i)}$, i.e., la densité d'une valeur $z^{(i)}$ qui n'est pas cohérente avec la réponse $r^{(i)}$ ($z^{(i)} \geq 0$ & $r^{(i)} = 0$ ou $z^{(i)} < 0$ & $r^{(i)} = 1$) doit être nulle.

Ensuite, la nature Gaussienne de la densité $p(z^{(i)}|\omega, r^{(i)})$ vient de la nature de la variable aléatoire $\varepsilon^{(i)}$. En effet, pour tout $t \in \mathbb{R}$, la fonction de répartition de $z^{(i)}$ conditionnellement à ω et $r^{(i)}$ est donnée par :

$$\begin{aligned} p(z^{(i)} < t|\omega, r^{(i)}) &= p(z^{(i)} < t|\omega, r^{(i)})(\mathbb{1}_{\{t<0 \ \& \ r^{(i)}=0\}} + \mathbb{1}_{\{t<0 \ \& \ r^{(i)}=1\}}) \\ &+ p(z^{(i)} < 0|\omega, r^{(i)})\mathbb{1}_{\{t \geq 0 \ \& \ r^{(i)}=0\}} + p(0 \leq z^{(i)} < t|\omega, r^{(i)})\mathbb{1}_{\{t \geq 0 \ \& \ r^{(i)}=1\}} \end{aligned}$$

où $\mathbb{1}_{\{t<0 \ \& \ r^{(i)}=1\}} = 0$ car dans ce cas toutes les valeurs possibles de $z^{(i)}$ qui sont inférieures à t ne sont pas cohérentes avec $r^{(i)}$, et le terme $p(z^{(i)} < 0|\omega, r^{(i)})$ est une constante et peut être omis. On a alors :

$$\begin{aligned} p(z^{(i)} < t|\omega, r^{(i)}) &\propto p(z^{(i)} < t|\omega, r^{(i)})\mathbb{1}_{\{t<0 \ \& \ r^{(i)}=0\}} + p(0 \leq z^{(i)} < t|\omega, r^{(i)})\mathbb{1}_{\{t \geq 0 \ \& \ r^{(i)}=1\}} \\ &\propto p(z^{(i)} < t|\omega, r^{(i)})\mathbb{1}_{\{t<0 \ \& \ r^{(i)}=0\}} \\ &+ [p(z^{(i)} < t|\omega, r^{(i)}) - p(z^{(i)} < 0|\omega, r^{(i)})]\mathbb{1}_{\{t \geq 0 \ \& \ r^{(i)}=1\}} \end{aligned}$$

Ici aussi le terme $p(z^{(i)} < 0|\omega, r^{(i)})$ est une constante et peut être omis :

$$\begin{aligned} p(z^{(i)} < t|\omega, r^{(i)}) &\propto p(z^{(i)} < t|\omega, r^{(i)})(\mathbb{1}_{\{t<0 \ \& \ r^{(i)}=0\}} + \mathbb{1}_{\{t \geq 0 \ \& \ r^{(i)}=1\}}) \\ &\propto p(\omega d^{(i)} + \varepsilon^{(i)} < t|\omega, r^{(i)})(\mathbb{1}_{\{t<0 \ \& \ r^{(i)}=0\}} + \mathbb{1}_{\{t \geq 0 \ \& \ r^{(i)}=1\}}) \\ &\propto p(\varepsilon^{(i)} < t - \omega d^{(i)}|\omega, r^{(i)})(\mathbb{1}_{\{t<0 \ \& \ r^{(i)}=0\}} + \mathbb{1}_{\{t \geq 0 \ \& \ r^{(i)}=1\}}) \end{aligned}$$

Or, $\varepsilon^{(i)}$ est une variable aléatoire Gaussienne $\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ donc :

$$p(z^{(i)} < t|\omega, r^{(i)}) \quad \propto \quad \Phi(t - \omega d^{(i)}; 0, \sigma^2)(\mathbb{1}_{\{t<0 \ \& \ r^{(i)}=0\}} + \mathbb{1}_{\{t\geq 0 \ \& \ r^{(i)}=1\}})$$

où la fonction $\Phi(.; a, b^2)$ désigne la fonction de répartition de la distribution $\mathcal{N}(a, b^2)$. Un résultat bien connu sur la transformation affine d'une variable aléatoire Gaussienne nous permet alors d'écrire :

$$p(z^{(i)} < t|\omega, r^{(i)}) \quad \propto \quad \Phi(t; \omega d^{(i)}, \sigma^2)(\mathbb{1}_{\{t<0 \ \& \ r^{(i)}=0\}} + \mathbb{1}_{\{t\geq 0 \ \& \ r^{(i)}=1\}})$$

Puisque la fonction de répartition de la variable $z^{(i)}$ conditionnellement à $\omega, r^{(i)}$ est donnée par la fonction de répartition d'une Gaussienne tronquée de moyenne $\omega d^{(i)}$ et d'écart type σ, on en déduit :

$$p(z^{(i)}|\omega, r^{(i)}) \propto \mathcal{N}(\omega d^{(i)}, \sigma^2)(\mathbb{1}_{\{z\geq 0 \ \& \ r^{(i)}=1\}} + \mathbb{1}_{\{z<0 \ \& \ r^{(i)}=0\}})$$

□

Montrons maintenant que la distribution de probabilité a posteriori $p(\omega|r^{(i)})$ est également Gaussienne. Si l'on suppose que l'incertitude concernant les préférences du décideur peut être représentée par une distribution Gaussienne tronquée associée à $\mathcal{W}$ (on tronque pour respecter la contrainte de positivité des composantes), i.e., $p(\omega) = \mathcal{N}(\mu_0, S_0) \times \mathbb{1}_{\omega\in\mathcal{W}}$, alors la distribution de probabilité a posteriori est une Gaussienne tronquée :

Proposition 4.2. *Si la distribution a priori $p(\omega)$ est une Gaussienne tronquée alors l'algorithme 4.2 permet d'approximer $p(\omega|r^{(i)})$ par une Gaussienne tronquée.*

Démonstration. Nous allons montrer par récurrence que la distribution retournée par l'algorithme 4.2 est une Gaussienne tronquée. Par hypothèse, nous savons que la distribution a priori $p_0(\omega)$ est une Gaussienne tronquée à 0 (les composantes de ω sont toutes positives). Supposons qu'après k mises à jour par l'algorithme 4.2, $p_k(\omega)$ est une Gaussienne de paramètres μ^k et S^k et montrons qu'à l'issue d'un nouveau tour de boucle *répéter*, p_{k+1} est également une Gaussienne. Pour cela, nous allons d'abord montrer que $p(w|z_j, r^{(i)}), \forall j \in \{1, \ldots, m\}$, est une Gaussienne. En utilisant le théorème de Bayes, on obtient pour toute valeur $j \in \{1, \ldots, m\}$:

$$p(\omega|z_j, r^{(i)}) \propto p(z_j|\omega, r^{(i)})p(\omega|r^{(i)})$$

Par la proposition 4.1 et par l'hypothèse de récurrence on a :

$$p(\omega|z_j, r^{(i)}) \propto \mathcal{N}(\omega d^{(i)}, \sigma^2) \times (\mathbb{1}_{\{z_j\geq 0 \ \& \ r^{(i)}=1\}} + \mathbb{1}_{\{z_j<0 \ \& \ r^{(i)}=0\}}) \times \mathcal{N}(\mu^k, S^k) \times \mathbb{1}_{\omega\in\mathcal{W}}$$

Pour simplifier la lecture, on notera dans le reste de la preuve :

$$p(\omega|z_j, r^{(i)}) \propto \mathcal{N}(\omega d^{(i)}, \sigma^2) \times \mathbb{1}_{z_j, r^{(i)}} \times \mathcal{N}(\mu^k, S^k) \times \mathbb{1}_{\omega\in\mathcal{W}} \tag{4.7}$$

On obtient alors le produit d'une distribution Gaussienne univariée tronquée (distribution de $z_j|\omega, r^{(i)}$) et d'une distribution Gaussienne multivariée tronquée (distribution courante de w). Nous allons alors reformuler l'expression de l'équation 4.7 de manière à obtenir

un produit de deux distributions normales multivariées afin de déterminer la nature de la distribution résultante, i.e., $p(\omega|z_j, r^{(i)})$.

En écrivant $(z_j - \omega d^{(i)}) = (\hat{\omega} - \omega)d^{(i)}$ avec $\hat{\omega} = ({d^{(i)}}^T d^{(i)})^{-1}{d^{(i)}}^T z_j$, la fonction de densité de $z_j|\omega, r^{(i)}$ s'écrit :

$$\begin{aligned} p(z_j|\omega, r^{(i)}) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\{-\frac{1}{2}\left(\frac{z_j - \omega d^{(i)}}{\sigma}\right)^2\} \times \mathbb{1}_{z_j, r^{(i)}} \\ &= \frac{1}{\sigma\sqrt{2\pi}} \exp\{-\frac{1}{2}\left(\frac{(\omega - \hat{\omega})d^{(i)}{d^{(i)}}^T(\omega - \hat{\omega})^T}{\sigma^2}\right)\} \times \mathbb{1}_{z_j, r^{(i)}} \\ &\approx \exp\{-\frac{1}{2}\left(\frac{(\omega - \hat{\omega})d^{(i)}{d^{(i)}}^T(\omega - \hat{\omega})^T}{\sigma^2}\right)\} \times \mathbb{1}_{z_j, r^{(i)}} \end{aligned} \tag{4.8}$$

où $d^{(i)}{d^{(i)}}^T$ désigne la matrice carrée $n \times n$ obtenue par le produit matriciel du vecteur colonne $d^{(i)}$ et du vecteur ligne ${d^{(i)}}^T$. On déduit de cette dernière formulation que :

$$p(z_j|\omega, r^{(i)}) \propto \mathcal{N}(\hat{\omega}, \sigma^2({d^{(i)}}{d^{(i)}}^T)^{-1}) \times \mathbb{1}_{z_j, r^{(i)}}$$

On obtient alors :

$$p(\omega|z_j, r^{(i)}) \propto \mathcal{N}(\hat{\omega}, \sigma^2({d^{(i)}}{d^{(i)}}^T)^{-1}) \times \mathcal{N}(\mu_k, \Sigma_k) \times \mathbb{1}_{z_j, r^{(i)}} \times \mathbb{1}_{\omega \in \mathcal{W}} \tag{4.9}$$

Le produit des deux distributions Gaussiennes mène à une autre distribution Gaussienne (cf. annexe A). Pour tout $j \in \{1, \ldots, m\}$, $p(\omega|z_j, r^{(i)})$ est donc bien une Gaussienne tronquée :

$$p(\omega|z_j, r^{(i)}) \propto \begin{cases} 0 & \text{si } \omega \notin \mathcal{W} \text{ ou } z_j \times r^{(i)} \leq 0 \\ \mathcal{N}(\mu^j, S) & \text{sinon} \end{cases}$$

avec :

$$S^{-1} = \frac{1}{\sigma^2}{d^{(i)}}{d^{(i)}}^T + S_k^{-1} \tag{4.10}$$

$$\mu^j = S(\frac{1}{\sigma^2}{d^{(i)}}^T z_j + S_k^{-1}\mu_k) \tag{4.11}$$

De ce fait, la distribution p_{k+1} obtenue à l'étape $k+1$ de l'algorithme en calculant la somme donnée par l'expression (4.5) est également une Gaussienne tronquée (cf. annexe A pour la somme de Gaussiennes) dont les paramètres μ_{k+1} et S_{k+1} sont :

$$\mu_{k+1} = \frac{1}{m}\sum_{j=1}^{m}\mu^j \tag{4.12}$$

$$S_{k+1} = S \tag{4.13}$$

□

Nous avons donc déterminé une méthode de mise à jour de la densité $p(\omega)$ par régression linéaire Bayésienne. Pour l'appliquer, on remplacera la ligne 9 de l'algorithme 4.1 par un appel de la fonction définie par l'algorithme 4.2. Ainsi, le choix du critère d'arrêt de l'algorithme 4.1 se justifie par une tendance à la baisse de la valeur du MMER

au fur et à mesure de l'acquisition de nouvelles informations. En effet, on peut observer empiriquement une réduction de la variance de chaque composante $\omega_i, i \in \{1, \ldots, n\}$, au fur et à mesure de l'application de l'algorithme 4.2. De ce fait, la valeur du MMER tend asymptotiquement vers 0. Nous verrons dans la partie expérimentale que l'on observe effectivement une tendance à la baisse de la valeur du MMER au cours de l'algorithme.

Comme nous avons pu le préciser précédemment, l'opération de mise à jour définie par l'algorithme 4.2 est une méthode de régression Bayésienne à partir des réponses données par le décideur pour l'élicitation du paramètre ω (à composantes positives) d'une fonction *linéaire*. Elle ne s'applique donc pas directement à un OWA à poids décroissants ou à une intégrale de Choquet 2-additive qui sont linéaires *par morceaux*. De plus, lorsque l'on représente des préférences qui privilégient les vecteurs de performances équilibrés (exigence d'équité ou de robustesse), des contraintes autres que la positivité des composantes (voir plus loin) sont ajoutées à l'espace des paramètres d'OWA et de l'intégrale de Choquet. Il en est de même lorsque l'on considère qu'il existe des interactions entre des paires de critères (intégrale de Choquet 2-additive). Ces nouvelles contraintes rendent la mise à jour de la distribution p plus complexe car elles ne sont pas assurées par les distributions obtenues par l'algorithme 4.2 (les paramètres ne respectant pas ces contraintes n'ont pas nécessairement une densité nulle). De plus, l'application de cet algorithme ne convient pas (directement) à l'intégrale de Choquet qui est paramétrée par une capacité et non par un vecteur de poids. On propose dans la suite de surmonter ces difficultés en utilisant des formulations de ces opérateurs par combinaisons *convexes* de fonctions de base qui permettent d'appliquer l'algorithme 4.2 facilement.

4.2.3 Régression linéaire Bayésienne pour OWA et Choquet

Afin d'étendre la régression linéaire Bayésienne à une fonction f_ω définie par un OWA ou une intégrale de Choquet, on propose de reformuler f_ω comme une combinaison linéaire des fonctions non-linéaires d'une base génératrice $g_1, \ldots, g_q$ [Bishop, 2006], i.e. d'écrire f_ω sous la forme :

$$f_\omega(x) = \alpha g(x) = \sum_{i=1}^{q} \alpha_i g_i(x) \tag{4.14}$$

où $\alpha = (\alpha_1, \ldots, \alpha_q)$ est un vecteur de pondération et $g(x) = (g_1(x), \ldots, g_q(x))$ donne les q fonctions de la base génératrice. Ainsi, la fonction f_ω est exprimée comme une fonction linéaire en $g(x)$. On introduit maintenant les bases de fonctions génératrices pour OWA et l'intégrale de Choquet 2-additive.

L'opérateur OWA général. Pour rappel, l'OWA d'une solution x associée au vecteur de performance $u(x)$ est donnée par :

$$\mathrm{OWA}_w(x) = \sum_{i=1}^{n} w_i u_{(i)}(x) \tag{4.15}$$

où w est un vecteur de pondération à composantes positives et $u_{()}$ représente le vecteur de performance $u(x)$ trié dans l'ordre croissant. De manière évidente, l'équation 4.15 est déjà sous la forme indiquée dans l'équation 4.14. En effet, $\mathrm{OWA}_w(x)$ est une combinaison linéaire des fonctions $g_i(x) = u_{(i)}(x), \forall i \in \{1, \ldots, n\}$, dont les coefficients sont $\alpha_i = w_i, \forall i \in \{1, \ldots, n\}$.

L'opérateur OWA concave. Lorsque l'on souhaite modéliser une exigence d'équité ou de robustesse, le paramètre w de l'opérateur OWA doit respecter, en plus de la contrainte de positivité des w_i, la contrainte $w_1 \geq \ldots \geq w_n$ afin de favoriser les solutions ayant un vecteur de performance équilibré (voir la partie dédiée à OWA dans la sous-section 1.2.2). On obtient alors une classe de fonctions OWA concaves. L'ajout de la contrainte des composantes décroissantes rend la régression plus difficile à effectuer car l'algorithme 4.2 ne permet pas de garantir que les vecteurs de poids non triés auront une densité nulle. Une première solution pourrait être de simplement rejeter tout vecteur non trié lors des opérations d'échantillonnage, mais cette solution peut s'avérer être très coûteuse en temps. Pour pallier cela, on propose d'utiliser la formulation d'$\mathrm{OWA}_w(x)$ utilisant le vecteur de Lorenz associé à la solution x (cf. équation (1.4)) :

$$\mathrm{OWA}_w(x) = \sum_{i=1}^{n-1}(w_i - w_{i+1})L_i(x) + w_n L_n(x)$$

où $L_i(x)$ est la i^e composante du vecteur de Lorenz associé à x et est définie par $L_i(x) = \sum_{k=1}^{i} u_{(k)}(x)$. Ainsi, cette formulation nous permet de définir notre base de fonctions g par $g_i(x) = L_i(x), \forall i \in \{1, \ldots, n\}$. Ici, les coefficients α_i de la combinaison convexe sont donnés par $\alpha_i = w_i - w_{i+1}, \forall i \in \{1, \ldots, n-1\}$, et $\alpha_n = w_n$. On peut facilement voir que la contrainte $w_1 \geq \cdots \geq w_n \geq 0$ est équivalente à $\alpha_i \geq 0$ pour $i \in \{1, \ldots, n\}$. On écrit alors :

$$\mathrm{OWA}_w(x) = \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{i} u_{(j)}(x) \right) = \sum_{i=1}^{n} \alpha_i L_i(x) \tag{4.16}$$

Exemple 4.2. *Supposons que l'on dispose d'une solution x associée au vecteur de performance $u(x) = (0.3, 0.5, 0.8)$ et que les préférences du décideur sont représentées par le vecteur de pondération $w = (0.5, 0.3, 0.2)$. En utilisant la formule classique de l'OWA, la solution x est évaluée par $\mathrm{OWA}_w(x) = 0.5 \times 0.3 + 0.3 \times 0.5 + 0.2 \times 0.8 = 0.46$.*

Utilisons maintenant la formulation de l'équation 4.16 : le vecteur de Lorenz associé à x est donné par $L(x) = (0.3, 0.3+0.5, 0.3+0.5+0.8) = (0.3, 0.8, 1.6)$. Le vecteur de pondération α correspondant à w est donné par $\alpha = (0.5-0.3, 0.3-0.2, 0.2) = (0.2, 0.1, 0.2)$. La valeur $\mathrm{OWA}_w(x)$ peut alors être calculée par $\mathrm{OWA}_w(x) = 0.2\times0.3+0.1\times0.8+0.2\times1.6 = 0.46$.

Intégrale de Choquet 2-additive. Pour rappel, l'intégrale de Choquet d'une solution x associée au vecteur de performance $u(x)$ est donnée par

$$C_v(x) = \sum_{i=1}^{n} [u_{(i)}(x) - u_{(i-1)}(x)] v(X_{(i)}) \tag{4.17}$$

où $u_{(.)}$ représente le vecteur de performance $u(x)$ trié dans l'ordre croissant et $X_{(i)}$ est l'ensemble des critères pour lesquels la performance de x est au moins aussi bonne que la performance $u_{(i)}(x)$. Une manière simple d'exprimer l'intégrale de Choquet comme une combinaison linéaire de fonctions non-linéaires est de considérer une décomposition linéaire de la capacité v sous la forme $v = \sum_{i=1}^{q} \alpha_i v_i$, où $v_1, \ldots, v_q$ est une base de capacités formant une famille génératrice de l'ensemble des capacités possibles. En effet, en utilisant

une telle formulation pour v on a :

$$\begin{aligned} C_v(x) &= \sum_{i=1}^{n}[u_{(i)}(x) - u_{(i-1)}(x)]v(X_{(i)}) \\ &= \sum_{i=1}^{n}[u_{(i)}(x) - u_{(i-1)}(x)]\sum_{i=1}^{q}\alpha_i v_i(X_{(i)}) \\ &= \sum_{i=1}^{q}\alpha_i \sum_{i=1}^{n}[u_{(i)}(x) - u_{(i-1)}(x)]v_i(X_{(i)}) \\ &= \sum_{i=1}^{q}\alpha_i C_{v_i}(x) \end{aligned}$$

qui est bien une formulation linéaire en α. Une telle décomposition de v est connue pour la classe de capacités 2-additives [Grabisch *et al.*, 2009, Theorem 2.65, p.89] (voir la partie dédiée à Choquet dans la sous-section 1.2.2 pour la définition d'une capacité 2-additive). En effet, l'ensemble des capacités 2-additives est défini par un polyèdre convexe dont les points extrêmes sont des capacités particulières prenant leurs valeurs dans $\{0, 1\}$. Ainsi, toute capacité v peut être définie comme une combinaison convexe de ces points extrêmes.

Définition 4.4. *Soit $N = \{1, \ldots, n\}$, la classe des capacités 2-additives $v : 2^N \to [0, 1]$ est caractérisée par un polyèdre convexe dont les points extrêmes sont définis par deux familles de capacités prenant leurs valeurs dans $\{0, 1\}$:*

— Les jeux unanimes *: pour tout ensemble non vide $X \subseteq N$ défini par un singleton ou une paire :*

$$v_i(X) = \begin{cases} 1 & si\ Y_i \subseteq X \\ 0 & sinon \end{cases} \qquad \forall i \in \{1, \frac{n(n+1)}{2}\}$$

où $Y_i \subseteq N$ est un sous-ensemble non vide de N de taille au plus 2 ;

— Les conjuguées *: pour tout ensemble non vide $X \subseteq N$ défini par une paire :*

$$v_i(X) = \begin{cases} 1 & si\ Y_i \cap X \neq \emptyset \\ 0 & sinon \end{cases} \qquad \forall i \in \{\frac{n(n+1)}{2} + 1, n^2\}$$

où $Y_i \subseteq N$ est un sous-ensemble non vide de N de taille 2.

Ainsi, pour toute capacité 2-additive v, il existe n^2 coefficient positifs α_i tels que $\sum_{i=1}^{n^2}\alpha_i = 1$ et $v = \sum_{i=1}^{n^2}\alpha_i v_i$. On définit alors $g_i(x) = C_{v_i}(x), \forall i \in \{1, \ldots, n\}$, et on écrit :

$$C_\alpha(x) = \sum_i \alpha_i C_{v_i}(x) \tag{4.18}$$

Notons que la contrainte de normalisation des coefficients α_i n'est pas toujours respectée par l'algorithme 4.2 mais, comme nous avons pu le voir dans la sous-section 4.2.1, les éléments des échantillons générés sont tous normalisés pour le calcul des regrets. Ainsi, la condition de normalisation $\sum_{i=1}^{n^2}\alpha_i = 1$ sera respectée lors du calcul des valeurs $C_v(x), \forall x \in \mathcal{X}$.

Exemple 4.3. *Reprenons la solution x de l'exemple précédent, cette solution est associée au vecteur de performances $u(x) = (0.3, 0.5, 0.8)$. Supposons que l'on représente les préférences du décideur par une intégrale de Choquet dont la capacité v est définie par le tableau suivant :*

X	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$
$v(X)$	0	0.1	0.2	0.3	0.5	0.5	0.6	1

En utilisant la formule classique de l'intégrale de Choquet (équation 4.17), on évalue x par $C_v(x) = 0.3v(N) + (0.5 - 0.3)v(\{2,3\}) + (0.8 - 0.5)v(\{3\}) = 0.3 + 0.12 + 0.09 = 0.51$.

Utilisons maintenant la formulation de l'équation 4.18 : dans l'espace tricritère, toute capacité v peut être écrite comme une combinaison convexe des capacités $v_1, \ldots, v_9$ donnée par la table 4.1. Ici, les jeux unanimes sont les capacités $v_1, \ldots, v_6$ associées aux ensembles $Y_1 = \{1\}, Y_2 = \{2\}, Y_3 = \{3\}, Y_4 = \{1,2\}, Y_5 = \{1,3\}$ et $Y_6 = \{2,3\}$, et les conjuguées sont $v_7, \ldots, v_9$ associées aux ensembles $Y_7 = \{1,2\}, Y_8 = \{1,3\}$ et $Y_9 = \{2,3\}$.

X	$\emptyset$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$
$v_1(X)$	0	1	0	0	1	1	0	1
$v_2(X)$	0	0	1	0	1	0	1	1
$v_3(X)$	0	0	0	1	0	1	1	1
$v_4(X)$	0	0	0	0	1	0	0	1
$v_5(X)$	0	0	0	0	0	1	0	1
$v_6(X)$	0	0	0	0	0	0	1	1
$v_7(X)$	0	1	1	0	1	1	1	1
$v_8(X)$	0	1	0	1	1	1	1	1
$v_9(X)$	0	0	1	1	1	1	1	1

TABLE 4.1 – Exemple de base de capacités.

Les valeurs $C_{v_i}(x)$ de l'intégrale de Choquet de la solution x pour chacune des capacités $v_1, \ldots, v_9$ sont données par le tableau ci-dessous :

v_i	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
$C_{v_i}(x)$	0.3	0.5	0.8	0.3	0.3	0.5	0.5	0.8	0.8

On peut vérifier facilement que $v = 0.1v_2 + 0.3v_3 + 0.3v_4 + 0.1v_5 + 0.1v_6 + 0.1v_7$. De ce fait, en utilisant l'équation 4.18 on trouve $C_v(x) = 0 \times 0.3 + 0.1 \times 0.5 + 0.3 \times 0.8 + 0.3 \times 0.3 + 0.1 \times 0.3 + 0.1 \times 0.5 + 0.1 \times 0.5 + 0.8 \times 0 + 0.8 \times 0 = 0.51$.

En utilisant les expressions linéaires données par les équations 4.15, 4.16 et 4.18, on peut donc facilement utiliser l'algorithme 4.2 pour estimer la distribution de probabilité associée à l'espace des paramètres d'un OWA général, d'un OWA concave ou d'une intégrale de Choquet 2-additive. Notons que dans le cas de l'intégrale de Choquet 2-additive, la dimension de l'espace des paramètres est augmentée puisque la base de capacités est composée de n^2 éléments, le paramètre à éliciter étant donc de dimension n^2.

4.2.4 Expérimentations numériques

Nous avons implémenté et testé [3] l'algorithme 4.1 pour l'élicitation des paramètres d'un OWA et d'une intégrale de Choquet 2-additive sur des ensembles explicites de solutions générées aléatoirement. Les différentes opérations d'échantillonnage selon une distribution Gaussienne tronquée ont été effectuées en utilisant la librairie R tmvtnorm. Pour pouvoir faire appel à cette librairie, nous avons utilisé la librairie Python rpy2. Tous les résultats donnés sont des moyennes sur l'exécution de l'algorithme 4.1 sur 50 instances générées aléatoirement.

Génération des instances et initialisation des paramètres de l'algorithme. Afin de tester l'algorithme, nous avons généré des instances contenant 100 solutions Pareto optimales évaluées selon 5 critères. Pour chaque instance $\mathcal{X}$, le vecteur de performance d'une solution $x \in \mathcal{X}$ est généré de la manière suivante : un premier vecteur v de dimension 4 est généré uniformément dans $[0,1]^4$, un second vecteur y est ensuite obtenu par $y_i = v_{(i)} - v_{(i-1)}$, pour tout $i \in \{1,\ldots,5\}$, où $v_{(i)}$ est le i^e plus petit élément de v, $v_{(0)} = 0$ et $v_{(5)} = 1$. Ensuite, pour éviter que toutes les solutions soient définies sur le même hyperplan de l'espace des critères, i.e., $\sum_{i=1}^{5} u_i(x) = 1, \forall x \in \mathcal{X}$, on applique la fonction racine carrée à toutes les composantes du vecteur y. On obtient alors $u_i(x) = \sqrt{y_i}, \forall i \in \{1,\ldots,5\}$. La figure 4.4 montre un exemple d'une telle instance dans l'espace bicritère :

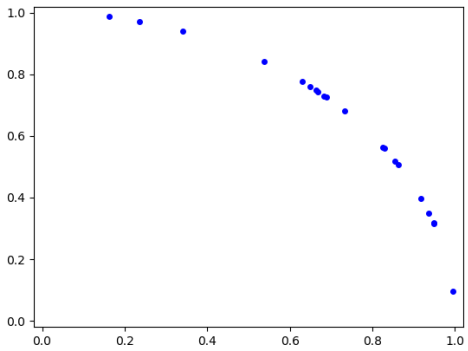


FIGURE 4.4 – Exemple d'instance à 20 solutions évaluées selon 2 critères.

Nous avons fixé un seuil de tolérance $\delta = 0.05$ pour la condition d'arrêt de l'algorithme 4.1. Pour l'élicitation des paramètres d'un OWA, nous avons fixé $\mathcal{N}((10,\ldots,10), 100I_5)$ comme fonction de densité a priori, où I_5 est la matrice identité de dimension 5. Quant à l'élicitation des paramètres de l'intégrale de Choquet, nous avons fixé $\mathcal{N}(\mu, 100I_5)$, où $\mu_i = 10$ si $|Y_i| = 1$ et $\mu_i = 0$ sinon (cf. définition 4.4). De cette manière, la densité a priori est centrée autour d'un vecteur de pondération pour lequel il n'y a aucune interaction entre les différents critères. Après chaque question, un échantillon de 100 vecteurs de pondération est tiré selon la distribution de probabilité courante (ligne 3 de l'algorithme) afin d'effectuer le calcul des regrets espérés. Pour finir, les appels à la fonction *Mise_à_jour*$(p(\omega|r^{(i)}))$ (algorithme 4.2) sont effectués avec $m = 100$ (tailles des échantillons de valeurs z) et $K = 5$ (nombre d'itérations de l'algorithme 4.2).

3. Implémentation en Python. Les tests ont été effectués sur une machine Intel(R) Core(TM) i7-4790 CPU avec 15GB de RAM.

Simulation des interactions avec le décideur. Pour chaque instance, un poids (caché) $\hat{\omega}$ est généré uniformément dans le simplexe afin de simuler les réponses du décideur. La réponse à la question $x^{(i)} \succsim_? y^{(i)}$ est alors générée en fonction du signe de $z^{(i)} = \omega(u(x^{(i)}) - u(y^{(i)})) + \varepsilon^{(i)}$ avec $\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$. La réponse est oui si $z^{(i)} \geq 0$, et non sinon. Afin de tester la tolérance de l'approche aux mauvaises réponses, on effectue des tests pour différentes valeurs de σ : $\sigma = 0$ pour simuler des interactions sans mauvaises réponses, et $\sigma = 0.1$ (respectivement $\sigma = 0.2$) pour simuler des interactions avec mauvaises réponses, ce qui a mené en pratique à 19% (respectivement 26%) de mauvaises réponses pour les instances considérées pour l'élicitation des paramètres d'OWA, et 10% (respectivement 17%) pour l'intégrale de Choquet 2-additive.

Avant de passer à la présentation des résultats expérimentaux, on donne d'abord un exemple d'exécution de l'algorithme 4.1 sur l'exemple donné en introduction :

Exemple 4.1 (suite). *Reprenons l'exemple donné en introduction et supposons que l'on applique l'algorithme 4.1 avec $\delta = 0.02$. Pour rappel, les solutions de $\mathcal{X}$ sont représentées sur la droite de la figure 4.5 :*

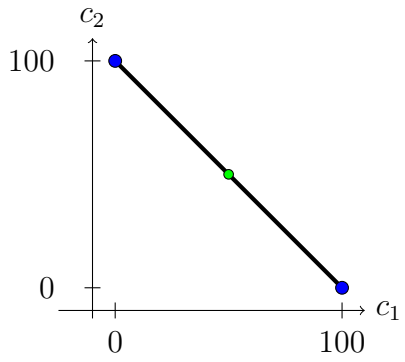


FIGURE 4.5 – Ensemble $\mathcal{X}$ représenté dans l'espace bicritère.

On suppose toujours que les préférences du décideur sont représentées par un OWA *concave de paramètre (caché) $\hat{w} = 0.6$. Initialement le* MMER *vaut $0.05 > \delta$, on demande alors au décideur de comparer la solution ayant la plus petite valeur de* MER *donnée par x^{24} à la solution lui assurant un regret espéré maximum donnée par x^{50}. Notons que le choix de la question n'est pas déterministe, il dépend de l'échantillon S considéré. Le décideur compare alors les valeurs de ces deux solutions :* $\text{OWA}_{\hat{w}}(x^{24}) = 40 + 0.2 \times 24 = 44.8$ *et* $\text{OWA}_{\hat{w}}(x^{50}) = 50$. *Il devrait donc déclarer préférer la solution x^{50} mais supposons que le décideur se trompe et déclare préférer x^{24}. On met alors à jour la densité de probabilité et un nouvel échantillon S est tiré selon cette densité. On trouve alors* $\text{MMER}(\mathcal{X}, S) = 0.04$. *Une seconde question est donc posée au décideur : il doit comparer les solutions x^{11} et x^{50} et donne cette fois une réponse correcte en déclarant préférer x^{50}. Après une nouvelle mise à jour de la densité de probabilité et après le tirage d'un nouvel échantillon on trouve* $\text{MMER}(\mathcal{X}, S) = 0.05 > \delta$. *Quatre questions supplémentaires sont posées avant que la condition d'arrêt ne soit vérifiée : comparaison entre x^{28} et x^0, entre x^{34} et x^0, entre x^{43} et x^0 et finalement entre x^{46} et x^0, le décideur déclarant préférer x^{28}, x^{34},x^{43} et x^{46} à x^0. Finalement, après un total de 6 réponses on obtient* $\text{MMER} = 0.01 < \delta$ *et l'algorithme s'arrête en recommandant la solution* $x^{47} \in \arg\min_{x \in \mathcal{X}} \text{MER}(x, \mathcal{X}, S)$. *Cette solution est évaluée (selon le système de valeurs du décideur) par* $\text{OWA}_w(x^{47}) =$

$40 + 0.2 \times 47 = 49.4$ *et est donc très proche de la vraie solution optimale du décideur, i.e.,* x^{50} *de valeur* $\text{OWA}_w(x^{50}) = 40 + 0.2 \times 50 = 50$.

On observe qu'avec la même erreur commise de la part du décideur, c'est maintenant la solution x^{47} qui est recommandée alors que l'algorithme déterministe recommande x^0 ou x^{100}. On peut voir que le regret réel de la recommandation x^{47} est bien plus petit que le regret réel de x^0 ou x^{100} : $\text{OWA}_{0.6}(x^{50}) - \text{OWA}_{0.6}(x^{47}) = 50 - 49.4 = 0.6$ (qui représente 1.2% de la valeur optimale) alors que $\text{OWA}_0.6(x^{50}) - \text{OWA}_{0.6}(x^0) = \text{OWA}_{0.6}(x^{50}) - \text{OWA}_{0.6}(x^{100}) = 50 - 40 = 10$ (20% de la valeur optimale). On voit alors à travers cet exemple l'intérêt de l'utilisation d'une densité de probabilité pour représenter l'état de connaissance des préférences lorsque des erreurs sont présentes dans les réponses du décideur. Passons maintenant aux résultats expérimentaux.

Analyse des résultats. Dans un premier temps, nous montrons l'évolution de la valeur du MMER au cours de l'exécution de l'algorithme. La figure 4.6 donne, à chaque étape de l'algorithme, la valeur moyenne du MMER sur les 50 instances considérées, en pourcentage du MMER initial :

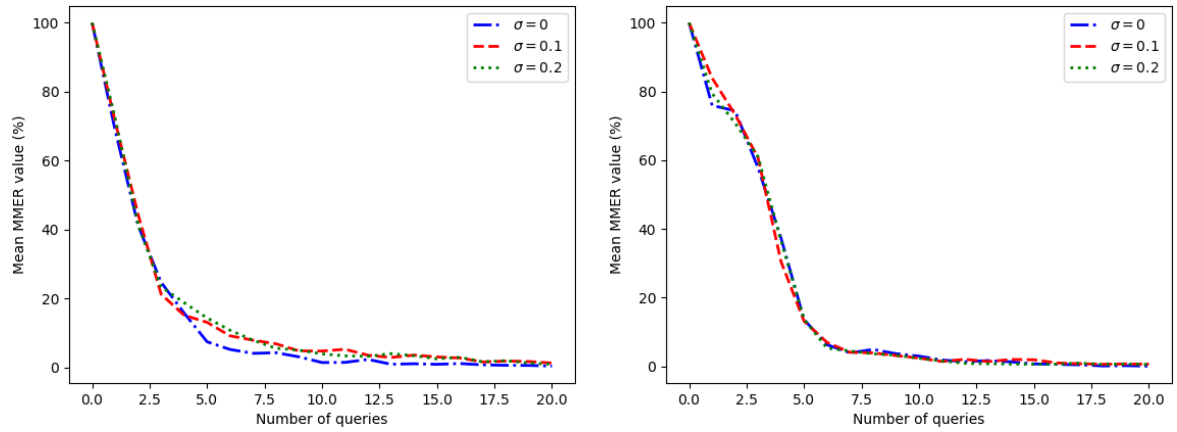


FIGURE 4.6 – Évolution de la valeur moyenne du MMER pour OWA (à gauche) et l'intégrale de Choquet 2-additive (à droite).

On peut voir que la valeur du MMER converge assez rapidement vers 0, même si elle peut parfois augmenter légèrement. On voit effectivement que, pour OWA comme pour Choquet, la valeur moyenne est inférieure à 3% de la valeur initiale après une douzaine de questions. Cette diminution est un effet de l'opération de mise à jour effectuée par l'algorithme 4.2 qui réduit, à chaque mise à jour, la variance de la distribution Gaussienne courante (indépendamment de la présence ou non d'erreurs dans les réponses). Ainsi, la dispersion des vecteurs de pondération de l'échantillon S courant est de plus en plus réduite. Plus les éléments de S sont proches, plus il y a de chances qu'ils correspondent tous (ou en majorité) à la même solution optimale ou à des solutions très proches (en termes de valeur de f_ω). De ce fait, plus les éléments de S sont proches et plus la valeur du MMER est petite (par définition du PER et du MER). En conclusion, le critère d'arrêt de l'algorithme 4.1 se justifie par la convergence empirique de la valeur du MMER vers 0.

On présente maintenant des résultats visant à montrer l'efficacité de l'algorithme 4.1 en termes de qualité de la solution recommandée. Pour cela, on observe le rang réel de

la recommandation courante après chaque question, i.e., le rang de la solution MMER (offrant le plus petit regret espéré maximum) dans le rangement des solutions selon le poids (caché) $\hat{\omega}$ représentant les préférences du décideur. Le rang 0 signifie que la solution est optimale pour $\hat{\omega}$ tandis que le rang 99 signifie que la solution est la pire solution de l'instance pour le paramètre $\hat{\omega}$. La figure 4.7 donne l'évolution du rang réel de la solution recommandée courante, moyenné sur 50 instances :

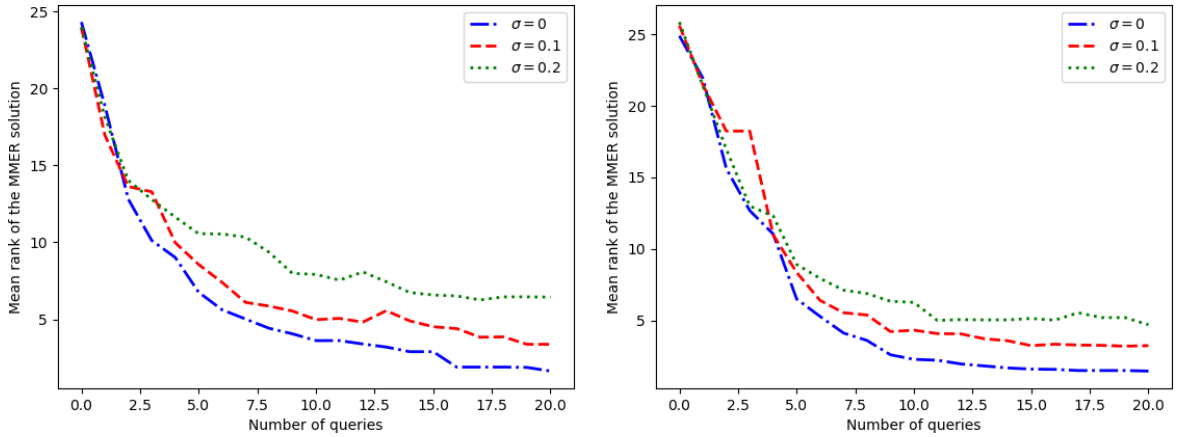


FIGURE 4.7 – Évolution du rang réel moyen de la solution MMER courante pour OWA (à gauche) et pour l'intégrale de Choquet 2-additive (à droite).

On peut voir que la solution recommandée est de très bonne qualité puisqu'elle a un rang d'au plus 7 (sur 100 solutions) quels que soient le taux d'erreur et le modèle de décision considérés. De manière évidente, le taux d'erreur, donné par σ, impacte la qualité de la recommandation. On voit en effet que la solution recommandée est en moyenne optimale (ou presque optimale) lorsque le décideur ne fait pas d'erreurs, mais plus la valeur de σ augmente et plus la qualité de la recommandation se détériore. Néanmoins, le rang moyen de la solution recommandée ne dépasse pas 5, excepté pour OWA pour le taux d'erreurs le plus grand (lorsque $\sigma = 0.2$) ; dans ce cas le rang moyen est d'environ 6.5.

Nous allons maintenant comparer l'efficacité de l'algorithme 4.1 à l'approche déterministe correspondant à l'algorithme 3.1 (qui ne prend pas en compte la possibilité de la présence d'erreurs dans les réponses du décideur et qui ne donne pas au décideur l'opportunité de se contredire). Pour les deux algorithmes, on considère deux critères d'évaluation : le nombre de questions posées nécessaires à la détermination d'une recommandation finale, et la qualité de cette recommandation donnée par son rang réel.

Concernant le nombre de questions, on peut voir sur les courbes de la figure 4.7 que la qualité de la recommandation est relativement bonne à partir d'une douzaine de questions (et ne s'améliore pas significativement au delà), ce qui est similaire à ce que l'on obtient dans le cas de l'algorithme déterministe. Quant à la qualité de la recommandation, l'histogramme de la figure 4.8 montre la distribution des rangs des recommandations faites pour les 50 instances considérées. Ces résultats sont donnés pour OWA avec $\sigma = 0.2$ mais les résultats sont similaires pour l'intégrale de Choquet 2-additive. On peut voir que l'algorithme 4.1 recommande la solution optimale dans 50% des cas tandis que l'algorithme 3.1 ne recommande la solution optimale que dans 36% des cas. On peut

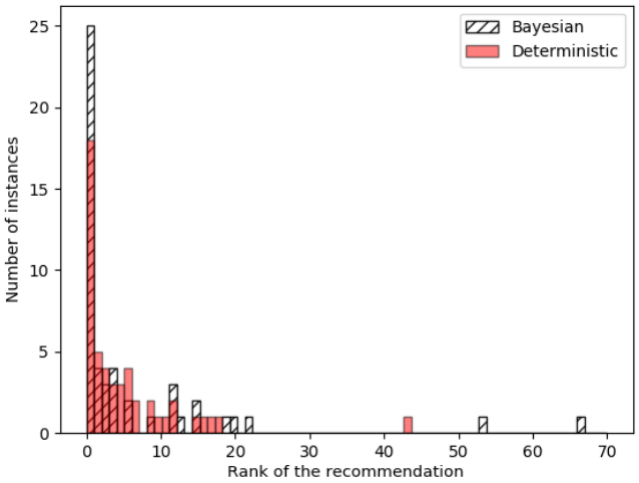


FIGURE 4.8 – Rang réel de la solution recommandée pour OWA.

ensuite remarquer que la dispersion est similaire pour les deux algorithmes sur les instances restantes. Nous verrons dans les expérimentations de la section suivante que la différence entre les deux algorithmes est beaucoup plus marquée lorsque l'on traite des problèmes définis sur domaine combinatoire.

Finalement, concernant le temps de calcul entre deux questions, il est d'environ 5 secondes pour OWA et 35 secondes pour l'intégrale de Choquet 2-additive. Cette différence considérable de temps de calcul s'explique par la taille de l'espace des paramètres. Lorsque les solutions sont évaluées selon 5 critères, la taille de l'espace des paramètres pour OWA est de 5 alors que celle de l'espace des paramètres pour l'intégrale de Choquet 2-additive est de 25 (en considérant la formulation donnée en sous-section 4.2.3) car le nombre de capacités de base est de 25 (15 pour les jeux unanimes et 10 pour les conjuguées).

4.3 Élicitation incrémentale par régression linéaire Bayésienne sur domaine combinatoire

L'approche d'élicitation par régression linéaire Bayésienne de l'algorithme 4.1 ne s'applique pas directement à la résolution de problèmes définis sur domaine combinatoire. Les principaux obstacles sont dus au calcul des valeurs de regrets espérés maximaux MER et du minimax regret espéré MMER. En effet, on peut voir que lorsque l'ensemble des solutions réalisables $\mathcal{X}$ contient un nombre exponentiel de solutions, le calcul des valeurs MER et MMER nécessite un nombre exponentiel de comparaisons par paires (cf. définitions 4.2 et 4.3). De plus, la programmation linéaire n'est pas évidente ici à cause de la fonction $\max\{0, .\}$ dans la définition du regret espéré par paire PER (définition 4.1). Ces deux obstacles sont d'autant plus critiques que les valeurs MER et MMER doivent être calculées à chaque étape de l'algorithme car elles sont utilisées pour déterminer le critère d'arrêt de l'algorithme, une recommandation ou la prochaine question à poser. On propose ici une nouvelle méthode de calcul de ces valeurs permettant de surmonter ces difficultés.

4.3.1 Linéarisation de l'expression du PER par l'introduction de variables binaires

Alors que l'utilisation de la programmation linéaire est plutôt standard pour l'optimisation des regrets (notamment pour les PMR), son utilisation dans le cadre de l'optimisation des regrets *espérés* est plus originale. On propose ici une méthode de calcul des valeurs MER et MMER par programmation linéaire en variables mixtes. Pour simplifier, on considère ici une fonction $f_\omega(x)$ linéaire en $u(x)$, mais l'approche s'adapte à des fonctions d'agrégation non-linéaires pour lesquelles il existe une linéarisation (par exemple, la linéarisation d'OWA [Ogryczak et Śliwiński, 2003]). On suppose de plus que $f_\omega(x) \in [0,1]$. Afin de pouvoir utiliser l'algorithme 4.2 pour la mise à jour de la distribution de probabilité courante, on suppose évidemment que f_ω est linéaire en ω.

Formulation linéaire pour le PER

Étant données deux solutions $x, y \in \mathcal{X}$ et un échantillon S de paramètres ω générés selon la distribution courante $p(\omega)$, le regret espéré par paire est donné par :

$$\mathrm{PER}(x,y,S) = \frac{1}{|S|}\sum_{\omega\in S} \max\{0, f_\omega(y) - f_\omega(x)\}$$

Cette expression n'est pas linéaire du fait de la fonction $\max\{0,.\}$. Afin d'obtenir une expression linéaire de ce regret, on introduit, pour tout $\omega \in S$, une variable binaire b_ω qui joue le rôle de la fonction $\max\{0, f_\omega(y)-f_\omega(x)\}$ et qui prend la valeur 1 si $f_\omega(y)-f_\omega(x) > 0$ et 0 si $f_\omega(y) - f_\omega(x) < 0$. Pour cela, on remplace la fonction $\max\{0, f_\omega(y) - f_\omega(x)\}$ par $b_\omega[f_\omega(y) - f_\omega(x)]$ pour tout paramètre ω de l'échantillon S, et on pose les contraintes suivantes :

$$\begin{cases} b_\omega \leq f_\omega(y) - f_\omega(x) + 1 & \forall \omega \in S \qquad (c_\leq) \\ b_\omega \geq f_\omega(y) - f_\omega(x) & \forall \omega \in S \qquad (c_\geq) \end{cases}$$

Notons que la valeur de b_ω est libre lorsque $f_\omega(y) - f_\omega(x) = 0$ puisque quelle que soit sa valeur le produit $b_\omega[f_\omega(y) - f_\omega(x)]$ vaut 0. La proposition 4.3 montre que l'ajout des variables b_ω et des contraintes $c_\leq$ et $c_\geq$ correspondantes permet bien de calculer le PER de la paire x, y :

Proposition 4.3. *Soient un vecteur de pondération $\omega \in S$, et deux solutions $x \in \mathcal{X}$ et $y \in \mathcal{X}$. Si f_ω est une fonction d'agrégation telle que $f_\omega(z) \in [0,1]$ pour toute solution $z \in \mathcal{X}$ et pour tout vecteur de pondération $\omega \in S$, et si b_ω satisfait les contraintes $(c_\leq)$ and $(c_\geq)$, alors :*

$$\max\{0, f_\omega(y) - f_\omega(x)\} = b_\omega[f_\omega(y) - f_\omega(x)]$$

Démonstration. Notons d_ω la valeur de la différence $f_\omega(y) - f_\omega(x)$ pour tout $\omega \in S$. Notons tout d'abord que $d_\omega \in [0,1], \forall \omega \in S$, car f_ω est telle que $f_\omega(z) \in [0,1], \forall z \in \mathcal{X}$, par hypothèse. Pour tout $\omega \in S$, on peut distinguer trois cas possibles : *Cas 1.* ω est tel que $d_\omega > 0$: la contrainte $(c_\geq)$ devient alors $b_\omega \geq d_\omega > 0$, d'où $b_\omega = 1$. On obtient alors $b_\omega d_\omega = d_\omega \geq 0$. *Cas 2.* ω est tel que $d_\omega < 0$: la contrainte $(c_\leq)$ devient alors $b_\omega \leq d_\omega + 1 < 1$, la variable b_ω prend donc la valeur 0 et on obtient $b_\omega d_\omega = 0$. *Cas 3.* ω est tel que $d_\omega = 0$ alors $b_\omega d_\omega = 0, \forall b_\omega \in \{0,1\}$.

Dans les trois cas, on obtient bien $b_\omega d_\omega = \max\{0, d_\omega\}$. □

En utilisant les variables binaires b_ω, on réécrit le PER de la paire x, y comme un problème de satisfaction de contraintes en variables binaires :

$$\mathrm{PER}(x, y, S) = \sum_{\omega \in S} b_\omega [f_\omega(y) - f_\omega(x)]$$

sous les contraintes :

$$\begin{cases} b_\omega \leq f_\omega(y) - f_\omega(x) + 1 & \forall \omega \in S \qquad (c_\leq) \\ b_\omega \geq f_\omega(y) - f_\omega(x) & \forall \omega \in S \qquad (c_\geq) \\ b_\omega \in \{0, 1\} & \forall \omega \in S \end{cases}$$

Calcul du MER par un programme linéaire en variables mixtes

En utilisant les variables binaires b_ω et les contraintes correspondantes dans la formulation de $\mathrm{MER}(x, \mathcal{X}, S)$, on obtient un système de contraintes linéaires avec une fonction objectif quadratique :

$$\begin{cases} \max_{y \in \mathcal{X}} \sum_{\omega \in S} b_\omega [f_\omega(y) - f_\omega(x)] & \\ b_\omega \leq f_\omega(y) - f_\omega(x) + 1 & \forall \omega \in S \\ b_\omega \geq f_\omega(y) - f_\omega(x) & \forall \omega \in S \\ b_\omega \in \{0, 1\} & \forall \omega \in S \\ y \in \mathcal{X} & \end{cases}$$

En effet, les contraintes $(c_\leq)$ et $(c_\geq)$ sont linéaires car $f_\omega(x)$ est une constante (x est fixée dans le calcul de $\mathrm{MER}(x, \mathcal{X}, S)$) et $f_\omega(y)$ est linéaire en $u(y)$ (par hypothèse sur f_ω). Quant à la fonction objectif, elle est bien quadratique puisque l'expression $b_\omega f_\omega(y)$ contient le produit des deux variables b_ω et y. Afin de linéariser ce programme, on utilise une linéarisation standard du produit d'une variable binaire et d'une variable réelle. Cette linéarisation consiste à introduire des variables réelles positives p_ω, pour tout $\omega \in S$, pour remplacer le terme $b_\omega f_\omega(y)$. Notons que le terme $b_\omega f_\omega(x)$ ne nécessite pas d'être linéarisé puisque la solution x est fixée dans le calcul du MER. Le programme linéaire en variables mixtes obtenu est donc :

$$(P_{\mathrm{MER}}) : \begin{cases} \max \frac{1}{|S|} \sum_{w \in S} [p_w - b_\omega f_\omega(x)] & & \\ b_\omega \leq f_\omega(y) - f_\omega(x) + 1 & \forall \omega \in S & \\ b_\omega \geq f_\omega(y) - f_\omega(x) & \forall \omega \in S & \\ p_w \leq b_\omega & \forall \omega \in S & (c_1) \\ p_w \leq f_\omega(y) & \forall \omega \in S & (c_2) \\ p_w \geq b_\omega + f_\omega(y) - 1 & \forall \omega \in S & (c_3) \\ b_\omega \in \{0, 1\} & \forall \omega \in S & \\ p_w \in \mathbb{R}^+ & \forall \omega \in S & \\ y \in \mathcal{X} & & \end{cases}$$

Il est facile de voir que, pour tout vecteur de pondération $\omega \in S$, si p_ω respecte les contraintes $(c_1), (c_2)$ et (c_3) alors $p_\omega = b_\omega f_\omega(y)$. En effet, si $b_\omega = 0$ alors $p_\omega = 0$ par la contrainte (c_1), et si $b_\omega = 1$ alors $p_\omega = f_\omega(y)$ grâce aux contraintes (c_2) et (c_3) qui deviennent $p_\omega \leq f_\omega(y)$ et $p_\omega \geq 1 + f_\omega(y) - 1 = f_\omega(y)$.

Nous venons donc de définir un programme linéaire en variables mixtes pour le calcul de la valeur du regret espéré maximum d'une solution $x \in \mathcal{X}$. Au total, $2|S|$ variables sont

ajoutées pour la linéarisation de l'expression de la fonction objectif $\frac{1}{|S|}\sum_{w\in S}\max\{0, f_\omega(y) - f_\omega(x)\}$: $|S|$ variables binaires b_ω sont nécessaires à la linéarisation de la fonction $\max\{0, .\}$, et $|S|$ variables réelles p_ω sont ensuite nécessaires à la linéarisation du produit $b_\omega f_\omega(y)$. Afin d'assurer la validité de ces linéarisations, $6|S|$ contraintes liées aux variables b_ω et p_ω sont également ajoutées.

Formulation du calcul du MMER

On rappelle que le minimax regret espéré est donné par l'expression :

$$\text{MMER}(\mathcal{X}, S) = \min_{x\in\mathcal{X}} \max_{y\in\mathcal{X}} \frac{1}{|S|}\sum_{\omega\in S} \max\{0, f_\omega(y) - f_\omega(x)\}$$

On peut tout d'abord linéariser l'objectif $\min_{x\in\mathcal{X}}\max_{y\in\mathcal{X}}$ en utilisant une linéarisation standard d'un *min max* lorsque le *max* est calculé sur un ensemble fini (c'est le cas de $\mathcal{X}$) :

$$\begin{cases} \min t \\ t \geq \frac{1}{|S|}\sum_{w\in S}\max\{0, f_\omega(y) - f_\omega(x)\} \quad \forall y \in \mathcal{X} \qquad (*) \\ x \in \mathcal{X} \\ t \in \mathbb{R}^+ \end{cases}$$

Pour obtenir un programme linéaire, on doit maintenant linéariser les contraintes $(*)$. Pour cela, on remplace la fonction $\max\{0, .\}$ par des variables binaires de manière similaire à la formulation du PER. Ici, nous introduisons une variable binaire b_ω^y pour chaque paire (poids, solution) $\omega, y \in S \times \mathcal{X}$. On obtient alors :

$$\max\{0, f_\omega(y) - f_\omega(x)\} = b_\omega^y(f_\omega(y) - f_\omega(x))$$

pour tout poids $\omega \in S$ et pour toute solution $y \in \mathcal{X}$. Notons que la différence avec la formulation du MER réside dans le fait que, pour ce dernier, la solution x est fixée et la condition $\max\{0, f_\omega(y) - f_\omega(x)\} = b_\omega(f_\omega(y) - f_\omega(x))$ ne doit être vérifiée que pour une instanciation donnée de la variable y (notamment $y \in \arg\max_{z\in\mathcal{X}} \text{PER}(x, z, S)$). De ce fait, une seule variable binaire b_ω (par ω) est nécessaire. Tandis que pour le calcul du MMER, quelle que soit la solution x considérée, la condition $\max\{0, f_\omega(y) - f_\omega(x)\} = b_\omega(f_\omega(y) - f_\omega(x))$ doit être vérifiée pour toute solution $y \in \mathcal{X}$ afin que les contraintes $(*)$ soient toutes vérifiées et que la valeur de t corresponde à un regret espéré maximum (celui de x). En ajoutant les variables b_ω^y on obtient le programme quadratique suivant :

$$(\text{QP}_{\text{MMER}}) : \begin{cases} \min t & \\ t \geq \frac{1}{|S|}\sum_{\omega\in S} b_\omega^y[f_\omega(y) - f_\omega(x)] & \forall y \in \mathcal{X} \\ b_\omega^y \leq f_\omega(y) - f_\omega(x) + 1 & \forall \omega, y \in S \times \mathcal{X} \\ b_\omega^y \geq f_\omega(y) - f_\omega(x) & \forall \omega, y \in S \times \mathcal{X} \\ b_\omega^y \in \{0, 1\} & \forall \omega, y \in S \times \mathcal{X} \\ x \in \mathcal{X} & \\ t \in \mathbb{R} & \end{cases}$$

La proposition suivante établit le fait que ce programme donne bien la valeur du MMER :

Proposition 4.4. *Une solution $x^* \in \mathcal{X}$ qui est optimale pour le programme* $(\text{QP}_{\text{MMER}})$ *est telle que* $\text{MER}(x^*, \mathcal{X}, S) = \text{MMER}(\mathcal{X}, S)$.

Démonstration. On note t^* la valeur optimale du programme ($\mathrm{QP}_{\mathrm{MMER}}$). Nous allons prouver que la valeur t^* correspond à la valeur de $\mathrm{MER}(x^*, \mathcal{X}, S)$ et est égale à la valeur de $\mathrm{MMER}(\mathcal{X}, S)$. Pour une instanciation donnée de la variable x, les contraintes $(*)$ doivent être vérifiées pour *toute* instanciation possible de la variable y. On a donc $t \geq \frac{1}{|S|}\sum_{\omega \in S} b_\omega^y [f_\omega(y) - f_\omega(x)]$ pour tout $y \in \mathcal{X}$. Et donc, par la proposition 4.3 :

$$t \geq \mathrm{PER}(x, y, S), \forall y \in \mathcal{X}$$

on a alors :

$$t \geq \max_y \mathrm{PER}(x, y, S) = \mathrm{MER}(x, \mathcal{X}, S)$$

Ainsi, pour toute instanciation possible de la variable x, la plus petite valeur que la variable t peut prendre est celle du regret espéré maximum de x. Puisque l'objectif est de minimiser t, alors pour tout $x \in \mathcal{X}$ on aura $t = \mathrm{MER}(x, \mathcal{X}, S)$. En particulier, la solution x^* est telle que

$$t^* = \mathrm{MER}(x^*, \mathcal{X}, S)$$

De plus, la minimisation de t implique que la valeur t^* est telle que $t^* \leq \mathrm{MER}(x, \mathcal{X}, S)$, $\forall x \in \mathcal{X}$, On en conclut donc :

$$t^* = \mathrm{MER}(x^*, \mathcal{X}, S) = \mathrm{MMER}(\mathcal{X}, S).$$

□

Le programme quadratique (QP_{MMER}) permet donc bien de calculer la valeur du minimax regret espéré. Afin de linéariser ce programme, les termes quadratiques $b_\omega^y f_\omega(x)$ sont remplacés par $|S| \times |\mathcal{X}|$ variables réelles positives p_ω^y de la même manière que pour le MER :

$$(P_{\mathcal{X}}) : \begin{cases} \min t & \\ t \geq \frac{1}{|S|}\sum_{w \in S} b_\omega^y [f_\omega(y) - p_\omega^y] & \forall y \in \mathcal{X} \\ b_\omega^y \leq f_\omega(y) - f_\omega(x) + 1 & \forall \omega, y \in S \times \mathcal{X} \\ b_\omega^y \geq f_\omega(y) - f_\omega(x) & \forall \omega, y \in S \times \mathcal{X} \\ p_\omega^y \leq b_\omega^y & \forall \omega, y \in S \times \mathcal{X} \\ p_\omega^y \leq f_\omega(x) & \forall \omega, y \in S \times \mathcal{X} \\ p_\omega^y \geq b_\omega^y + f_\omega(x) - 1 & \forall \omega, y \in S \times \mathcal{X} \\ b_\omega^y \in \{0, 1\} & \forall \omega, y \in S \times \mathcal{X} \\ p_\omega^y \in \mathbb{R}^+ & \forall \omega, y \in S \times \mathcal{X} \\ x \in \mathcal{X} & \\ t \in \mathbb{R} & \end{cases}$$

On obtient alors un programme linéaire en variables mixtes défini par : $|S| \times |\mathcal{X}|$ variables binaires b_ω^y, $|S| \times |\mathcal{X}|$ variables réelles positives p_ω^y et $|\mathcal{X}| + 6 \times |S| \times |\mathcal{X}|$ contraintes. $P_{\mathcal{X}}$ comporte donc un nombre exponentiel de variables et de contraintes du fait de la nature combinatoire de l'ensemble des solutions réalisables $\mathcal{X}$. On propose dans la suite un algorithme de génération de contraintes et de variables permettant de surmonter cette difficulté.

4.3.2 Algorithme de calcul du MMER

On introduit ici un algorithme fondé sur la génération de variables et de contraintes pour le calcul du minimax regret espéré $\text{MMER}(\mathcal{X}, S)$. Cet algorithme nous permet également de déterminer la recommandation associée au minimax regret espéré, i.e., $x_S^* \in \arg\min_{x\in\mathcal{X}} \text{MER}(x, \mathcal{X}, S)$, ainsi que son meilleur adversaire défini par $y_S^* \in \arg\max_{y\in\mathcal{X}} \text{PER}(x_S^*, y, S)$.

Nous introduisons d'abord le programme linéaire P_A qui contient un sous-ensemble de variables b_ω^y et p_ω^y de $P_\mathcal{X}$, et un sous-ensemble de contraintes du type $(*)$: étant donné un ensemble de solutions réalisables $A \subseteq \mathcal{X}$, P_A contient uniquement les variables et contraintes liées aux solutions de A. Ce programme nous permet de calculer le minimax regret espéré $\text{MMER}_A(\mathcal{X}, S)$ que l'on définit par :

Définition 4.5. *Soit S un échantillon de vecteurs de pondérations généré selon la densité courante p. Le minimax regret espéré de $\mathcal{X}$ par rapport à l'ensemble d'adversaires A est défini par :*

$$\text{MMER}_A(\mathcal{X}, S) = \min_{x\in\mathcal{X}} \text{MER}(x, A, S) = \min_{x\in\mathcal{X}} \max_{y\in A} \text{PER}(x, y, S)$$

Autrement dit, on ne considère que l'ensemble des solutions de A comme adversaires potentiels dans le calcul du regret espéré maximum d'une solution $x \in \mathcal{X}$. Plus formellement, P_A s'écrit :

$$(P_A) : \begin{cases} \min t & \\ t \geq \frac{1}{|S|} \sum_{w\in S} b_\omega^y [f_\omega(y) - p_\omega^y] & \forall y \in A \\ b_\omega^y \leq f_\omega(y) - f_\omega(x) + 1 & \forall w, y \in S \times A \\ b_\omega^y \geq f_\omega(y) - f_\omega(x) & \forall w, y \in S \times A \\ p_\omega^y \leq b_\omega^y & \forall w, y \in S \times A \\ p_\omega^y \leq f_\omega(x) & \forall w, y \in S \times A \\ p_\omega^y \geq b_\omega^y + f_\omega(x) - 1 & \forall w, y \in S \times A \\ b_\omega^y \in \{0, 1\} & \forall w, y \in S \times A \\ p_\omega^y \in \mathbb{R}^+ & \forall w, y \in S \times A \\ x \in \mathcal{X} & \\ t \in \mathbb{R} & \end{cases}$$

Ce programme linéaire comporte $|S| \times |A|$ variables binaires b_ω^y, $|S| \times |A|$ variables réelles positives p_ω^y et $|A| + 6 \times |S| \times |A|$ contraintes liées à ces variables.

Notons que lorsque $A = \mathcal{X}$ le programme (P_A) est équivalent à $P_\mathcal{X}$. Lorsqu'en revanche $A \subset \mathcal{X}$, la valeur retournée par ce programme est une approximation de la valeur du minimax regret espéré, elle constitue en effet une borne inférieure de la valeur $\text{MMER}(\mathcal{X}, S)$. On a :

$$\text{MMER}_A(\mathcal{X}, S) = \min_{x\in\mathcal{X}} \max_{y\in A} \text{PER}(x, y, S) \leq \min_{x\in\mathcal{X}} \max_{y\in\mathcal{X}} \text{PER}(x, y, S) = \text{MMER}(\mathcal{X}, S) \tag{4.19}$$

Plus l'ensemble A contient de solutions, plus la valeur $\text{MMER}_A(\mathcal{X}, S)$ peut être proche de la valeur du minimax regret espéré $\text{MMER}(\mathcal{X}, S)$ puisque, par définition, la valeur de $\text{MMER}_A(\mathcal{X}, S)$ est croissante avec A. On propose alors un algorithme de calcul du

MMER qui consiste à ajouter progressivement des solutions de $\mathcal{X} \setminus A$ dans A pour obtenir des approximations de plus en plus précises du MMER. L'ajout de solutions s'arrête lorsque l'on peut vérifier $\text{MMER}_A(\mathcal{X}, S) = \text{MMER}(\mathcal{X}, S)$. Cette condition constitue donc un critère d'arrêt de l'algorithme de génération de contraintes et de variables. Avant de montrer comment déterminer si elle est vérifiée, nous allons présenter comment déterminer une solution à ajouter à A lorsque la condition n'est pas vérifiée, i.e., lorsque $\text{MMER}_A(\mathcal{X}, S) < \text{MMER}(\mathcal{X}, S)$.

Notons que l'ajout d'une solution dans A entraîne une charge de calcul plus importante car cela conduit à l'introduction de $2|S|$ variables additionnelles ($|S|$ variables binaires et $|S|$ variables réelles positives) et de $6|S|$ contraintes supplémentaires. De ce fait, il est nécessaire de n'ajouter à A que des solutions susceptibles d'améliorer l'estimation du MMER afin de minimiser le nombre total de contraintes et de variables ajoutées. On propose alors un choix de solution à ajouter fondée sur la proposition suivante :

Proposition 4.5. *Soit $x_A \in \mathcal{X}$ une solution telle que $x_A \in \arg\min_{x \in \mathcal{X}} \text{MER}(x, A, S)$. Si le minimax regret espéré par rapport à A est tel que $\text{MMER}_A(\mathcal{X}, S) < \text{MMER}(\mathcal{X}, S)$ alors la solution x_A viole au moins une contrainte du type $(*)$ dans $P_\mathcal{X}$.*

Démonstration. Par l'absurde. Supposons que la solution x_A ne viole aucune contrainte du type $(*)$ dans $P_\mathcal{X}$. Notons t_A la valeur optimale du programme (P_A). Cette valeur correspond donc à :

$$t_A = \text{MMER}_A(\mathcal{X}, S) = \text{MER}(x_A, A, S) = \text{PER}(x_A, y_A, S)$$

pour une solution donnée $y_A \in A$. Puisque x_A respecte toutes les contraintes du type $(*)$, alors on a :

$$t_A \geq \text{PER}(x_A, y, S), \forall y \in \mathcal{X}$$

Autrement dit $\text{PER}(x_A, y_A, S) \geq \text{PER}(x_A, y, S)$ pour toute solution $y \in \mathcal{X}$. De ce fait, et par définition du regret espéré maximum :

$$t_A = \text{MER}(x_A, \mathcal{X}, S)$$

Ainsi, $\text{MMER}_A(\mathcal{X}, S) < \text{MMER}(\mathcal{X}, S)$ signifie $\text{MER}(x_A, \mathcal{X}, S) < \text{MMER}(\mathcal{X}, S)$ ce qui est en contradiction avec la définition du MMER. On en conclut alors que x_A viole au moins une contrainte du type $(*)$ dans $(P_\mathcal{X})$. □

Cette proposition nous permet de définir un choix pertinent de solution à ajouter à l'ensemble A : nous allons déterminer la contrainte $(*)$ la plus violée par x_A afin d'améliorer au mieux l'estimation donnée par MMER_A. Les contraintes $(*)$ violées par x_A s'écrivent sous la forme $t_A \geq \frac{1}{|S|} \sum_{\omega \in S} \max\{0, f_\omega(y) - f_\omega(x_A)\}$ avec $y \in \mathcal{X} \setminus A$. De manière évidente, ces contraintes correspondent toutes à des variables $y \in \mathcal{X} \setminus A$ telles que :

$$\frac{1}{|S|} \sum_{\omega \in S} \max\{0, f_\omega(y) - f_\omega(x_A)\} \geq \text{MER}(x_A, A, S) \qquad (a)$$

On peut donc définir la contrainte la plus violée par x_A par la contrainte impliquant la variable $\tilde{y}$ donnée par

$$\tilde{y} \in \arg\max_{y \in \mathcal{X} \setminus A} \frac{1}{|S|} \sum_{\omega \in S} \max\{0, f_\omega(y) - f_\omega(x_A)\} \qquad (b)$$

Les inégalités (a) et (b) nous mènent à la solution :

$$\tilde{y} \in \arg\max_{y \in \mathcal{X}} \frac{1}{|S|} \sum_{\omega \in S} \max\{0, f_\omega(y) - f_\omega(x_A)\} = \arg\max_{y \in \mathcal{X}} \mathrm{PER}(x_A, y, S)$$

c'est-à-dire la solution assurant un regret espéré maximum à x_A, on peut la calculer à l'aide du programme (P_{MER}).

L'algorithme de génération de contraintes et de variables consiste alors à alterner des résolutions de (P_A) et de (P_{MER}) en ajoutant les solutions $\tilde{y}$ à l'ensemble A jusqu'à pouvoir montrer que $\mathrm{MMER}_A(\mathcal{X}, S) = \mathrm{MMER}(\mathcal{X}, S)$. L'approche est décrite par l'algorithme 4.3 :

Algorithme 4.3 : Calcul_MMER$(\mathcal{X}, A, S)$

Entrées : $\mathcal{X}$: ensemble de solutions définies par des contraintes linéaires ; $A \subseteq \mathcal{X}$: sous-ensemble d'adversaires ; S : échantillon de vecteurs de pondération.

```
ỹ ← null ;
répéter
    si ỹ ≠ null alors
        A ← A ∪ {ỹ} ;
    fin
    (mmer_A, x_A) ← MMER_A(X, S) ; (en utilisant le programme (P_A))
    (mer_x_A, ỹ) ← MER(x_A, X, S) ; (en utilisant le programme (P_MER))
jusqu'à ỹ ∈ A;
retourner mmer_A, x_A, ỹ
```

Dans cet algorithme, on note x_A (respectivement $\tilde{y}$) la solution optimale obtenue en résolvant (P_A) (respectivement (P_{MER}) pour $x = x_A$). L'algorithme commence avec un ensemble de solutions réalisables A (l'initialisation de A peut être faite de manière arbitraire, voir la sous-section 4.3.4 pour l'initialisation que l'on propose), et ajoute progressivement les adversaires $\tilde{y}$ de x_A à A. L'algorithme s'arrête lorsque P_{MER} retourne une solution $\tilde{y}$ qui existe déjà dans A car dans ce cas $\mathrm{MMER}_A(\mathcal{X}, S) = \mathrm{MMER}(\mathcal{X}, S)$ (proposition 4.6). Par abus de notations $\mathrm{MMER}_A(\mathcal{X}, S)$ est vue comme une fonction qui retourne le couple constitué de la valeur optimale $mmer_A$ de (P_A) et de la solution optimale correspondante x_A. De manière similaire, $\mathrm{MER}(x_A, \mathcal{X}, S)$ est vue comme une fonction qui retourne le couple constitué de la valeur optimale mer_x_A de (P_{MER}) et de la solution optimale correspondante $\tilde{y}$. À la fin de l'algorithme, $mmer_A$ correspond à $\mathrm{MMER}(\mathcal{X}, S)$, la paire de solutions $x_A, \tilde{y}$ correspond à la paire x_S^*, y_S^* donnée par l'adaptation de la stratégie CSS (cf. sous-section 4.2.1).

Proposition 4.6. *L'algorithme 4.3 termine et retourne la valeur du* MMER *sur* $\mathcal{X}$ *pour l'échantillon* S*, une solution* x_A *telle que* $\mathrm{MER}(x_A, \mathcal{X}, S) = \mathrm{MMER}(\mathcal{X}, S)$ *ainsi que le meilleur adversaire de* x_A *:* $\tilde{y} \in \arg\max_{y \in \mathcal{X}} \mathrm{PER}(x_A, y, S)$.

Démonstration. Tout d'abord, il est facile de voir que l'algorithme 4.3 termine toujours. En effet, à chaque étape de l'algorithme, si la condition d'arrêt n'est pas vérifiée alors une nouvelle solution $\tilde{y} \notin A$ est ajoutée à A et une nouvelle itération est exécutée. Dans le pire cas, toutes les solutions de $\mathcal{X}$ sont ajoutées à A et la condition d'arrêt est vérifiée de manière triviale après $|\mathcal{X}|$ itérations.

On prouve maintenant la validité de l'algorithme, i.e., $\text{MMER}_A(\mathcal{X}, S) = \text{MMER}(\mathcal{X}, S)$ si $\tilde{y} \in A$. Supposons que $A \subsetneq \mathcal{X}$ (si $A = \mathcal{X}$ alors l'égalité $\text{MMER}_A(\mathcal{X}, S) = \text{MMER}(\mathcal{X}, S)$ est vraie de manière triviale). D'une part, la condition

$$mmer_A \leq mer\_x_A \tag{4.20}$$

est vérifiée à chaque étape de l'algorithme car $mmer_A = \text{MER}(x_A, A, S) \leq \text{MER}(x_A, \mathcal{X}, S)$ puisque $A \subsetneq \mathcal{X}$. Et d'autre part, si $\tilde{y} \in A$ alors la contrainte $t \geq \frac{1}{|S|} \sum_{w \in S} [f_\omega(\tilde{y}) - f_\omega(x_A)]$ est vérifiée pour $t = mmer_A$. Autrement dit, $mmer_A \geq \text{PER}(x_A, \tilde{y}, S)$. Comme $\text{PER}(x_A, \tilde{y}, S) = \text{MER}(x_A, \mathcal{X}, S)$ par définition de $\tilde{y}$, alors :

$$mmer_A \geq \text{MER}(x_A, \mathcal{X}, S) = mer\_x_A \tag{4.21}$$

Par les inégalités (4.20) et (4.21) on obtient

$$mmer_A = mer\_x_A \tag{4.22}$$

Pour finir, la valeur $mmer_A$ minimise la quantité $\frac{1}{|S|} \sum_{\omega \in S} b_\omega [f_\omega(\tilde{y}) - f_\omega(x)]$ pour tout $x \in \mathcal{X}$: elle minimise donc $\text{PER}(x, \tilde{y}, S)$ pour tout $x \in \mathcal{X}$. De ce fait :

$$\begin{array}{rrcll} & mmer_A & \leq & \text{PER}(x, \tilde{y}, S) & \forall x \in \mathcal{X} \\ \Leftrightarrow & mer\_x_A & \leq & \text{PER}(x, \tilde{y}, S) & \forall x \in \mathcal{X} \end{array}$$

du fait de l'égalité entre $mmer_A$ et mer_x_A. On sait ensuite que pour toute solution $x \in \mathcal{X}$ et pour toute solution $y \in \mathcal{X}$ on a $\text{PER}(x, y, S) \leq \text{MER}(x, \mathcal{X}, S)$ par définition du regret espéré maximum. L'inégalité est donc en particulier vraie pour $y = \tilde{y}$. Donc :

$$mer\_x_A \leq \text{MER}(x, \mathcal{X}, S) \quad \forall x \in \mathcal{X}$$

Par définition du minimax regret espéré, si l'inégalité $mer\_x_A \leq \text{MER}(x, \mathcal{X}, S)$ est vérifiée pour tout $x \in \mathcal{X}$ alors :

$$mer\_x_A \leq \text{MMER}(\mathcal{X}, S)$$

On en déduit par définition de mer_x_A et du MMER que

$$mer\_x_A = \text{MMER}(\mathcal{X}, S) \tag{4.23}$$

Finalement, par les égalités (4.22) et (4.23) on obtient :

$$\text{MMER}_A(\mathcal{X}, S) = mmer_A = \text{MMER}(\mathcal{X}, S)$$

□

L'algorithme 4.3 permet donc bien de calculer le minimax regret espéré (pour un échantillon S donné) de manière exacte. À chaque fois qu'une question ou qu'une recommandation est nécessaire, on utilise cet algorithme afin d'appliquer l'adaptation de la stratégie CSS. Notons que la charge de calcul induite par la résolution des différents programmes linéaires en variables mixtes dépend fortement de la taille des échantillons considérés. En effet, les nombres de variables et de contraintes additionnelles des programmes (P_A) et (P_{MER}) augmentent linéairement avec la taille de l'échantillon S. On montre dans la suite comment approcher la valeur $\text{MMER}(\mathcal{X}, S)$ pour réduire le temps d'exécution de l'algorithme tout en gardant un calcul efficace.

4.3.3 Amélioration du temps de calcul des regrets par *clustering*

Les regrets espérés étant calculés à chaque question ou recommandation, il est donc nécessaire de s'assurer que l'on dispose d'une méthode de calcul rapide et efficace pour garantir au décideur un temps d'attente entre deux questions le plus petit possible. Afin de réduire le temps de calcul induit par l'algorithme 4.3, on propose ici de réduire le nombre de variables et de contraintes des programmes (P_{MER}) et (P_A) en appliquant une opération de clustering sur l'échantillon S. Cette opération consiste à partitionner les éléments de S en un ensemble C de *clusters* (ou paquets) d'éléments proches. L'idée est que lorsque des vecteurs de pondérations sont suffisamment proches, ils représentent des préférences proches. Ainsi, au lieu de considérer la totalité des $|S|$ éléments de l'échantillon dans la définition des programmes (P_{MER}) et (P_A), on ne considérera qu'un ensemble réduit de $|C| < |S|$ *représentants* des clusters. Notons que différents algorithmes de clustering existent dans la littérature, on utilise ici l'algorithme de *K-moyennes* Bishop [2006].

Exemple 4.4. *On considère un échantillon de* 200 *vecteurs générés uniformément dans l'espace* $\mathcal{W} = \{\omega \in \mathbb{R}^2_+ | \omega_1 + \omega_2 \leq 1\}$. *La partition de cet échantillon en* 5 *clusters est donné sur la figure 4.9. Chaque cluster est représenté par des points d'une même couleur et d'une même forme, et son représentant est donné par un point de plus grande taille que les autres éléments.*

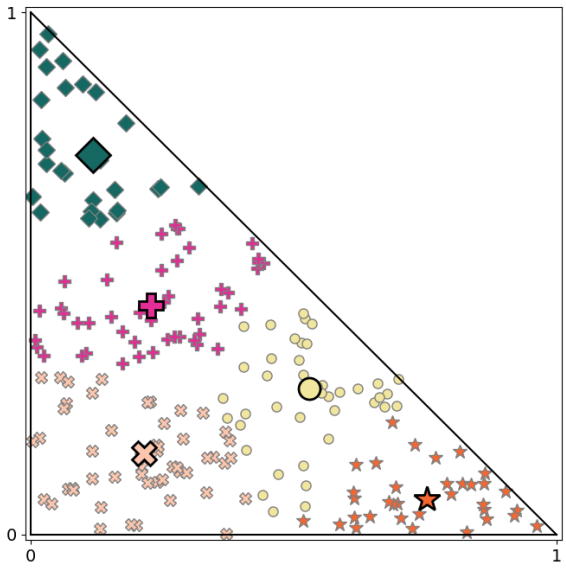


FIGURE 4.9 – Exemple de clustering dans le simplexe avec 5 clusters.

Étant donné un ensemble C de représentants des clusters associés à un échantillon S, on peut calculer une approximation des regrets espérés en remplaçant les éléments de S par ceux de C dans la définition de ces regrets. On obtient :

$$\begin{array}{lcl} \text{PER}(x, y, C) & = & \sum_{c \in C} \rho_c \max\{0, f_c(y) - f_c(x)\} \\ \text{MER}(x, \mathcal{X}, C) & = & \max_{y \in \mathcal{X}} \text{PER}(x, y, C) \\ \text{MMER}(\mathcal{X}, C) & = & \min_{x \in \mathcal{X}} \text{MER}(x, \mathcal{X}, C) \end{array}$$

où ρ_c est le poids de chaque cluster $c \in C$ et représente la proportion d'éléments de S qui appartiennent au cluster c.

4.3.4 Algorithme d'élicitation incrémentale des préférences

Le principe général de l'algorithme est le même que dans le cas de la résolution de problèmes définis sur domaines explicites : on associe une densité de probabilité a priori à $\mathcal{W}$, puis on alterne calcul des regrets en utilisant l'algorithme 4.3 et mise à jour de la densité de probabilité en utilisant l'algorithme 4.2. La méthode est détaillée dans l'algorithme 4.4 :

Algorithme 4.4 :

Entrées : $\mathcal{X}$: ensemble de solutions défini explicitement ; f_ω : fonction d'agrégation de paramètre ω ; $p(\omega)$: fonction de densité a priori ; T : nombre de questions maximum.

1 $i \leftarrow 1$;
2 **répéter**
3 $\quad S \leftarrow$ échantillon selon $p(\omega)$;
4 $\quad C \leftarrow$ centres des clusters de S normalisés ;
5 $\quad A \leftarrow \{\arg\max_{x \in \mathcal{X}} f_\omega(x) | \omega \in C\}$;
6 $\quad (mmer, x^{(i)}, y^{(i)}) \leftarrow$ Calcul_MMER$(\mathcal{X}, A, C)$; (algorithme 4.3)
7 $\quad$ Demander au décideur s'il préfère $x^{(i)}$ ou $y^{(i)}$;
8 $\quad r^{(i)} \leftarrow 1$ si la réponse est $x^{(i)} \succsim y^{(i)}$ et 0 sinon ;
9 $\quad p(w) \leftarrow$ Mise_à_jour$(p(\omega | r^{(i)}))$; (algorithme 4.2)
10 $\quad i \leftarrow i + 1$;
11 **jusqu'à** *stabilisation de la valeur mmer* ou $i > T$;
12 **retourner** $x^* \in \arg\min_{x \in \mathcal{X}} \text{MER}(x, \mathcal{X}, C)$

Dans cet algorithme, l'ensemble A d'adversaires initial est choisi heuristiquement comme l'ensemble de solutions f_ω-optimales pour tout $\omega \in C$ (ligne 5 de l'algorithme). L'ensemble A peut être choisi d'une autre manière sans impacter le résultat de la proposition 4.6.

4.3.5 Expérimentations numériques

Afin de tester l'efficacité de l'algorithme 4.4 nous l'avons implémenté et testé [4] sur différentes instances aléatoires du problème de sac à dos multi-agents ainsi que sur des instances aléatoires du problème d'affectation de ressources multicritère. On considère ici que les préférences du décideur sont représentées par une somme pondérée WS_w (cf. définition 1.9) dont on élicitera le vecteur de pondération w défini dans $W = \{w \in \mathbb{R}^n_+ | \sum_{i=1}^n w_i = 1\}$. Les différents programmes linéaires sont résolus en utilisant la librairie gurobipy de Python. Et les opérations de clustering ont été effectuées en utilisant la librairie sklearn de Python qui implémente l'algorithme de K-moyennes.

4. Implémentation en Python. Caractéristiques de la machine : Intel(R) Core(TM) i7-4790 CPU avec 15 GB de RAM.

Problème du sac à dos multi-agents. Le problème du sac à dos multi-agents que l'on considère ici est le même que celui que l'on considérait dans le chapitre 2 (tel que défini dans la partie expérimentale de la sous-section 2.3.4). On suppose ici que les préférences du décideur sont représentées par le paramètre $\hat{w}$ de l'opérateur WS, ainsi l'objectif est de trouver une solution x telle que la valeur $\mathrm{WS}_{\hat{w}}(x)$ est la plus grande possible. Dans notre contexte où les préférences ne sont pas précisément connues, $\hat{w}$ n'est évidemment pas initialement connu mais on le sait défini dans W. On l'élicitera de manière incrémentale à l'aide de l'algorithme 4.4.

Pour effectuer nos tests, nous avons généré des instances comprenant 5 agents et 100 objets de la manière suivante : chaque objet $k \in \{1, \ldots, 100\}$ est caractérisé par un poids β_k tiré uniformément dans $\{1, \ldots, 20\}$. La capacité totale du sac à dos C est définie par $C = \frac{1}{2} \sum_{k=1}^{100} \beta_k$ afin d'obtenir des instances difficiles à résoudre. Enfin, les valeurs d'utilité $u_k^i, k \in \{1, \ldots, 100\}, i \in \{1, \ldots, 5\}$ des agents pour les objets sont tirées aléatoirement dans $[0, \frac{1}{100}]$ afin de garantir que la condition $f_w(x) \in [0, 1], \forall x \in \mathcal{X}, \forall w \in W$ soit vérifiée. On rappelle que cette hypothèse garantit la validité des programmes linéaires à variables mixtes permettant de calculer les regrets espérés.

Problème d'affectation de ressources multicritère. Le problème d'affectation de ressources *partageables* que nous considérons ici se définit de la manière suivante : étant donné un ensemble de n critères, un ensemble de m agents, un ensemble de $r < m$ ressources et une borne b sur le nombre d'agents à qui une ressource peut être affectée, on cherche à trouver une affectation des ressources aux différents agents de manière à respecter la capacité des ressources (la borne b) et à minimiser les coûts induits par cette affectation. Plus formellement, l'ensemble $\mathcal{X}$ des solutions réalisables est caractérisé par une matrice binaire X de taille $m \times r$ dont chaque composante x_{ij} est une variable de décision qui prend la valeur 1 si l'agent i est affecté à la ressource j, et 0 sinon. Plus précisément, $\mathcal{X}$ est défini par :

$$x \in \mathcal{X} \Leftrightarrow \begin{cases} \sum_{j=1}^{r} x_{ij} = 1 & \forall i \in \{1, \ldots, m\} \\ \sum_{i=1}^{m} x_{ij} \leq b & \forall j \in \{1, \ldots, r\} \\ x_{ij} \in \{0, 1\} \end{cases}$$

où la contrainte $\sum_{j=1}^{r} x_{ij} = 1$ traduit le fait que l'agent $i \in \{1, \ldots, m\}$ se voit affecté exactement une ressource, tandis que $\sum_{i=1}^{m} x_{ij} \leq b$ exprime la contrainte de capacité de la ressource $j \in \{1, \ldots, r\}$. Une affectation réalisable $x \in \mathcal{X}$ est associée au vecteur de coûts $c(x)$, où la k^e composante $c_k(x)$ donne le coût de x selon le critère k et est définie par $c_k(x) = \sum_{i=1}^{m} \sum_{j=1}^{r} c_{ij}^k x_{ij}$, où c_{ij}^k est le coût de l'affectation de la ressource j à l'agent i selon le critère k. Les préférences du décideur étant modélisées par une somme pondérée, une solution $x \in \mathcal{X}$ est évaluée par $\mathrm{WS}_{\hat{w}}(x) = \sum_k \hat{w}_k c_k(x)$, où $\hat{w}$ est le paramètre caractérisant les préférences du décideur. L'objectif est donc de trouver une solution réalisable x telle que la valeur $\mathrm{WS}_{\hat{w}}(x)$ est la plus petite possible. On se place ici dans le cadre où les préférences ne sont pas connues de manière précise : le vecteur de pondération $\hat{w}$ est initialement inconnu mais on le sait défini dans W. On élicitera sa valeur à l'aide de l'algorithme 4.4.

Exemple 4.5. *On considère l'instance du problème d'affectation donnée par le graphe de la figure 4.10. Il y a 4 agents $\{a_1, \ldots, a_4\}$, 2 ressources $\{s_1, s_2\}$, 2 critères et une limite de capacité des ressources fixée à $b = 3$:*

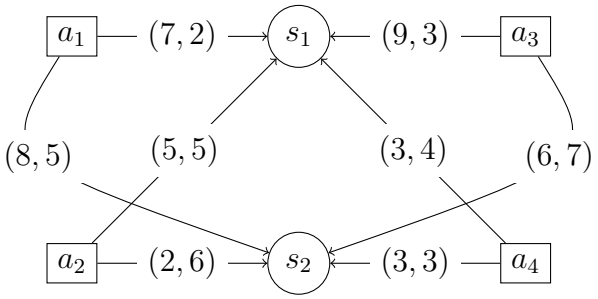


FIGURE 4.10 – Exemple de graphe pour le problème d'affectation de ressources bi-critère.

Dans cette instance, une solution réalisable est donnée par la solution x constituée des arcs $(a_1, s_1), (a_2, s_1), (a_3, s_1)$ et (a_4, s_2) et associée au vecteur de coûts $c(x) = (7 + 5 + 9 + 3, 2 + 5 + 3 + 3) = (24, 13)$. Pour un vecteur de pondération $w \in \mathbb{R}^2_+$ tel que $w_1 + w_2 = 1$, le score de x est donné par $\mathrm{WS}_w(x) = 24w_1 + 13w_2$ (à minimiser).

Pour les tests numériques, nous avons généré des instances comprenant $n = 5$ critères, $m = 50$ agents, $r = 5$ ressources et une limite de capacité de $b = 15$. Les valeurs de coûts $c^k_{ij}, i \in \{1, \ldots, 50\}, j \in \{1, \ldots, 5\}, k \in \{1, \ldots, 5\}$, sont générées uniformément dans $[0, 20]$, puis normalisées (les valeurs c^k_{ij} sont divisées par $\sum_i \sum_j c^k_{ij}$) afin de garantir $f_w(x) \in [0, 1], \forall x \in \mathcal{X}, \forall w \in W$ (pour garantir la validité de l'algorithme de calcul des regrets espérés).

Initialisation des paramètres de l'algorithme La densité de probabilité a priori de l'algorithme 4.4 est fixée à $\mathcal{N}((10, \ldots 10)^T, 100I_5)$, où I_5 est la matrice identité 5×5, ainsi la densité est relativement plate. À chaque étape de l'algorithme, un nouvel échantillon S de 100 vecteurs de pondération est généré selon la distribution de probabilité courante. Les vecteurs de S sont ensuite regroupés selon 20 clusters et le représentant de chaque cluster est normalisé (à 1). Enfin, l'algorithme est stoppé à 15 questions (lorsque le critère d'arrêt ne se déclenche pas plus tôt).

Simulation des interactions avec le décideur. Pour chaque instance, un poids (caché) $\hat{w}$ est généré dans le simplexe afin de simuler les réponses du décideur. Nous avons généré deux types de vecteurs de poids cachés :

1. des vecteurs générés uniformément dans W, notés w_u,
2. des vecteurs déséquilibrés générés uniformément parmi les points extrêmes de W, notés w_{ext}.

Ainsi, on teste l'efficacité de l'algorithme 4.4 dans deux contextes différents : lorsque la recommandation initiale est bonne (en moyenne) et lorsqu'elle est relativement mauvaise (en moyenne). En effet, lorsque le vecteur $\hat{w}$ est tiré uniformément dans le simplexe, une grande partie des vecteurs générés sont situés autour du centre du simplexe. Or, la moyenne de la densité a priori (donnée par $\mathcal{N}((10, \ldots 10)^T, 100I_5)$) favorise les vecteurs de pondération qui se situent au centre du simplexe et donne donc une bonne heuristique pour la détermination d'une bonne recommandation initiale. À l'inverse, cette heuristique est, en moyenne, très mauvaise lorsque le poids caché est du type w_{ext}.

Étant donné un poids caché $\hat{w}$, la réponse à la question "$x^{(i)} \succsim y^{(i)}$?" est générée en fonction du signe de $z^{(i)} = \hat{w}(u(x^{(i)}) - u(y^{(i)})) + \varepsilon^{(i)}$ avec $\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$. La réponse est oui si $z^{(i)} \geq 0$ et non sinon. Afin de tester la tolérance de l'approche aux mauvaises réponses, on effectue des tests pour différentes valeurs de σ : 0, 0.1 et 0.2. Notons que fixer $\sigma = 0$ mène à 0 fausse réponse. Quant aux valeurs strictement positives de σ : pour les instances du problème du sac à dos (respectivement d'affectation de ressources), fixer $\sigma = 0.1$ a mené à 30% (respectivement 24%) de fausses réponses dans le cas des poids w_u et à 16% (respectivement 14%) de fausses réponses dans le cas des poids w_{ext}. Et la valeur $\sigma = 0.2$ a mené à 39% (respectivement 31%) de fausses réponses dans le cas des poids w_u et à 24% (respectivement 21%) de fausses réponses dans le cas des poids w_{ext}.

Avant de présenter les résultats obtenus, nous donnons un exemple montrant la convergence des échantillons générés durant le déroulement de l'algorithme 4.4 (ligne 3) vers le poids réel (caché) modélisant les préférences du décideur :

Exemple 4.6. *Nous avons appliqué l'algorithme 4.4 à une instance aléatoire du problème du sac à dos multi-agents avec* 3 *agents et* 100 *objets. On suppose que les préférences du décideur sont modélisées par une somme pondérée de paramètre* $\hat{w} = (0, 1, 0)$. *Dans cet exemple, la variance du bruit Gaussien a été fixée à* $\sigma = 0.02$, *ce qui a mené à* 20% *d'erreurs dans les réponses. La figure 4.11 montre la dispersion des poids de l'échantillon* S *lors de* 3 *étapes de l'exécution de l'algorithme : avant de commencer la procédure d'élicitation (figure 4.11(a)), après avoir posé* 3 *questions (figure 4.11(b)) et après avoir posé* 10 *questions (figure 4.11(c)). On suppose ici que les poids sont normalisés ; on peut donc représenter* w *en omettant la composante* w_3 *puisque* $w_3 = 1 - w_1 - w_2$.

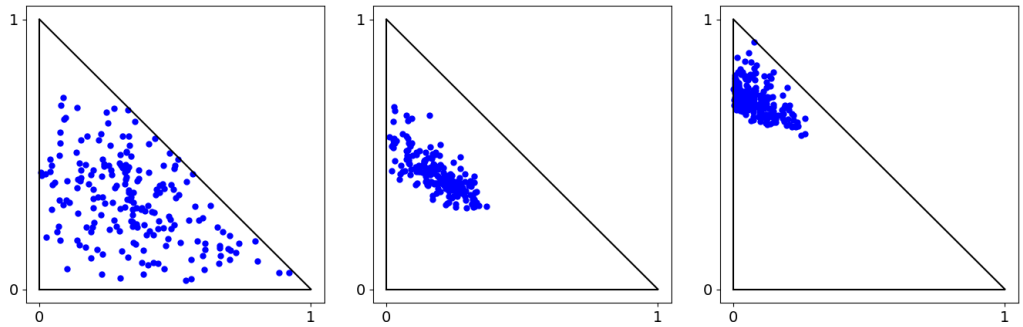


FIGURE 4.11 – Évolution de l'échantillon en direction du vecteur de poids caché.

On voit bien sur cet exemple que les mises à jour de la densité de probabilité mène à l'obtention d'échantillons de vecteurs de pondération de plus en plus proches entres eux et de plus en plus proches du poids réel $\hat{w}$.

Analyse des résultats Nous avons d'abord testé la robustesse aux erreurs de l'algorithme 4.4 en évaluant son efficacité selon différents taux d'erreur en faisant varier σ dans $\{0, 0.01, 0.02\}$. On observe alors la qualité de la recommandation x^* (la solution ayant le plus petit regret espéré maximum) après chaque question. La qualité de x^* est donnée par un score $s_{\hat{w}}(x^*)$ calculé selon la valeur réelle de x^* (par rapport au poids réel caché $\hat{w}$). Dans le cas du problème du sac à dos multi-agents, ce score est calculé par

$s_{\hat{w}}(x^*) = f_{\hat{w}}(x^*)/f_{\hat{w}}(\hat{x})$, où $\hat{x}$ est une solution $\mathrm{WS}_{\hat{w}}$-optimale. Dans le cas du problème d'affectation de ressources, ce score est donné par $s_{\hat{w}}(x^*) = \frac{1-f_{\hat{w}}(x^*)}{1-f_{\hat{w}}(\hat{w})}$. Ainsi, le score $s_{\hat{w}}(x^*)$ évalue la qualité de la recommandation x^* relativement à la valeur optimale réelle selon le système de valeurs du décideur. Plus la valeur du score est proche de 1, plus la solution recommandée à une valeur $\mathrm{WS}_{\hat{w}}$ proche de la valeur de la solution optimale. Le score moyen (sur 50 instances) en fonction du nombre de questions posées est indiqué sur les figures 4.12 et 4.13.

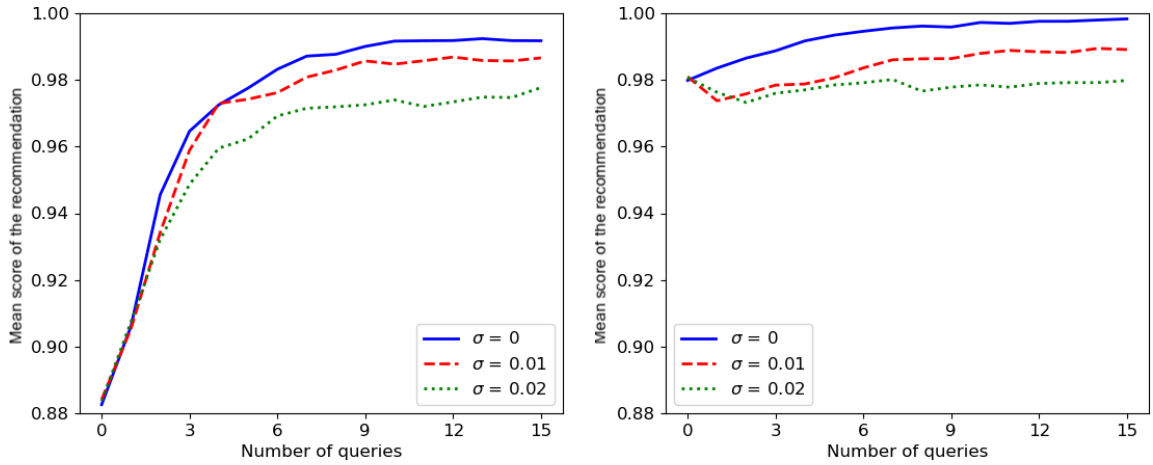


FIGURE 4.12 – Score moyen de la recommandation pour le poids caché de type w_{ext} (à gauche) et w_u (à droite) pour le problème du sac à dos multi-agents.

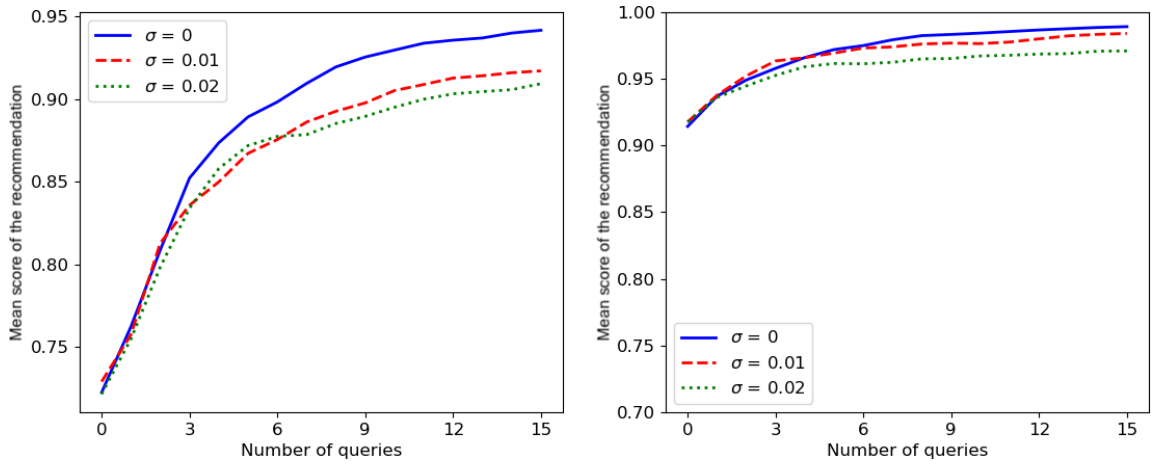


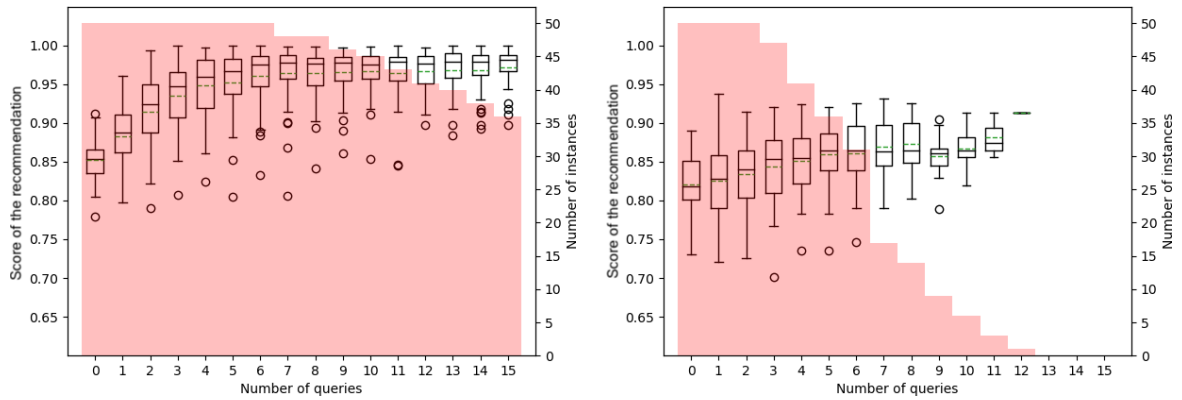
FIGURE 4.13 – Score moyen de la recommandation pour le poids caché de type w_{ext} (à gauche) et w_u (à droite) pour le problème d'affectation de ressources multicritère.

On observe, pour les instances des deux problèmes considérés, que le score de la recommandation est toujours amélioré par l'algorithme 4.4. Cette amélioration est plus ou moins significative selon la nature du poids caché (w_{ext} ou w_u) : lorsque le poids caché est généré uniformément dans le simplexe, la recommandation initiale est, en moyenne, de très bonne qualité, elle ne peut donc pas être significativement améliorée. Alors que dans le cas des poids w_{ext}, la qualité de la recommandation initiale est beaucoup moins

bonne, ce qui explique cette différence. Naturellement, la qualité de la recommandation est dégradée lorsque des erreurs sont présentes dans les réponses du décideur. Néanmoins, la recommandation à la fin de l'algorithme reste de très bonne qualité : dans le cas du problème du sac à dos multi-agents, la valeur du score est toujours ≥ 0.97 pour toute valeur de σ et pour les deux types de vecteurs w_{ext} et w_u. On peut également voir que la convergence de l'algorithme commence après seulement une dizaine de questions. Quant au problème d'affectation de ressources multicritère, la valeur du score à la fin de l'algorithme est ≥ 0.9 dans le cas des vecteurs du type w_{ext} et ≥ 0.96 dans le cas des vecteurs de type w_u. La qualité est moins bonne que pour le problème précédent, mais on peut voir que l'algorithme commence avec une recommandation initiale de qualité bien moins bonne. Dans les deux cas, on peut dire que l'algorithme est très efficace et permet de déterminer de très bonnes recommandations avec un nombre de questions relativement petit. Par exemple, après seulement 5 questions, l'algorithme recommande une solution avec un score ≥ 0.96 pour le problème du sac à dos et ≥ 0.86 pour le problème d'affectation.

Concernant les temps de calcul, le temps moyen d'exécution (sur les 50 instances) entre deux questions est d'environ 2 secondes (respectivement 4 secondes) pour les instances du problème du sac à dos multi-agents lorsque le poids caché est du type w_u (respectivement w_{ext}), et d'environ 1.3 secondes pour les instances du problème d'affectation de ressources multicritère (quel que soit le type de poids caché).

Nous avons ensuite comparé les performances de l'algorithme 4.4 avec celles de l'algorithme déterministe 3.1 (qui ne prend pas en compte la possibilité de la présence d'erreurs dans les réponses du décideur et qui ne donne pas au décideur la possibilité de se contredire). Pour cela, nous avons appliqué les deux algorithmes aux mêmes instances du problème du sac à dos multi-agents. Nous avons utilisé des poids cachés du type w_{ext}, et les réponses ont été simulées en appliquant un bruit Gaussien avec $\sigma = 0.02$. Notons qu'en utilisant un vecteur du type w_{ext}, l'impact des erreurs dans les réponses du décideur, dans le pire cas, est plus important que pour un poids du type w_u. La comparaison des performances des deux algorithmes est donnée par la figure 4.14. Dans ces figures, la i^e boîte à moustache donne les valeurs des scores de la recommandation effectuée après la question i pour chacune des 50 instances considérées : les extrémités des moustaches donnent les



(a) (b)

FIGURE 4.14 – Algorithme probabiliste 4.4 (figure (a)) vs. Algorithme déterministe 3.1 (figure (b)).

valeurs de score minimum et maximum, les extrémités de la boîte donnent les valeurs des 1^{er} (en bas) et 3^{e} quartiles, la ligne qui coupe la boîte représente la médiane tandis que la ligne pointillée représente la moyenne, enfin les cercles représentent des valeurs isolées. Quant à l'histogramme, il donne le nombre de valeurs observées pour chaque question : la colonne i donne le nombre d'instances pour lesquelles au moins i questions sont posées avant que le critère d'arrêt de l'algorithme ne se déclenche.

Ces derniers résultats numériques montrent l'intérêt de la prise en compte de la présence d'erreurs dans la procédure d'élicitation des préférences. En effet, les deux histogrammes montrent bien que l'approche déterministe (algorithme 3.1) converge, certes, plus rapidement (en termes de nombre de questions posées) que l'algorithme 4.4, mais les boîtes à moustaches de la figure 4.14(b) montrent cependant que l'algorithme converge vers une recommandation de mauvaise qualité. En effet, la valeur du score est moins bonne que pour l'algorithme 4.4 : l'algorithme 3.1 recommande toujours une solution avec un score ≤ 0.94 et avec un score ≤ 0.9 pour 75% des instances considérées, alors que l'algorithme 4.4 recommande, dès la 6^{e} question, une solution avec un score ≥ 0.96 pour 75% des instances considérées.

4.4 Conclusion du chapitre

Nous nous sommes intéressés dans ce chapitre à la conception d'algorithmes d'élicitation incrémentale qui prennent en compte la possibilité de la présence d'erreurs ou d'incohérences dans les réponses du décideur. Pour cela, nous associons à l'espace des paramètres une distribution de probabilité continue qui modélise notre incertitude concernant la valeur réelle des paramètres. La distribution de probabilité est mise à jour au fur et à mesure de l'acquisition de nouvelles informations sur les préférences du décideur. Ces informations sont récoltées par le biais de questions préférentielles déterminées à l'aide de la minimisation de regrets espérés. Nous nous sommes, dans un premier temps, intéressés à la conception d'un tel algorithme pour la détermination d'une solution optimale parmi un ensemble de solutions définies *explicitement*, puis nous avons étendu l'approche à la résolution de problèmes définis *sur domaine combinatoire*.

Sur domaine explicite, nous avons conçu un algorithme d'élicitation qui met à jour la densité de probabilité par régression *linéaire* Bayésienne. L'approche présentée ne se limite néanmoins pas à des modèles linéaires simples : on l'applique également à l'opérateur OWA et à l'intégrale de Choquet 2-additive qui sont des modèles non-linéaires. Pour cela, on utilise des bases de fonctions permettant de reformuler ces opérateurs comme des combinaisons linéaires des fonctions de ces bases. L'élicitation du paramètre d'un opérateur revient alors à éliciter la valeur des coefficients positifs de cette combinaison. On montre ensuite pourquoi cet algorithme ne peut être appliqué directement à la résolution de problèmes définis sur domaine combinatoire : le calcul des regrets espérés pour la détermination des questions et des recommandations nécessite alors un nombre exponentiel de calculs. Une extension possible de ce travail est de l'adapter à une plus large classe de capacités pour l'intégrale de Choquet afin d'accroître le pouvoir descriptif du modèle. Cette extension ne va pas de soi car le polyèdre des capacités k-additives pour $k \geq 3$ admet un grand nombre de points extrêmes autres que les capacités 0-1 données dans le cas $k = 2$ [Miranda *et al.*, 2006].

Nous avons ensuite étendu l'approche à la résolution de problèmes définis sur domaine combinatoire, en développant une nouvelle méthode de génération de contraintes et de variables permettant de calculer les regrets par programmation linéaire en variables mixtes. Cette méthode est générique et s'applique à tout problème possédant une formulation en programmation linéaire à variables entières. Elle peut également être appliquée à plusieurs modèles décisionnels s'ils sont linéaires ou qu'ils possèdent une linéarisation efficace (pour pouvoir appliquer l'algorithme de calcul des regrets) ainsi qu'une base de fonctions génératrices permettant d'appliquer la méthode de régression linéaire Bayésienne. Néanmoins, la linéarisation d'un modèle non-linéaire nécessite souvent l'introduction de nouvelles variables et contraintes (voir, par exemple, la linéarisation d'OWA [Ogryczak et Śliwiński, 2003]) qui peut ensuite impliquer une dégradation de la méthode de génération de contraintes et de variables (en termes de temps de calcul). Notons que notre méthode de calcul des regrets espérés est, à notre connaissance, la seule méthode permettant le calcul des regrets espérés sur domaine combinatoire.

Notons que les deux approches introduites dans ce chapitre sont *anytime* et génériques.

Même si l'efficacité théorique des méthodes proposées est difficile à montrer, nous présentons des expérimentations qui montrent l'efficacité empirique des deux algorithmes. En effet, différents tests numériques montrent l'efficacité de la méthode et sa robustesse par rapport aux erreurs commises par le décideur. Nous montrons également que même si ces méthodes convergent moins rapidement que des approches déterministes (telles que celles présentées dans le chapitre 2), elles présentent l'avantage d'offrir de bien meilleures recommandations que ces dernières lors de la présence d'erreurs. Néanmoins, ces méthodes peuvent s'avérer être coûteuses en temps de calcul dans certains cas : les opérations d'échantillonnage selon des distributions tronquées (algorithme 4.2) peuvent devenir lentes lorsque le nombre de critères augmente, puisque la taille des échantillons doit également augmenter si on veut obtenir un ensemble d'éléments suffisamment représentatifs de l'espace des paramètres et de la distribution de probabilité. On présente, dans la suite, des algorithmes permettant de surmonter cette difficulté en utilisant d'autres manières de représenter l'incertitude concernant les préférences du décideur.

Chapitre 5

Élicitation incrémentale par mise à jour Bayésienne fondée sur des polyèdres d'optimalité

Résumé du chapitre

Nous introduisons, dans ce chapitre, une nouvelle approche d'élicitation incrémentale, tolérante aux erreurs et aux incohérences du décideur, fondée cette fois sur la mise à jour d'une distribution de probabilité associée à l'espace des paramètres *discrétisé*. L'idée est de partitionner l'espace des paramètres en polyèdres d'optimalité, i.e. des polyèdres regroupant chacun l'ensemble des paramètres pour lesquels une solution donnée est optimale au sens de l'agrégateur considéré. Une distribution de probabilité est ensuite associée à cette partition et donne, pour chaque polyèdre, la probabilité qu'il contienne le paramètre représentant les préférences du décideur. Cette distribution peut également être interprétée comme la probabilité que chaque solution soit optimale. Sa mise à jour se fait de manière Bayésienne en utilisant les réponses du décideur aux questions que l'on choisit sur la base d'une évaluation des solutions par des regrets espérés. On alternera alors calcul des regrets espérés et mise à jour de la distribution de probabilité jusqu'à pouvoir déterminer une recommandation avec un niveau de confiance suffisant. La méthode que l'on présente ici est fondée sur l'hypothèse qu'il convient de modéliser les préférences du décideur par une fonction d'agrégation paramétrée linéaire *en son paramètre*. On montre l'efficacité pratique de notre méthode par des expérimentations numériques où l'on élicite les paramètres d'une somme pondérée (WS pour *Weighted Sum*) et d'une somme pondérée ordonnée (OWA pour *Ordered Weighted Average*). Le travail présenté dans ce chapitre a fait l'objet d'une publication à la conférence SUM [Bourdache *et al.*, 2019a].

5.1 Introduction

Nous avons pu voir dans le chapitre précédent l'intérêt d'utiliser une distribution de probabilité pour modéliser la connaissance partielle des préférences du décideur. L'utilisation d'une densité de probabilité que l'on met à jour lors de l'acquisition de nouvelles informations préférentielles (au lieu de la réduction systématique de l'espace des paramètres) nous a permis de concevoir une méthode d'élicitation des préférences tolérante aux erreurs ou aux incohérences dans les réponses du décideur. Ainsi, il est donné au décideur l'opportunité de se tromper ou de se contredire sans pour autant impacter de manière trop importante la qualité de la recommandation finale de l'algorithme. Néanmoins, la mise à jour d'une distribution de probabilité continue peut poser des problèmes computationnels. En effet, lorsqu'il n'existe pas de formule analytique permettant de mettre à jour la distribution de probabilité (comme c'était le cas dans le chapitre 4), il est nécessaire d'avoir recours à l'ajout de nouvelles variables (latentes) et à l'utilisation de méthodes d'échantillonnage afin d'approcher cette densité. Ces méthodes peuvent être coûteuses en temps de calcul lorsque le nombre de paramètres augmente, elle nécessitent donc un compromis entre coût computationnel et précision de l'approximation. De plus, notons que l'information apportée par une distribution de probabilité *continue* est beaucoup plus riche que l'information réellement nécessaire à la détermination d'une bonne recommandation. En effet, il est généralement suffisant de savoir que le paramètre représentant les préférences du décideur appartient à une zone restreinte de l'espace des paramètres pour pouvoir identifier une solution optimale sans ambiguïté. On peut donc envisager une modélisation consistant à découper l'espace des paramètres en zones particulières (à définir) puis à associer une probabilité à chacune de ces zones en fonction des informations préférentielles que l'on acquiert. Un choix pertinent de découpage peut être de partitionner l'espace des paramètres en polyèdres d'optimalité : à chaque solution réalisable correspond un polyèdre d'optimalité contenant toutes les instanciations possibles des paramètres pour lesquelles cette solution est optimale. Ainsi, ce découpage définit une partition de l'ensemble d'incertitude, et la probabilité que le paramètre réel du décideur appartienne à un polyèdre d'optimalité donne la probabilité que la solution associée soit optimale. Nous introduisons donc dans ce chapitre une nouvelle approche d'élicitation incrémentale fondée sur la mise à jour Bayésienne d'une distribution de probabilité *discrète* associée aux polyèdres d'optimalité.

L'idée de partitionner l'espace des paramètres est proche des travaux concernant la méthodologie SMAA (Stochastic Multiobjective Acceptability Analysis) qui a été introduite par Charnetski et Soland [1978] : étant donnés un ensemble de solutions (défini explicitement) et un ensemble de contraintes linéaires caractérisant l'espace des paramètres d'une somme pondérée (information partielle donnée par le décideur en amont de la résolution), les auteurs définissent la probabilité qu'une solution soit optimale proportionnellement à l'hypervolume de son polyèdre d'optimalité. Cette idée a été reprise ensuite par Lahdelma *et al.* [1998] où *l'indice d'acceptabilité* d'une solution est défini proportionnellement à l'hypervolume du polyèdre d'optimalité associé à cette solution. Cet indice donne donc une mesure de l'étendue des différentes valeurs de paramètres pour lesquelles cette solution est optimale. Ils étudient également le cas de la présence d'imprécisions ou d'incertitudes concernant les données du problème, i.e., les valeurs d'utilité des solutions selon les différents critères. Ils considèrent alors ces valeurs comme des variables aléatoires permettant

d'une part de définir des *indices d'acceptabilité stochastiques* à l'aide des hypervolumes espérés des polyèdres d'optimalité, et d'autre part d'introduire un *facteur de confiance* qui mesure le degré de précision des données afin de déterminer la pertinence d'une recommandation. La méthodologie SMAA a plus tard été adaptée aux intégrales de Choquet 2-additives par Angilella *et al.* [2015].

Ces travaux considèrent tous que l'incertitude concerne les évaluations des solutions (valeurs des critères) ou qu'elle provient de la variation entre les réponses données par différents décideurs. Ils considèrent également qu'un ensemble d'informations préférentielles est obtenu a priori, et qu'il n'est pas possible de demander au(x) décideur(s) de nous procurer de nouvelles informations en cours de résolution. Le travail présenté dans ce chapitre se différencie de ces travaux par :

- le fait de considérer que les *valeurs des critères* sont *précisément connues* et que seules les *valeurs des paramètres* de la fonction d'agrégation sont *à éliciter*,
- le fait de considérer que l'*incertitude* provient de potentielles *erreurs ou incohérences commises par le décideur* lorsqu'il répond aux questions préférentielles,
- le fait de se placer dans un cadre où la prise de décision est réalisée à l'aide d'interactions avec le décideur en cours de résolution, l'élicitation des préférences étant donc *incrémentale*.

Contexte et notations. On considère dans ce chapitre que l'on résout un problème où l'espace des solutions $\mathcal{X}$ est défini de manière explicite, et que les solutions de $\mathcal{X}$ sont évaluées selon n critères. L'image de $\mathcal{X}$ dans l'espace des critères est alors donnée par $\mathcal{Y} = \{u(x) \in \mathbb{R}^n, \forall x \in \mathcal{X}\}$ où $u(x)$ est le vecteur de performances associé à la solution x, et dont la $i^{ème}$ composante représente la performance de x sur le critère i. On présente notre méthode dans le cadre d'une modélisation des préférences du décideur par une fonction d'agrégation f_w linéaire en son paramètre $w \in W$, où l'espace des paramètres est défini par $W = \{w \in \mathbb{R}^n | \sum_{i=1}^{n} w_i = 1\}$. Notons que la contrainte de normalisation permet de réduire l'espace des paramètres possibles sans restreindre les possibilités de modélisation des compromis possibles. Pour f_w, on considère dans la suite le cas particulier de la modélisation des préférences par une somme pondérée (WS pour *Weighted Sum*) et par un OWA (*Ordered Weighted Average*). On se place ici dans le cadre d'une maximisation de l'opérateur d'agrégation considéré.

Organisation du chapitre. Ce chapitre est organisé de la manière suivante : nous formalisons dans la section 5.2 la notion de polyèdres d'optimalité et nous définissons une distribution de probabilité a priori associant, à chaque polyèdre d'optimalité, la probabilité que la solution associée soit optimale. Nous donnons ensuite dans la section 5.3 notre algorithme d'élicitation incrémentale en explicitant la stratégie du choix de la question posée, consistant en une nouvelle adaptation de la stratégie CSS. Nous spécifions également notre méthode de mise à jour Bayésienne. Enfin, la section 5.4 donne des résultats numériques montrant l'efficacité de la méthode et sa robustesse par rapport aux erreurs et aux incohérences du décideur dans ses réponses.

5.2 Représentation de l'incertitude sur les pramètres préférentiels via des polyèdres d'optimalité

Les méthodes d'élicitation incrémentale classiques consistent à réduire progressivement l'espace des paramètres jusqu'à obtenir un polyèdre suffisamment réduit pour qu'une solution nécessairement optimale existe (une solution optimale pour toute instanciation possible des paramètres). Nous appelons un tel polyèdre un *polyèdre d'optimalité*. Lorsque certaines réponses sont erronées, une telle méthode peut mener à une recommandation plus ou moins mauvaise (cf. exemple 4.1). On propose alors de procéder différemment : nous allons définir une approche consistant à déterminer tous les polyèdres d'optimalité possibles a priori, puis à déterminer, à l'aide des informations préférentielles données par le décideur, lequel contient le plus probablement le paramètre représentant les préférences du décideur afin d'en déduire une recommandation pertinente.

Étant donnée une fonction d'agrégation f_w (linéaire en w) représentant le système de valeurs du décideur, on définit un *polyèdre d'optimalité* dans W par :

Définition 5.1. *Soit $x \in \mathcal{X}$ une solution réalisable et soit W l'ensemble des paramètres possibles de la fonction d'agrégation f_w. Le polyèdre d'optimalité associé à la solution x, noté W_x, est donné par :*

$$W_x = \{w \in W | f_w(x) \geq f_w(y), \forall y \in \mathcal{X}\}$$

Autrement dit, W_x désigne l'ensemble des valeurs de paramètres de W pour lesquels la solution x est f_w-optimale. Notons que lorsque la fonction d'agrégation f_w est linéaire en son paramètre, comme c'est le cas des opérateurs WS et OWA, l'ensemble W_x est représenté par un polyèdre convexe car les contraintes $f_w(x) \geq f_w(y)$, pour tout couple de solutions $(x, y) \in \mathcal{X}^2$ fixé, sont linéaires.

On peut alors déterminer une partition $W_{\mathcal{X}}$ de l'espace W en calculant les polyèdres d'optimalité associés aux solutions de $\mathcal{X}$, et en omettant les ensembles W_x tels que :

- W_x est vide, i.e., x n'est pas potentiellement f_w-optimale. Autrement dit, il n'existe aucun paramètre w tel que la solution x est f_w-optimale,
- W_x n'est pas de pleine dimension, i.e., pour tout paramètre $w \in W_x$, il existe une solution $y \in \mathcal{X}$ telle que $f_w(x) = f_w(y)$, ce qui exprime le fait que W_x est l'une des faces d'un autre polyèdre d'optimalité,
- il existe une solution $y \in \mathcal{X}$ telle que $y \neq x$ et $W_y = W_x$ (on omet arbitrairement W_x ou W_y).

Notons que pour éviter le cas des polyèdres confondus (dernier cas), on peut supposer que les vecteurs de performances (triés) sont tous distincts dans le cas de la somme pondérée (respectivement OWA).

Exemple 5.1. *Soient x, y, z et t quatre solutions associées aux vecteurs de performances $u(x) = (14, 9, 10), u(y) = (10, 12, 10), u(z) = (9, 16, 6)$ et $u(t) = (16, 9, 6)$. La figure 5.1(a) (respectivement 5.1(b)) montre l'espace W partitionné en polyèdres d'optimalité lorsque l'on considère l'opérateur WS (respectivement OWA) pour modéliser les préférences du décideur. Sur cette figure, on ne représente que les composantes w_1 et w_2 puisque la composante w_3 peut être inférée grâce à la contrainte de normalisation.*

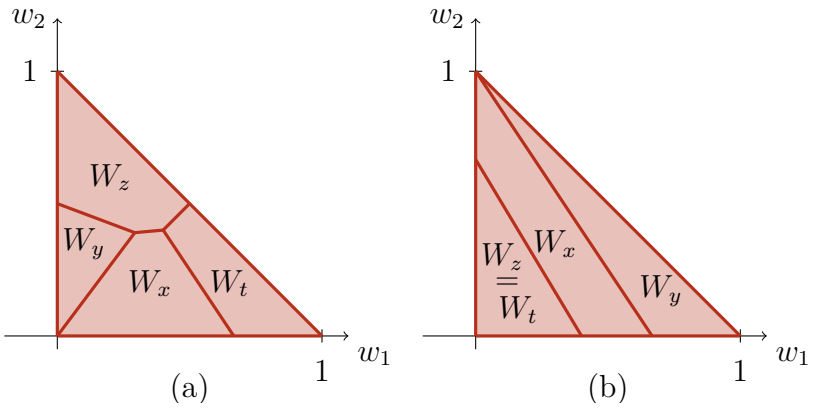


FIGURE 5.1 – Polyèdres d'optimalité pour les solutions x, y, z et t pour WS (à gauche) et OWA (à droite).

On peut voir ici que les deux solutions z et t ont des polyèdres d'optimalité identiques dans le cas de la modélisation des préférences par un OWA. Ceci est dû au fait que z et t ont le même vecteur de performances trié ; on a alors $\text{OWA}_w(z) = \text{OWA}_w(t)$ *pour tout paramètre $w \in W$. Dans ce cas, la partition de W sera donnée par $W_{\mathcal{X}} = \{W_x, W_y, W_z\}$ ou par $W_{\mathcal{X}} = \{W_x, W_y, W_t\}$.*

Une fois cette partition obtenue, déterminer une solution optimale pour le décideur revient à déterminer quel est le polyèdre d'optimalité contenant le paramètre $\hat{w}$ représentant ses préférences. Afin de modéliser l'incertitude concernant la localisation de $\hat{w}$ dans la partition (et donc le polyèdre auquel $\hat{w}$ appartient), nous allons associer une probabilité à chaque polyèdre d'optimalité de la partition. On note $P(\hat{w} \in W_x), \forall x \in \mathcal{X}$, la probabilité que $\hat{w}$ se situe dans le polyèdre W_x. La valeur $P(\hat{w} \in W_x)$ nous donne donc également la probabilité que la solution x soit optimale ; on note alors cette probabilité $P(x) = P(\hat{w} \in W_x) \in [0, 1], \forall x \in \mathcal{X}$. Dans la suite, on parlera de P comme de la distribution de probabilité associée aux solutions de l'ensemble $\mathcal{X}$.

Puisque l'on ne dispose initialement d'aucune information concernant les préférences du décideur, on définit la distribution a priori de manière à ce que la probabilité d'optimalité de chaque solution soit proportionnelle à l'hypervolume de son polyèdre d'optimalité. Cette initialisation est similaire à ce qui a été proposé par Charnetski et Soland [1978] dans un cadre d'élicitation a priori (donc non incrémentale) où aucune mise à jour de la distribution de probabilité n'est envisagée. L'idée étant que l'hypervolume de W_x donne une mesure de la proportion de paramètres $w \in W$ pour lesquels la solution x est f_w-optimale. Plus formellement, on définit la probabilité a priori que la solution x soit optimale par :

$$P(x) = \frac{vol_{W_x}}{vol_W} \tag{5.1}$$

où $vol_{\mathcal{W}}$ désigne l'hypervolume d'un polyèdre convexe $\mathcal{W}$. Notons que cette initialisation revient à supposer une ignorance complète concernant la densité de probabilité au sein de chaque polyèdre $W_x, x \in \mathcal{X}$. Cette hypothèse n'est pas contraignante puisque l'on ne s'intéresse ici qu'à la probabilité d'optimalité des solutions ou, autrement dit, à la localisation du paramètre $\hat{w}$ dans la partition $W_{\mathcal{X}}$.

Nous allons maintenant présenter une méthode d'élicitation incrémentale fondée sur une mise à jour Bayésienne de la distribution P.

5.3 Algorithme d'élicitation incrémentale

Une fois l'ensemble W partitionné, notre algorithme suit le principe suivant : on associe une probabilité a priori à chaque solution $z \in \mathcal{X}$ suivant la formule de l'équation (5.1), puis on alterne question préférentielle et mise à jour Bayésienne de la distribution de probabilité en utilisant les réponses du décideur.

Le choix de la question à poser est, encore une fois, un élément clé de l'efficacité de la procédure d'élicitation. On propose ici une adaptation de la stratégie CSS à notre contexte.

5.3.1 Stratégie d'élicitation incrémentale

Afin d'obtenir la question la plus informative possible, on utilise une stratégie qui est une nouvelle fois fondée sur la minimisation des regrets espérés. Nous allons tout d'abord définir les différents regrets espérés dans ce nouveau cadre où l'incertitude est représentée par une distribution de probabilité discrète associée aux solutions de $\mathcal{X}$.

Définition 5.2. *Soit f_w une fonction d'agrégation de paramètre $w \in W$ et soit P une distribution de probabilité associée à $\mathcal{X}$. On définit le regret espéré de la paire (Pairwise Expected Regret) $x, y \in \mathcal{X}$ par :*

$$\mathrm{PER}(x, y, \mathcal{X}, P) = \sum_{z \in \mathcal{X}} \max\{0, \mathrm{PMR}(x, y, W_z)\} P(z)$$

avec pour rappel :

$$\mathrm{PMR}(x, y, W_z) = \max_{w \in W_z} \{f_w(y) - f_w(x)\}$$

En d'autres termes, le PER donne la perte de performance espérée induite par le choix de la solution x au lieu de la solution y, tandis que le PMR désigne la perte de performance maximum de ce même choix dans le polyèdre d'optimalité W_z. Il est donné par une espérance du PMR qui donne, pour chaque polyèdre W_z, la pire perte d'utilité induite par ce même choix. L'utilisation du PMR pour déterminer le regret d'une paire dans un polyèdre W_z se justifie par l'hypothèse de l'ignorance complète concernant la densité de probabilité des paramètres au sein de W_z : on considère alors le pire cas possible. Le PER est utilisé pour définir le pire regret espéré de chaque solution :

Définition 5.3. *Soit f_w une fonction d'agrégation de paramètre $w \in W$ et soit P une distribution de probabilité associée à $\mathcal{X}$. On définit le regret espéré maximum (Maximum Expected Regret) de la solution $x \in \mathcal{X}$ par :*

$$\mathrm{MER}(x, \mathcal{X}, P) = \max_{y \in \mathcal{X}} \mathrm{PER}(x, y, \mathcal{X}, P)$$

En d'autres termes, le MER donne la pire perte d'utilité espérée induite par le choix de la solution x au lieu de n'importe quelle autre solution de l'ensemble $\mathcal{X}$. Le MER permet alors d'évaluer chaque solution $x \in \mathcal{X}$ et de déterminer la meilleure solution au sens de ce critère d'évaluation :

Définition 5.4. *Soit f_w une fonction d'agrégation de paramètre $w \in W$ et soit P une distribution de probabilité associée à $\mathcal{X}$. On définit le minimax regret espéré (MiniMax Expected Regret) de $\mathcal{X}$ par :*

$$\mathrm{MMER}(\mathcal{X}, P) = \min_{x \in \mathcal{X}} \mathrm{MER}(x, \mathcal{X}, P)$$

Autrement dit, le MMER donne la meilleure valeur de regret espéré pire cas.

On peut alors utiliser ces regrets espérés afin de déterminer une recommandation ou une question à poser, de manière similaire à ce que l'on a présenté dans le chapitre précédent. À n'importe quelle étape de l'algorithme, la solution ayant le plus petit regret espéré maximum donne une recommandation pertinente puisqu'elle minimise la perte d'utilité espérée au vu des informations acquises. Si on souhaite approfondir la connaissance des préférences du décideur, on adopte une version adaptée de la stratégie CSS [Boutilier *et al.*, 2006b] : on propose au décideur de comparer la meilleure recommandation courante (au sens du regret espéré maximum) $x^* = \arg\min_{x\in\mathcal{X}} \mathrm{MER}(x, \mathcal{X}, P)$ à son meilleur adversaire (toujours au sens du regret espéré maximum) donné par $y^* = \arg\max_{y\in\mathcal{X}} \mathrm{PER}(x^*, y, P)$. La réponse du décideur nous apporte une nouvelle information préférentielle nous permettant de mettre à jour la distribution P. La méthode est détaillée dans l'algorithme 5.1.

Algorithme 5.1 :

Entrées : $\mathcal{X}$: ensemble de solutions défini explicitement ; W : espace des paramètres ; δ : seuil de tolérance.

1 Déterminer $W_x, \forall x \in \mathcal{X}$;
2 Supprimer de $\mathcal{X}$ toute solution telle que $W_x = \emptyset$ **ou** W_x n'est pas de pleine dimension **ou** $\exists y \in \mathcal{X}$ telle que $W_x = W_y$;
3 Poser $P(x) \leftarrow \frac{vol_{W_x}}{vol_W}$ pour tout $x \in \mathcal{X}$;
4 $i \leftarrow 1$;
5 **répéter**
6 $\quad x^{(i)} \leftarrow \arg\min_{x\in\mathcal{X}} \mathrm{MER}(x, \mathcal{X}, P)$;
7 $\quad y^{(i)} \leftarrow \arg\max_{y\in\mathcal{X}} \mathrm{PER}(x^{(i)}, y, P)$;
8 $\quad$ Demander au décideur s'il préfère $x^{(i)}$ ou $y^{(i)}$;
9 $\quad r^{(i)} \leftarrow 1$ si la réponse est oui et 0 sinon ;
10 $\quad$ Calculer $P(z|r^{(i)})$ pour tout $x \in \mathcal{X}$;
11 $\quad i \leftarrow i + 1$;
12 **jusqu'à** $\mathrm{MMER}(\mathcal{X}, P) \leq \delta$;
13 **retourner** $\hat{x} \in \arg\min_{x\in\mathcal{X}} \mathrm{MER}(x, \mathcal{X}, P)$

Le calcul de la distribution de probabilité a posteriori $P(z|r^{(i)})$ (ligne 10 de l'algorithme) se fait par une méthode de mise à jour Bayésienne que nous présentons dans la sous-section suivante.

5.3.2 Mise à jour Bayésienne de la distribution de probabilité

À chaque étape i de l'algorithme 5.1, on demande au décideur de comparer deux solutions $x^{(i)}$ et $y^{(i)}$. Soit $r^{(i)}$ la variable binaire qui modélise la réponse du décideur et qui vaut 1 si et seulement si le décideur déclare préférer la solution $x^{(i)}$ à la solution $y^{(i)}$. En utilisant la formule de Bayes, la probabilité a posteriori qu'une alternative $z \in \mathcal{X}$ soit optimale est donnée par :

$$P(z|r^{(1)}, \ldots, r^{(i)}) = \frac{P(r^{(1)}, \ldots, r^{(i)}|z)P(z)}{\sum_{x\in\mathcal{X}} P(r^{(i)}, x)} \tag{5.2}$$

où $P(r^{(1)}, \ldots, r^{(i)}|z)$ est la fonction de vraisemblance, $P(z)$ est la probabilité a priori que la solution z soit optimale, et $\sum_{x \in \mathcal{X}} P(r^{(i)}, x)$ est la somme des probabilités jointes de $r^{(i)}$ et de z. On peut exprimer $P(z|r^{(1)}, \ldots, r^{(i)})$ plus simplement par :

$$P(z|r^{(1)}, \ldots, r^{(i)}) \propto P(r^{(1)}, \ldots, r^{(i)}|z)P(z)$$

On suppose ici que les réponses $r^{(1)}, \ldots, r^{(i)}$ sont des variables aléatoires mutuellement indépendantes conditionnellement au paramètre (caché) représentant les préférences du décideur. La valeur de $r^{(i)}$ ne dépend pas des réponses précédentes mais dépend uniquement du paramètre (caché) $\hat{w}$ ainsi que des vecteurs de performance de $x^{(i)}$ et de $y^{(i)}$. Notons que cette indépendance conditionnellement à $\hat{w}$ signifie que chaque réponse dépendra de la position de $\hat{w}$ dans la partition mais pas des réponses précédentes. Ainsi, par définition de notre distribution de probabilité ($P(\hat{w} \in W_z) = P(z)$), les variables $r^{(1)}, \ldots, r^{(i)}$ sont des variables mutuellement indépendantes conditionnellement à la solution optimale. Cette hypothèse d'indépendance est standard en élicitation Bayésienne (voir par exemple [Vendrov *et al.*, 2020]), et nous permet d'isoler la réponse $r^{(i)}$ des réponses précédentes en écrivant :

$$P(z|r^{(1)}, \ldots, r^{(i)}) \propto P(r^{(i)}|z)P(r^{(1)}, \ldots, r^{(i-1)}|z)P(z)$$

En appliquant la formule de Bayes, l'expression $P(r^{(1)}, \ldots, r^{(i-1)}|z)P(z)$ est proportionnelle à $P(z|r^{(1)}, \ldots, r^{(i-1)})$, et on obtient :

$$P(z|r^{(1)}, \ldots, r^{(i)}) \propto P(r^{(i)}|z)P(z|r^{(1)}, \ldots, r^{(i-1)})$$

Afin de simplifier la lecture, on notera à présent la probabilité a posteriori $P(z|r^{(1)}, \ldots, r^{(i)})$ par $P(z|r^{(i)})$, et de manière similaire on notera la probabilité $P(z|r^{(1)}, \ldots, r^{(i-1)})$, par $P(z)$. On écrit alors :

$$P(z|r^{(i)}) \propto P(r^{(i)}|z)P(z) \tag{5.3}$$

La fonction de vraisemblance $P(r^{(i)}|z)$ est la probabilité conditionnelle que la réponse du décideur soit $r^{(i)}$ sachant que la solution z est f_w-optimale. On propose de la définir par :

$$P(r^{(i)} = 1|z) = \begin{cases} \lambda & \text{si } W_z \subseteq W_{x^{(i)} \succsim y^{(i)}} \\ 1 - \lambda & \text{si } W_z \cap W_{x^{(i)} \succsim y^{(i)}} = \emptyset \\ \sum_{z \in \mathcal{X}} P(r^{(i)} = 1, z) & \text{sinon} \end{cases} \tag{5.4}$$

où $W_{x^{(i)} \succsim y^{(i)}}$ désigne le sous-ensemble de W contenant tous les paramètres w compatibles avec la réponse $r^{(i)} = 1$, i.e., $W_{x^{(i)} \succsim y^{(i)}} = \{w \in W | f_w(x^{(i)}) \geq f_w(y^{(i)})\}$, et où $\lambda \in (\frac{1}{2}, 1]$ est une constante donnant une estimation de la probabilité que la réponse du décideur à chaque question soit correcte.

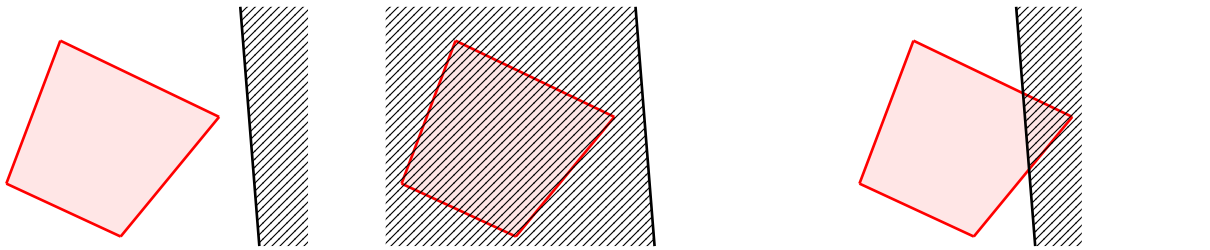
Cette définition de la fonction de vraisemblance est fondée sur une idée simple (utilisée notamment par Nowak [2009]) : les probabilités des polyèdres W_z qui sont partiellement compatibles avec la nouvelle information préférentielle ($W_z \cap W_{x^{(i)} \succsim y^{(i)}} \neq \emptyset$ et $W_z \cap W_{x^{(i)} \prec y^{(i)}} \neq \emptyset$) sont inchangées. En effet, en utilisant la formule de l'équation (5.2), on voit bien que la mise à jour effectuée par (5.4) revient à poser $P(z|r^{(i)}) = P(z)$. Quant aux polyèdres W_z qui sont entièrement compatibles avec la nouvelle information préférentielle ($W_z \subseteq W_{x^{(i)} \succsim y^{(i)}}$), leur probabilité est renforcée par rapport à celles des polyèdres

qui ne le sont pas. Le paramètre λ contrôle alors l'intensité du renforcement de ces probabilités. Plus λ est grand, plus les probabilités des polyèdres compatibles avec la nouvelle information préférentielle seront renforcées. Il faut tout de même veiller à ne pas choisir un λ trop grand car cela engendrerait des mises à jour drastiques et reviendrait à supposer que la probabilité que le décideur se trompe est très faible. Une telle supposition n'est pas toujours vraie et peut entraîner la détermination d'une recommandation de très mauvaise qualité lors de la présence de mauvaises réponses.

Les trois cas de mise à jour sont illustrés par l'exemple suivant :

Exemple 5.2. *Supposons que l'on dispose d'une solution z dont le polyèdre d'optimalité est représenté en rouge sur la figure 5.2, et qu'à l'étape i de l'algorithme, le décideur déclare préférer une solution $x^{(i)}$ à une solution $y^{(i)}$. On note alors $r^{(i)} = 1$, et on met à jour la probabilité que z soit optimale en déterminant la vraisemblance $P(r^{(i)} = 1|z)$ selon l'un des 3 cas donnés plus haut. Ces cas sont illustrés par la figure 5.2 où la partie hachurée correspond au demi-espace qui n'est pas compatible avec la nouvelle information préférentielle (i.e., $W_{x^{(i)} \prec y^{(i)}}$) :*

- *premier cas : W_z est inclus dans l'espace des paramètres qui privilégient la solution $x^{(i)}$ à la solution $y^{(i)}$, i.e., tous les paramètres $w \in W_z$ respectent la contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$ (figure de gauche). La probabilité $P(z|r^{(i)} = 1)$ est donc renforcée (par rapport à celles des polyèdres dans les deux cas suivants) en posant $P(r^{(i)} = 1|z) = \lambda, \lambda \in (\frac{1}{2}, 1]$,*
- *deuxième cas : aucun élément de W_z ne respecte la contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$ (figure du milieu). La probabilité $P(z|r^{(i)} = 1)$ est donc réduite (par rapport à celles des polyèdres dans les deux autres cas) en posant $P(r^{(i)} = 1|z) = 1 - \lambda, , \lambda \in (\frac{1}{2}, 1]$,*
- *troisième cas : W_z est coupé par la contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$ (figure de droite). La probabilité $P(z|r^{(i)} = 1)$ est inchangée et vaut $P(z)$ en posant $P(r^{(i)} = 1|z) = \sum_{z \in \mathcal{X}} P(r^{(i)} = 1, z)$.*



$P(r^{(i)} = 1|z) = \lambda$ $P(r^{(i)} = 1|z) = 1 - \lambda$ $P(r^{(i)} = 1|z) = \sum_{z \in \mathcal{X}} P(r^{(i)} = 1, z)$

FIGURE 5.2 – Positions du polyèdre W_z par rapport à la nouvelle contrainte.

Du fait de l'hypothèse d'ignorance complète concernant la densité de probabilité au sein d'un polyèdre d'optimalité, on considère dans le troisième cas que la nouvelle information préférentielle n'est pas suffisamment informative pour pouvoir mettre à jour la probabilité que la solution z soit optimale. Ainsi, pour toute solution z telle que le polyèdre d'optimalité associé est coupé par la contrainte induite par la nouvelle information préférentielle, on fixe la vraisemblance $P(r^{(i)}|z)$ à $\sum_{z \in \mathcal{X}} P(r^{(i)}, z)$ de manière à ce qu'aucune mise à jour ne soit faite.

Notons que dans les trois cas la mise à jour de la probabilité de z s'exprime en fonction de la probabilité $\sum_{z\in\mathcal{X}} P(r^{(i)}, z)$ que l'on ne peut obtenir par :

$$\sum_{z\in\mathcal{X}} P(r^{(i)}, z) = \sum_{z\in\mathcal{X}} P(r^{(i)}|z)P(z)$$

Or, la vraisemblance $P(r^{(i)}|z)$ correspond elle même à $\sum_{z\in\mathcal{X}} P(r^{(i)}, z)$ dans le cas où le polyèdre W_z est coupé par la nouvelle contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$. Néanmoins, il n'est, en pratique, pas nécessaire de déterminer cette probabilité pour calculer $P(z|r^{(i)})$. En effet, dans le cas où W_z est coupé par la nouvelle contrainte, la détermination de la valeur exacte de la vraisemblance n'est pas nécessaire à la détermination de $P(z|r^{(i)})$ qui vaut alors simplement $P(z)$. Dans les autres cas, on utilise le fait que la probabilité $P(z|r^{(i)})$ est proportionnelle à $P(r^{(i)}|z)P(z)$ (car $\forall z \in \mathcal{X}$, la quantité $P(r^{(i)}|z)P(z)$ est divisée par la même quantité $\sum_{z\in\mathcal{X}} P(r^{(i)}, z)$).

Plus précisément, si on note $\mathcal{P}$ le sous-ensemble des solutions de $\mathcal{X}$ pour lesquelles les polyèdres d'optimalité associés ne sont pas coupés par la contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$, on obtient la valeur exacte de $P(z|r^{(i)})$ pour tout $z \in \mathcal{P}$ par :

$$P(z|r^{(i)}) = P(r^{(i)}|z)P(z) \times \frac{\sum_{y\in\mathcal{P}} P(y)}{\sum_{y\in\mathcal{P}} P(r^{(i)}|y)P(y)} \tag{5.5}$$

Ainsi, la probabilité a posteriori de tout polyèdre est déterminée sans avoir à calculer $\sum_{z\in\mathcal{X}} P(r^{(i)}, z)$. On peut facilement vérifier que cette mise à jour mène bien à une distribution de probabilité qui respecte la condition $\sum_{z\in\mathcal{X}} P(z|r^{(i)}) = 1$:

$$\begin{aligned}
\sum_{z\in\mathcal{X}} P(z|r^{(i)}) &= \sum_{z\in\mathcal{P}} P(z|r^{(i)}) + \sum_{z\in\mathcal{X}\setminus\mathcal{P}} P(z|r^{(i)}) \\
&= \sum_{z\in\mathcal{P}} \left\{ P(r^{(i)}|z)P(z) \times \frac{\sum_{y\in\mathcal{P}} P(y)}{\sum_{y\in\mathcal{P}} P(r^{(i)}|y)P(y)} \right\} + \sum_{z\in\mathcal{X}\setminus\mathcal{P}} P(z) \\
&= \sum_{z\in\mathcal{P}} \left\{ \sum_{y\in\mathcal{P}} P(y) \frac{P(r^{(i)}|z)P(z)}{\sum_{y\in\mathcal{P}} P(r^{(i)}|y)P(y)} \right\} + \sum_{z\in\mathcal{X}\setminus\mathcal{P}} P(z) \\
&= \sum_{y\in\mathcal{P}} P(y) \left\{ \frac{\sum_{z\in\mathcal{P}} P(r^{(i)}|z)P(z)}{\sum_{y\in\mathcal{P}} P(r^{(i)}|y)P(y)} \right\} + \sum_{z\in\mathcal{X}\setminus\mathcal{P}} P(z) \\
&= \sum_{y\in\mathcal{P}} P(y) + \sum_{z\in\mathcal{X}\setminus\mathcal{P}} P(z) \\
&= \sum_{y\in\mathcal{X}} P(y) = 1
\end{aligned}$$

Afin de mettre à jour de la distribution de probabilité, il suffit donc de connaître la position de chaque polyèdre d'optimalité par rapport à l'hyperplan induit par la nouvelle information préférentielle. Avant de s'intéresser à la méthode de calcul pour déterminer la position exacte du polyèdre, on présente ici un résultat justifiant la pertinence de cette procédure de mise à jour Bayésienne.

Pertinence de la méthode de mise à jour

Lorsque la solution $f_{\hat{w}}$-optimale (où $\hat{w}$ est le poids caché représentant les préférences du décideur), notée $\hat{x}$, est unique et que le décideur répond correctement à la question i, alors la mise en œuvre de l'algorithme 5.1 en utilisant la fonction de vraisemblance de l'équation (5.4) pour la mise à jour des probabilités (ligne 10 de l'algorithme) garantit que la probabilité que la solution $\hat{x}$ soit optimale ne peut pas décroître :

Proposition 5.1. *S'il existe une unique solution optimale $\hat{x} \in \mathcal{X}$ et que le décideur répond correctement à la question i, alors :*

$$P(\hat{x}|r^{(i)}) \geq P(\hat{x})$$

Démonstration. Supposons, sans perte de généralité, que la réponse du décideur à la question i est "$x^{(i)} \succsim y^{(i)}$". On distingue deux cas :

Cas 1. Le polyèdre d'optimalité $W_{\hat{x}}$ associé à la solution optimale $\hat{x}$ est coupé par la contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$; dans ce cas, $P(\hat{x}|r^{(i)}) = P(\hat{x})$ par l'équation (5.4).

Cas 2. Lorsque $W_{\hat{x}}$ n'est pas coupé par la contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$; dans ce cas, on a on a $P(r^{(i)} = 1|\hat{x}) = \lambda$ (car par hypothèse le décideur répond correctement à la question). Par l'équation 5.5, on a :

$$P(\hat{x}|r^{(i)}) = \underbrace{\left(\frac{\lambda \sum_{y\in\mathcal{P}} P(y)}{\sum_{y\in\mathcal{P}} P(r^{(i)} = 1|y)P(y)}\right)}_{\text{ratio } \rho} P(\hat{x})$$

Nous allons maintenant montrer que $\rho \geq 1$. Notons $\mathcal{P}_\lambda$ le sous-ensemble de solutions $y \in \mathcal{P}$ tel que $P(r^{(i)} = 1|y) = \lambda$. On a :

$$\sum_{y\in\mathcal{P}} P(r^{(i)} = 1|y)P(y) = \lambda \sum_{y\in\mathcal{P}_\lambda} P(y) + (1-\lambda) \sum_{y\in\mathcal{P}_{1-\lambda}} P(y)$$

car $\mathcal{P} = \mathcal{P}_\lambda \cup \mathcal{P}_{1-\lambda}$ et $\mathcal{P}_\lambda \cap \mathcal{P}_{1-\lambda} = \emptyset$. Puisque $\lambda > 1/2$, on en déduit que :

$$\sum_{y\in\mathcal{P}} P(r^{(i)} = 1|y)P(y) \leq \lambda \sum_{y\in\mathcal{P}_\lambda} P(y) + \lambda \sum_{y\in\mathcal{P}_{1-\lambda}} P(y)$$

Notons que le seul cas pour lequel l'égalité est vérifiée est le cas où $\mathcal{P}_{1-\lambda} = \emptyset$. Finalement, on déduit de cette inégalité :

$$\sum_{y\in\mathcal{P}} P(r^{(i)} = 1|y)P(y) \leq \lambda \sum_{y\in\mathcal{P}} P(y)$$

Par conséquent,

$$\rho = \frac{\lambda \sum_{y\in\mathcal{P}} P(y)}{\sum_{y\in\mathcal{P}} P(r^{(i)} = 1|y)P(y)} \geq 1$$

En conclusion, $\rho \geq 1$ et $P(\hat{x}|r^{(i)}) = \rho P(\hat{x})$ impliquent $P(\hat{x}|r^{(i)}) \geq P(\hat{x})$. □

Notons que la solution $\hat{x}$ est la seule solution de l'ensemble $\mathcal{X}$ pour laquelle cette proposition est vraie. Il existe donc toujours, sous l'hypothèse que le décideur ne se trompe pas dans ses réponses, un questionnaire tel que la solution $\hat{x}$ vérifie, au bout d'un certain nombre de questions, l'inégalité $\text{MER}(\hat{x}, \mathcal{X}, P) \leq \text{MER}(x, \mathcal{X}, P), \forall x \in \mathcal{X}$, et devient ainsi la recommandation courante. En effet, lorsque la probabilité $P(\hat{x})$ est suffisamment grande, le PER entre $\hat{x}$ et toute autre solution de $\mathcal{X}$ devient de plus en plus petit car l'expression $\text{PER}(\hat{x}, y, W_{\hat{x}}) = \sum_{z\in\mathcal{X}} \max\{0, \text{PMR}(\hat{x}, y, W_z)\}P(z)$ tend vers 0. Ceci est dû au fait que tous les polyèdres W_z tels que $\text{PMR}(\hat{x}, y, W_z) > 0$ sont associés à des solutions z ayant une probabilité $P(z)$ faible ; ainsi le terme $\text{PMR}(\hat{x}, y, W_{\hat{x}})$, qui vaut 0 (par définition de $W_{\hat{x}}$ et du PMR), devient dominant dans $\text{PER}(\hat{x}, y, W_{\hat{x}})$.

Détermination de la position des polyèdres d'optimalité par rapport à la nouvelle contrainte

Supposons, sans perte de généralité, que la réponse du décideur à la question i est "$x^{(i)} \succsim y^{(i)}$". Lorsqu'un polyèdre $W_z, z \in \mathcal{X}$, est traversé par la contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$, alors il existe un sous-ensemble de paramètres $W'_z \subset W_z$ tel que $\forall w \in W'_z, f_w(x^{(i)}) > f_w(y^{(i)})$, ainsi qu'un sous-ensemble de paramètres $W''_z \subset W_z$ tel que $\forall w \in W''_z, f_w(x^{(i)}) < f_w(y^{(i)})$. En d'autres termes, il existe des éléments w de W_z pour lesquels le signe de la différence $f_w(x^{(i)}) - f_w(y^{(i)})$ est strictement positif, et d'autres pour lesquels il est strictement négatif. Un moyen simple de vérifier l'existence de tels éléments est de résoudre les deux programmes linéaires suivants :

$$\max_{w \in W_z} f_w(x^{(i)}) - f_w(y^{(i)})$$
$$\min_{w \in W_z} f_w(x^{(i)}) - f_w(y^{(i)})$$

Si les valeurs optimales des deux programmes linéaires ont le même signe (non nul) alors le polyèdre n'est pas coupé par la contrainte car cela signifie que tous les paramètres w de W_z modélisent la même préférence entre $x^{(i)}$ et $y^{(i)}$. Si, à l'inverse, les deux valeurs optimales n'ont pas le même signe alors tous les paramètres w de W_z ne modélisent pas la même préférence entre les deux solutions, ce qui signifie que le polyèdre W_z est coupé par la contrainte. Afin de déterminer le demi-espace auquel appartient un polyèdre W_z non coupé par la contrainte, il suffit de regarder le signe des valeurs optimales : si leur valeur est positive alors tous les éléments de W_z vérifient la nouvelle contrainte ($W_z \subseteq W_{x^{(i)} \succsim y^{(i)}}$), sinon les éléments de W_z ne vérifient pas la nouvelle contrainte ($W_z \cap W_{x^{(i)} \succsim y^{(i)}} = \emptyset$).

Ainsi, afin de mettre à jour la probabilité de chacune des solutions de $\mathcal{X}$, il faut résoudre au total $2|\mathcal{X}|$ programmes linéaires pour chaque question posée.

Optimisation du temps de calcul

Afin de limiter le nombre de programmes linéaires à résoudre, et donc de réduire le temps de calcul entre deux questions, on propose d'exploiter une approximation des polyèdres d'optimalité donnée par leur *boule de Chebyshev*. La boule de Chebyshev d'un polyèdre W_z est la plus petite boule contenant le polyèdre ; elle est caractérisée par son centre w_z^c et son rayon r_z, qui sont optimaux au sens du programme :

$$\min_{w_z^c, r_z} \{r_z : ||w_z^c - w||_2^2 \leq r_z^2, \ \forall w \in W_z\}$$

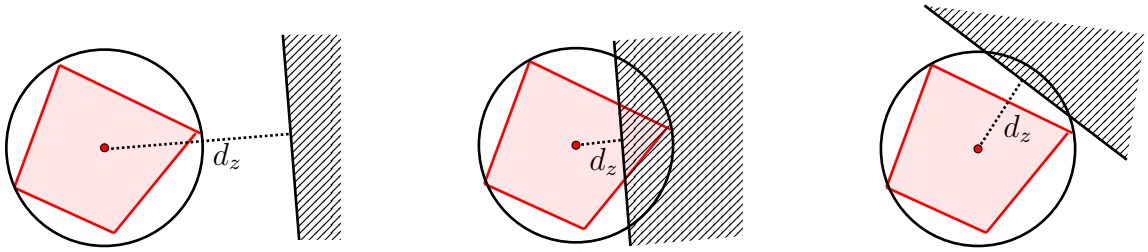
où $||x||_2$ désigne la norme euclidienne du vecteur x. La distance d_z du centre w_z^c à la contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$ est obtenue par la formule :

$$d_z = \frac{|\sum_{j=1}^{n} (u_j(x^{(i)}) - u_j(y^{(i)}))(w_z^c)_j|}{||u(x^{(i)}) - u(y^{(i)})||_2}$$

On peut alors distinguer deux cas :

- si $d_z \geq r_z$, alors le polyèdre W_z n'est pas coupé par la contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$ (voir l'exemple de la figure 5.3(a)) ; afin de déterminer si les éléments du polyèdre W_z vérifient ou non la contrainte, il suffit de vérifier si le centre w_z^c la vérifie ;

- si $d_z < r_z$, alors l'approximation n'est pas suffisante et un calcul exact par programmation linéaire est nécessaire. En effet, dans ce cas, le polyèdre peut être coupé par la contrainte ou non (voir les exemples des figures 5.3(b) et 5.3(c)).



(a) $d_z \geq r_z$ (b) $d_z < r_z$ (c) $d_z < r_z$

FIGURE 5.3 – Exemple d'une approximation de polyèdre par une boule Chebyshev.

Notons que les polyèdres d'optimalité restent inchangés durant tout le déroulement de l'algorithme ; la détermination de w_z^c et de r_z peut donc se faire en pré-traitement indépendamment de la procédure d'élicitation. Lorsqu'une contrainte $f_w(x^{(i)}) \geq f_w(y^{(i)})$ est déterminée par l'acquisition d'une nouvelle information préférentielle, seule la distance d_z et le signe de l'expression $f_{w_z^c}(x^{(i)}) - f_{w_z^c}(y^{(i)})$ sont calculés. Ainsi, l'utilisation des boules de Chebyshev peut permettre de réduire le temps de calcul entre deux questions sans impacter le résultat de la mise à jour Bayésienne.

5.4 Expérimentations numériques

Nous avons évalué l'efficacité de l'algorithme 5.1 en l'implémentant et en le testant [1] pour l'élicitation des paramètres d'une somme pondérée et d'un OWA. Les différents programmes linéaires sont résolus en utilisant la librairie gurobipy de Python.

Génération des instances et initialisation des paramètres de l'algorithme. Nous avons généré des instances de 100 solutions évaluées selon 5 critères. Pour cela, nous avons généré directement les vecteurs de performances $u(x)$ associées aux solutions $x \in \mathcal{X}$. Les valeurs de performances $u_i(x), i \in N$, ont été générées de manière à ce que les solutions des instances soient toutes potentiellement f_w-optimales (i.e., $W_x \neq \emptyset$ pour tout $x \in \mathcal{X}$). Pour cela, nous procédons, selon l'opérateur considéré, de la manière suivante :

- **Instances pour la somme pondérée WS.** Une alternative x de l'instance est générée comme suit (génération similaire à la génération d'instances pour un $\mathcal{X}$ explicite dans le chapitre précédent) : un vecteur y de taille 4 est généré uniformément dans $[0, 1]^4$. Ensuite, le vecteur de performances de x est obtenu en posant $u_i(x) = y_{(i)} - y_{(i-1)}$ pour tout $i = 1, \ldots, 5$, où $y_{(i)}$ est le i^e plus petit élément de y, $y_{(0)} = 0$ et $y_{(5)} = 1$. Les vecteurs ainsi générés sont tous sur le même hyperplan car $\sum_{i=1}^{5} u_i(x) = 1$ pour tout $x \in \mathcal{X}$. Ces solutions sont donc toutes Pareto optimales mais également potentiellement WS-optimales. Néanmoins, le nombre de

1. Implémentation en Python. Caractéristiques de la machine : Intel(R) Core(TM) i7-4790 CPU avec 15GB de RAM.

solutions potentiellement WS-optimales *uniques*[2] est très faible car un très grand nombre de polyèdres d'optimalité ne sont pas de pleine dimension. Afin d'obtenir des solutions potentiellement WS-optimales *uniques*, on applique, comme il a été suggéré par Li [1996], la fonction racine carrée à toutes les composantes $u_i(x)$ du vecteur de performances de chaque solution $x \in \mathcal{X}$ afin de rendre le front de Pareto concave. Un exemple des vecteurs de performances obtenus de cette manière dans le cas bicritère est donné sur la figure 5.4.

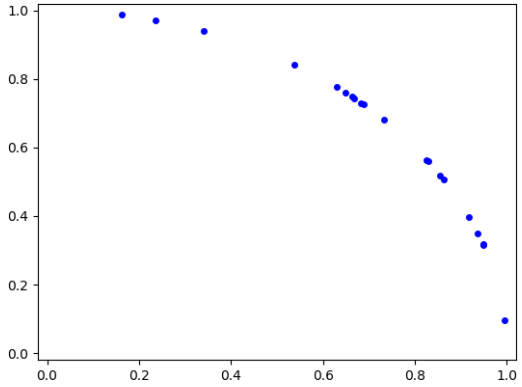


FIGURE 5.4 – Exemple d'instance pour WS avec 20 vecteurs de performances et 2 critères.

- **Instances pour OWA.** Rappelons d'abord quelques notions importantes : une alternative x est potentiellement OWA-optimale si son vecteur de *Lorenz* (définition 1.4) est potentiellement WS-optimal dans l'espace de Lorenz $\{L(x) \mid x \in \mathcal{X}\}$, où $L_i(x) = \sum_{j=1}^{i} u_{(j)}(x)$ avec $u_{(j)}(x)$ le j^e plus petit élément du vecteur de performances de x. On dit que le vecteur z est un vecteur de *Lorenz* s'il existe une solution x telle que $z = L(x)$. Étant donné un vecteur de Lorenz z, on note $L^{-1}(z)$ la solution x caractérisée par son vecteur de performances $u(x)$ telle que $L(x) = z$ (le vecteur de performances $u(x)$ est donc défini à une permutation de ses composantes près). Pour une telle solution x, on vérifie $u_{(i)}(x) = z_i - z_{i-1}$ pour tout $i = 1, \ldots, 5$, avec $z_0 = 0$.
  On définit maintenant le polyèdre suivant :

$$(P_{Lorenz}) : \begin{cases} y_{i+1} \geq y_i & \forall i \in \{0, \ldots, 4\} \quad (1) \\ (i+1)^2 y_{i+1} - i^2 y_i \geq i^2 y_i - (i-1)^2 y_{i-1} & \forall i \in \{1, \ldots, 4\} \quad (2) \\ \sum_{i=1}^{5} i^2 y_i = \sum_{i=1}^{5} i^2 & (3) \\ y_0 = 0 \end{cases}$$

  L'ensemble $\mathcal{L} = \{(i^2 y_i)_{i \in \{1,\ldots,5\}} : y \in (P_{Lorenz})\}$ contient des vecteurs qui sont tous des vecteurs de Lorenz grâce aux contraintes (1) et (2). De plus, ils appartiennent tous au même hyperplan grâce à la contrainte (3). Tous les vecteurs de $\mathcal{L}$ sont donc potentiellement WS-optimaux. Par conséquent, les éléments de l'ensemble $\{L^{-1}(z) : z \in \mathcal{L}\}$ sont potentiellement OWA-optimaux mais pas *uniques*. Comme nous l'avons fait plus haut, on peut appliquer la fonction racine carrée aux composantes des éléments z de $\mathcal{L}$ afin de résoudre ce problème. L'ensemble obtenu est

2. On parle ici d'unicité de la solution potentiellement optimale x dans le sens où il existe au moins une valeur de paramètre $w \in W_z$ telle que $f_w(x) > f_w(z), \forall z \in \mathcal{X}$.

donné par $\mathcal{L}' = \{(i\sqrt{y_i})_{i\in\{1,\dots,5\}} : y \in (P_{Lorenz})\}$. Finalement, tous les vecteurs dans $\mathcal{Y}' = \{L^{-1}(z) : z \in \mathcal{L}'\}$ sont potentiellement OWA-optimaux et uniques.
Une méthode de génération de solutions OWA-optimales peut donc être définie comme suit : pour générer un vecteur de performances $u(x)$ associé à une solution x, on génère d'abord un point y uniformément dans le polyèdre (P_{Lorenz}). Pour cela, il suffit de générer un vecteur de pondération α uniformément dans le simplexe de dimension m, où m est le nombre de points extrêmes du polyèdre défini par (P_{Lorenz}), puis de poser $y = \sum_{j=1}^{m} \alpha_j \hat{y}^j$, où $\hat{y}^1, \dots, \hat{y}^m$, sont les m points extrêmes du polyèdre. Le vecteur de performances de x est finalement obtenu en posant $u(x) = L^{-1}((i\sqrt{y_i})_{i\in\{1,\dots,5\}})$. Un exemple d'une telle instance dans le cas bicritère est donné sur la figure 5.5.

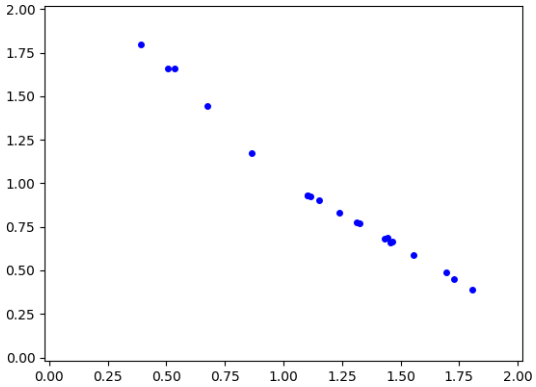


FIGURE 5.5 – Exemple d'instance pour OWA avec 20 vecteurs de performances et 2 critères

Une fois l'instance générée, nous appliquons l'algorithme 5.1 avec une estimation $\lambda = 0.8$, ce qui revient à supposer que le décideur se trompe avec une probabilité bornée supérieurement par 20%.

Simulation des interactions avec le décideur. Afin de simuler les interactions avec le décideur, un jeu de poids (caché) $\hat{w}$ est généré aléatoirement dans le simplexe. La réponse du décideur à la question i (comparaison entre $x^{(i)}$ et $y^{(i)}$) est donnée par le signe de $z^{(i)} = f_{\hat{w}}(x^{(i)}) - f_{\hat{w}}(y^{(i)}) + \varepsilon^{(i)}$, où $\varepsilon^{(i)}$ représente un bruit Gaussien modélisant l'erreur commise par le décideur. La valeur de $\varepsilon^{(i)}$ est donnée par un tirage aléatoire selon $\mathcal{N}(0, \sigma^2)$. On représentera différents taux d'erreur avec différentes valeurs de σ : le pourcentage d'erreur est d'autant plus grand que la valeur de σ est grande.

Avant de présenter les résultats des expérimentations que nous avons effectuées, nous donnons ci-dessous un exemple de déroulement de l'algorithme 5.1.

Exemple 5.3. *On déroule ici l'algorithme 5.1 sur une instance comportant* 10 *solutions* $x^1, \dots x^{10}$ *évaluées selon* 3 *critères. On suppose ici que les préférences du décideur sont modélisées par une somme pondérée* $\mathrm{WS}_{\hat{w}}$ *de paramètres* $\hat{w} = (0.5, 0.1, 0.4)$. *Le partitionnement de l'espace des paramètres correspond au partitionnement donné figure 5.6 où le point rouge représente* $\hat{w}$. *Notons que l'on représente l'ensemble* W *des jeux de paramètres possibles dans l'espace* (w_1, w_2) *car, les éléments de* W *étant normalisés, la* 3^e *composante*

peut être omise. Dans les figures qui suivent, la probabilité de chaque polyèdre d'optimalité est donnée en niveau de gris (plus le gris est foncé, plus la probabilité est élevée).

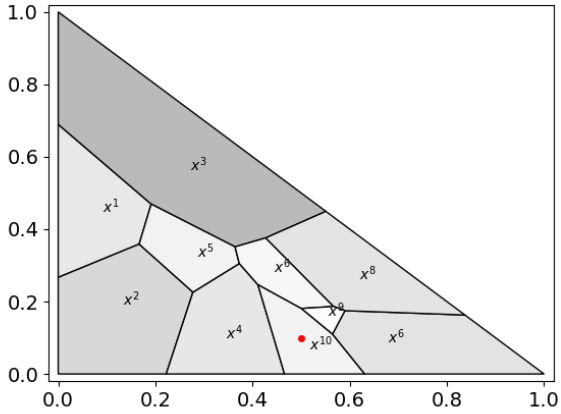


FIGURE 5.6 – Distribution de probabilité de la partition initiale.

Dans cet exemple, nous avons simulé les erreurs dans les réponses du décideur en utilisant un bruit Gaussien d'écart type $\sigma = 0.1$ *ce qui a mené à* 16% *de mauvaises réponses. On fixe ici* δ *à* 5% *de la valeur du MMER initiale.*

Initialement, le MMER *vaut* 0.152 *et la première question consiste à demander au décideur de comparer la recommandation courante (solution MMER) donnée par la solution* x^5 *à son meilleur adversaire (au sens du MER) donné par la solution* x^8*. Le décideur déclare préférer* x^8*, ce qui constitue une bonne réponse : on peut voir sur la figure 5.7 (à gauche) que le polyèdre d'optimalité de la solution* x^8 *se situe du bon côté (le même que* $\hat{w}$*) de l'hyperplan donné par* $f_w(x^8) - f_w(x^5) = 0$ *(représenté par une ligne rouge pointillée), contrairement au polyèdre d'optimalité de la solution* x^5 *qui se situe du mauvais côté. On constate alors sur la figure 5.7 (à droite) qu'après mise à jour de la distribution de probabilité, la probabilité des polyèdres d'optimalité se trouvant du bon côté de l'hyperplan* $f_w(x^8) - f_w(x^5) = 0$ *augmente (par rapport à la figure 5.6) :*

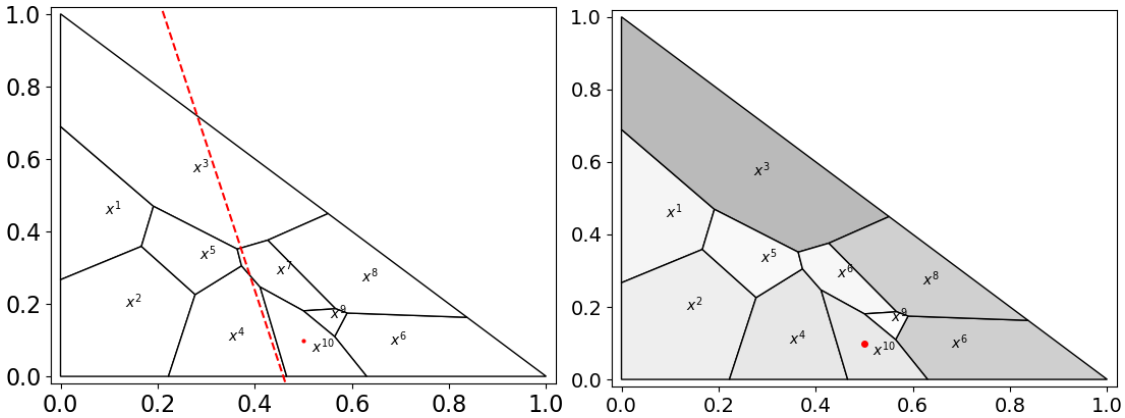


FIGURE 5.7 – Hyperplan induit par la question 1 (à gauche) et mise à jour de la distribution de probabilité (à droite).

Après cette mise à jour, la valeur du MMER *augmente légèrement à* $\text{MMER}(\mathcal{X}, W) = 0.156$*, par conséquent une autre question est posée. À l'issue de cette question, on obtient les probabilités représentées en niveau de gris sur la figure 5.8.*

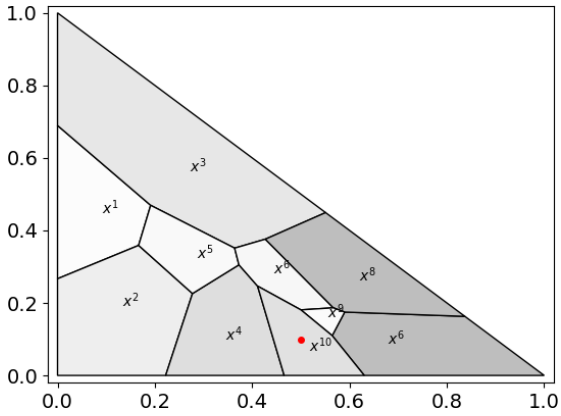


FIGURE 5.8 – Distribution de probabilité après la question 2.

À cette étape de l'algorithme, le MMER *vaut* $0.083 > 0.050 \times 0.162$*, la procédure se poursuit donc. À la question suivante, on demande au décideur de comparer les deux solutions* x^7 *et* x^2*. Le décideur donne alors une mauvaise réponse en déclarant préférer la solution* x^2 *: on peut en effet constater sur la figure 5.9 (à gauche) que le polyèdre d'optimalité associé à la solution* x^2 *ne se situe pas du même côté de l'hyperplan* $f_w(x^7) - f_w(x^2) = 0$ *que le poids* $\hat{w}$*, contrairement au polyèdre d'optimalité associé à la solution* x^7 *:*

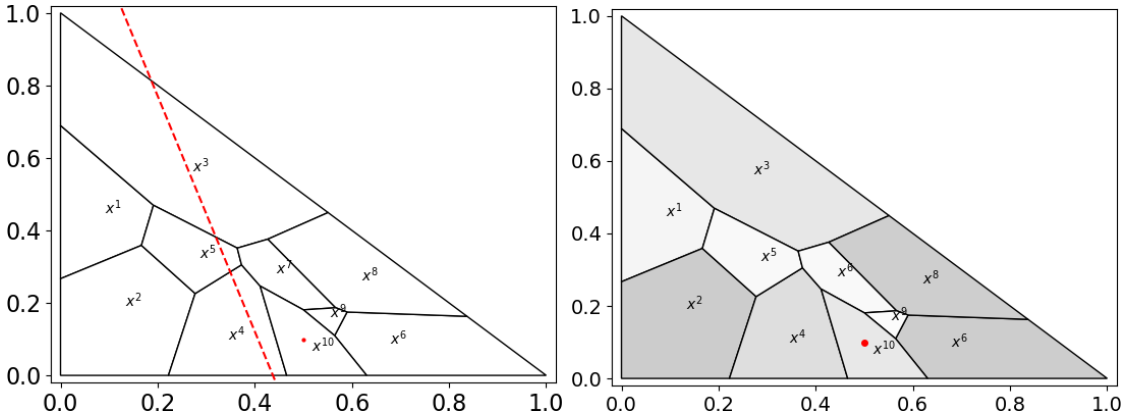


FIGURE 5.9 – Hyperplan induit par la question 3 (à gauche) et mise à jour de la distribution de probabilité (à droite).

Dans ce cas, on peut observer que les probabilités des polyèdres d'optimalité sont mises à jour de manière à ce que la probabilité des polyèdres d'optimalité incompatibles avec la préférence réelle du décideur augmente (figure 5.9 à droite). Ainsi, la valeur du MMER *augmente à la suite de cette nouvelle mise à jour puisqu'il vaut maintenant* 0.108*. La procédure continue ainsi, alternant questions et mises à jour de la distribution de probabilité, jusqu'à ce que le* MMER *atteigne la valeur* $0.007 < 0.05 \times 0.162$ *après la* 12^e *question. La distribution obtenue est donnée en niveau de gris sur la figure 5.10.*

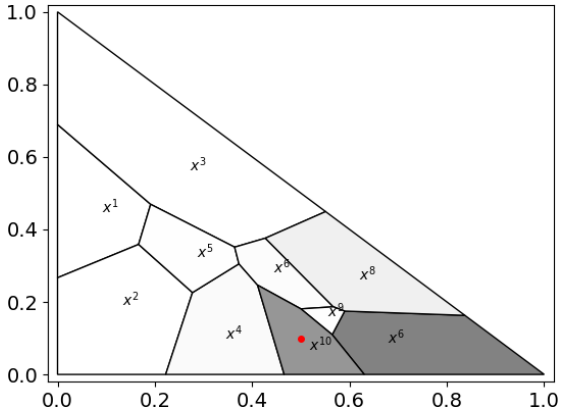


FIGURE 5.10 – Distribution de probabilité après la question 12.

Finalement, en utilisant les informations préférentielles induites par les réponses du décideur (dont 2 fausses) à ces 12 questions, on lui recommande la solution x^6 qui est en fait la deuxième solution préférée du décideur.

Analyse des résultats. Nous avons évalué l'efficacité de l'algorithme 5.1 en termes du rang réel (calculé à partir du jeu de poids caché) de la solution MMER (recommandation courante) : le rang 0 signifie que la solution MMER est optimale du point de vue du décideur, tandis que le rang 99 signifie que la solution MMER est la pire solution du point de vue du décideur. Pour évaluer la robustesse de l'approche par rapport à la présence d'erreurs dans les réponses, nous avons considéré différentes valeurs pour le paramètre σ : nous considérons une valeur nulle $\sigma = 0$ pour simuler un décideur qui répond toujours de manière correcte, tandis que des valeurs strictement positives $\sigma \in \{0.1, 0.2, 0.3\}$ modélisent différents taux d'erreurs. Pour les instances considérées, ces valeurs de σ ont mené à $3.6\%, 10\%$ et 22% (respectivement) de mauvaises réponses dans le cas de l'opérateur WS et à $3.2\%, 16\%$ et 25% (respectivement) de mauvaises réponses dans le cas de l'opérateur OWA. Notons que les valeurs de σ choisies ne correspondent pas forcément à l'a priori $1-\lambda$ sur le pourcentage d'erreurs, l'estimation donnée par $1-\lambda = 20\%$ (borne supérieure de la probabilité que le décideur donne une mauvaise réponse) est ici pessimiste lorsque $\sigma \in \{0.1, 0.2\}$ et optimiste lorsque $\sigma = 0.3$. Tous les résultats donnés sont des moyennes sur 40 instances.

Dans un premier temps, nous avons observé l'évolution du rang réel moyen (sur 40 instances) de la solution MMER après chaque question posée. La figure 5.11 donne l'évolution de ce rang moyen au cours de l'algorithme. Les courbes à gauche (respectivement à droite) de la figure sont celles obtenues pour WS (respectivement OWA) pour les différents taux d'erreurs considérés.

On peut observer que le rang moyen (sur 100 solutions) de la recommandation est, à partir de 14 questions :

- inférieur à 4 dans tous les cas,
- inférieur à 2 lorsque $\sigma < 0.3$ dans le cas de WS,
- inférieur à 2 pour toutes les valeurs de σ dans le cas d'OWA.

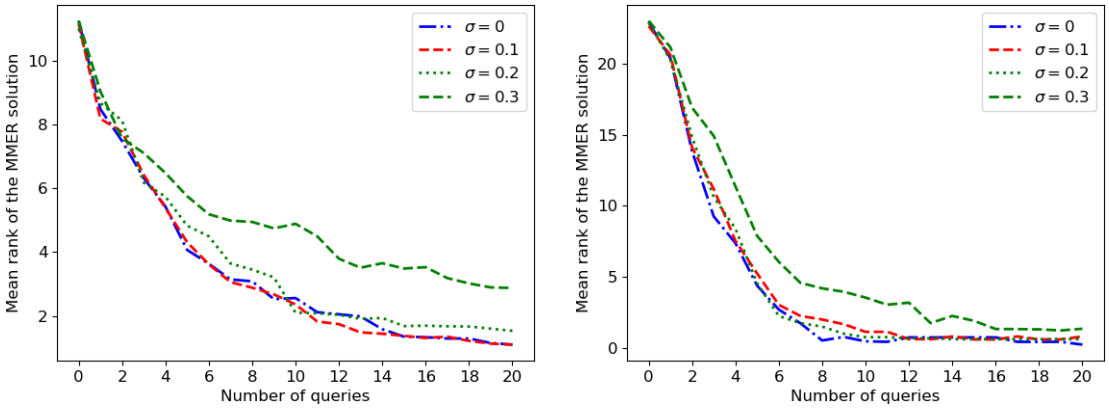


FIGURE 5.11 – Évolution du rang réel moyen de la solution MMER courante pour WS (à gauche) et pour OWA (à droite).

Sans surprise, on peut voir que la valeur de σ impacte négativement la qualité de la recommandation courante. Mais cet impact n'est significatif que lorsque le pourcentage d'erreur dépasse l'estimation $1-\lambda = 20\%$ et uniquement dans le cas de la somme pondérée. Même dans ce cas, la recommandation est relativement bonne puisqu'elle a un rang moyen inférieur à 4.

Nous présentons maintenant des résultats expérimentaux visant à montrer la pertinence de cette nouvelle approche par rapport à l'utilisation d'une méthode de résolution *"déterministe"* qui réduit progressivement l'espace des paramètres, sans prendre en compte la possibilité de la présence d'erreurs dans les réponses du décideur. Pour ce faire, nous avons appliqué l'algorithme 5.1 ainsi que l'algorithme 3.1 à un même ensemble de 40 instances et nous avons observé le rang réel de la solution MMER à la fin de l'exécution de l'algorithme (après 20 questions maximum). Les résultats de ces comparaisons sont illustrés par les boîtes à moustaches de la figure 5.12. Sur ces figures, les extrémités des moustaches donnent les rangs minimum et maximum obtenus ; les extrémités de la boîte donnent les valeurs des 1^{er} (en bas) et 3^{e} quartiles ; la ligne qui coupe la boîte représente la médiane tandis que la ligne pointillée représente la moyenne ; enfin, les cercles représentent des valeurs isolées. La figure de gauche donne la comparaison des performances pour WS tandis que la figure de droite donne la comparaison des performances pour OWA.

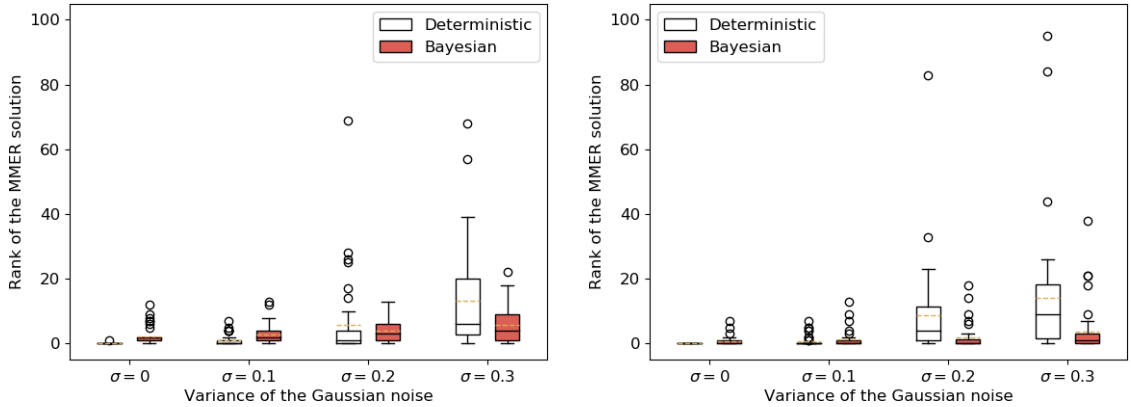


FIGURE 5.12 – Comparaison de l'approche déterministe (algorithme 3.1) et de l'approche Bayésienne (algorithme 5.1) - WS (à gauche) et pour OWA (à droite).

Clairement, l'approche Bayésienne est plus tolérante aux erreurs et offre de meilleures recommandations que l'algorithme 3.1. En effet, on peut voir que le taux d'erreur, contrôlé par σ, impacte très fortement la pertinence de la recommandation faite par l'algorithme 3.1. On peut voir que, dans le pire cas, le rang de la recommandation faite par ce dernier est 90, alors qu'il est de 40 (toujours dans le pire cas) lorsque l'on applique l'algorithme 5.1. On peut également voir que, lorsque $\sigma = 0.3$ et que l'on considère une modélisation des préférences par l'opérateur WS, l'algorithme 5.1 recommande, pour plus de 75% des instances, une solution ayant un rang < 10, ce qui est meilleur que le rang moyen obtenu avec l'algorithme 3.1. Pour une même valeur $\sigma = 0.3$ et une modélisation des préférences par l'opérateur OWA, l'algorithme 5.1 recommande, pour 75% des instances, une solution avec un rang < 3 et, pour 90% des instances, une solution ayant un rang < 8, ce qui est meilleur que le rang moyen obtenu avec l'algorithme 3.1. L'algorithme 5.1 est donc (pour des raisons évidentes) bien plus performant que l'algorithme 3.1 lorsque des erreurs sont présentes dans les réponses du décideur.

Pour finir, les temps de calcul entre deux questions sont inférieurs à 1 seconde dans tous les cas. En comparaison avec l'approche de l'algorithme 4.1 du chapitre précédent, on peut observer une performance équivalente (voire meilleure) en terme de qualité de la recommandation finale, pour un temps de calcul inférieur. Il est donc plus intéressant, du point de vue du temps de calcul, d'utiliser cette approche. Néanmoins, contrairement à l'approche proposée dans le chapitre 4, la possibilité d'étendre cette approche à un ensemble de solutions défini sur domaine combinatoire reste à étudier.

5.5 Conclusion du chapitre

Dans ce chapitre, nous nous sommes intéressés à la conception d'algorithmes d'élicitation incrémentale qui, comme dans le chapitre précédent, prennent en compte la possibilité de la présence d'erreurs ou d'incohérences dans les réponses du décideur. Nous nous sommes plus particulièrement intéressés à la conception d'un algorithme utilisant une modélisation de l'incertitude (concernant le système de valeurs du décideur) plus simple et plus rapide à manipuler qu'une distribution de probabilité *continue*. Pour cela, nous avons discrétisé l'espace des paramètres afin de manipuler une distribution de probabilité *discrète* simple à mettre à jour lors de l'acquisition de nouvelles informations préférentielles. La discrétisation de l'espace des paramètres consiste à le partitionner en polyèdres d'optimalité (ensemble d'instanciations des paramètres pour lesquelles une solution est optimale). On associe ensuite à chaque élément de cette partition une valeur indiquant la probabilité que le jeu de paramètres modélisant les préférences du décideur appartienne à ce polyèdre d'optimalité. En d'autres termes, cette distribution de probabilité indique la probabilité que chaque solution soit optimale. En utilisant cette représentation des connaissances concernant les préférences du décideur, nous introduisons une méthode d'élicitation incrémentale fondée sur une nouvelle adaptation de la stratégie CSS à ce nouveau contexte, ainsi qu'une nouvelle méthode de mise à jour Bayésienne de la distribution de probabilité.

La méthode que nous introduisons dans ce chapitre est générique et s'adapte à toute fonction d'agrégation linéaire en ses paramètres. Nous avons, en particulier, testé son efficacité pour les deux opérateurs WS et OWA en présence de taux d'erreurs plus ou moins

forts. Cette approche est très efficace du point de vue de la qualité de la recommandation mais également du temps de calcul. En effet, elle consiste à manipuler une distribution de probabilité discrète simple et rapide à mettre à jour. De plus, elle utilise une définition des regrets permettant de faire une partie non négligeable des calculs en amont de la méthode d'élicitation ; ainsi, les calculs à effectuer entre deux questions sont minimes. Néanmoins, contrairement à l'approche du chapitre précédent, on ne sait pas encore s'il est possible d'adapter l'approche à la résolution de problèmes définis sur domaine combinatoire. En effet, d'une part, il n'est pas possible de déterminer la totalité des polyèdres d'optimalité en temps polynomial, et d'autre part, la mise à jour de leur valeur de probabilité à chaque acquisition d'une nouvelle information préférentielle prendrait également un temps exponentiel.

Une extension intéressante de ce travail serait donc de concevoir une approche voisine que l'on puisse étendre de manière opérationnelle à des problèmes définis sur domaine combinatoire. Une idée naturelle serait de commencer l'algorithme sans partitionner l'espace des paramètres, puis de le découper au fur et à mesure de l'obtention de nouvelles informations préférentielles. Néanmoins, cette idée ne semble pas prometteuse car, d'une part, le fait de couper les polyèdres existants à une étape donnée lorsqu'une nouvelle information est disponible formerait des polyèdres d'ordre partiel [3] sur les solutions (au lieu de polyèdres d'optimalité), ce qui représente vite un très grand nombre de polyèdres à manipuler. De plus, de tels polyèdres apporteraient une information beaucoup plus riche (et donc plus coûteuse à éliciter) que celle réellement nécessaire à l'obtention d'une bonne recommandation. D'autre part, la construction progressive de polyèdres d'optimalité (et non d'ordre) n'est pas possible en temps polynomial : une zone d'optimalité est définie par $W_x = \{w \in W | f_w(x) \geq f_w(y), \forall y \in \mathcal{X}\}$ ce qui représente un nombre exponentiel de contraintes lorsque $\mathcal{X}$ contient un nombre exponentiel de solutions. En outre, cela constituerait une partition contenant des polyèdres non convexes pour lesquels le calcul des regrets par paires n'est pas évident.

Enfin, une voie d'amélioration intéressante de ce travail serait de trouver une meilleure approximation des polyèdres d'optimalité que les boules de Chebyshev (rappelons que leur emploi permet d'optimiser les temps de calcul requis pour déterminer la vraisemblance de l'optimalité d'une solution pour une réponse donnée) qui, selon la forme des polyèdres, peuvent dans certains cas constituer une approximation relativement grossière, comme par exemple dans le cas des polyèdres d'optimalité de la figure 5.1 pour OWA. Une telle amélioration n'est pas évidente car l'obtention d'une meilleure approximation ne doit pas se faire au détriment du gain en termes de temps de calcul.

3. Ce que l'on entend ici par polyèdre d'*ordre partiel* est un ensemble convexe de jeux de poids w pour lesquels les préférences induites par f_w correspondent à un même ordre partiel des solutions de $\mathcal{X}$. En d'autres termes, un polyèdre d'ordre $W_{\mathcal{Z},\mathcal{Z}'}$ est un ensemble de jeux de poids tel que $\forall (w, w') \in W^2_{\mathcal{Z},\mathcal{Z}'}, \forall (x, y) \in \mathcal{Z} \times \mathcal{Z}', f_w(x) \geq f_w(y) \Leftrightarrow f_{w'}(x) \geq f_{w'}(y)$. Un polyèdre d'optimalité W_x est donc un polyèdre d'ordre partiel $W_{\mathcal{Z},\mathcal{Z}'}$ tel que $\mathcal{Z} = \{x\}$ et $\mathcal{Z}' = \mathcal{X}$.

Conclusion et perspectives

Dans cette thèse, nous nous sommes intéressés à la conception de méthodes d'élicitation incrémentale pour la prise de décision multicritère, multi-agents ou dans le risque, principalement dans un cadre où l'ensemble des alternatives possibles est défini sur domaine combinatoire. Ces méthodes sont fondées sur l'utilisation de modèles de décision dont les paramètres modélisent le compromis que fait le décideur entre les différents objectifs (critères, agents ou scénarios). Éliciter le comportement décisionnel du décideur revient alors à réduire l'incertitude concernant la valeur des paramètres qui convient à la modélisation de ses préférences. Les méthodes que nous introduisons consistent à alterner questions préférentielles et exploration de l'espace des solutions afin d'exploiter au mieux les informations apportées par l'élicitation dans l'exploration de l'espace des solutions et vice versa, et ce, jusqu'à pouvoir déterminer une recommandation de bonne qualité. Notons que nous nous sommes efforcés, tout au long de cette thèse, de concevoir des méthodes génériques pouvant s'appliquer à un large spectre de problèmes d'optimisation, et nous n'avons pas cherché à les spécialiser dans le cas de problèmes particuliers. Dans ce contexte, nos contributions peuvent être divisées en trois parties principales :

- *L'élicitation de modèles complexes pour la résolution de problèmes définis sur domaine combinatoire* : dans un premier temps, nous nous sommes intéressés à la conception de méthodes permettant de résoudre des problèmes de choix multi-objectifs combinant les difficultés liées à : la *nature combinatoire du problème*, la *connaissance partielle des préférences du décideur*, et l'utilisation d'un *modèle non-linéaire* pour représenter les préférences du décideur. Les modèles d'agrégation non-linéaires tels qu'OWA (*Ordered Weighted Average*) et WOWA (*Weighted OWA*) offrent un grand pouvoir descriptif pour la modélisation des préférences en présence d'objectifs multiples (souvent conflictuels). Néanmoins, leur pouvoir descriptif vient avec un coût computationnel plus important que pour des modèles linéaires. En effet, on explicite dans le chapitre 1 les difficultés liées à l'optimisation de la valeur d'une fonction non-linéaire lorsque le problème porte sur un nombre exponentiel de solutions, et ce, même lorsque la valeur des paramètres du modèle est fixée. Cette optimisation est naturellement encore plus difficile lorsque la valeur des paramètres n'est pas précisément connue. Nous avons introduit, dans le chapitre 2, deux familles de méthodes fondées sur une énumération partielle de l'espace des solutions permettant de se ramener, par étapes, à des sous-ensembles explicites de solutions afin de contourner les difficultés posées par la nature combinatoire des problèmes considérés. Nous avons ensuite introduit, dans le chapitre 3, une méthode qui traite le problème sous un angle différent, et qui est fondée sur une exploration de l'espace des solutions guidée par une exploration de l'espace des paramètres. Pour cela, nous exploitons la linéarité des modèles décisionnels *en leurs paramètres* ainsi que la convexité de l'espace des paramètres. Les algorithmes introduits dans les chapitres 2 et 3 sont fondés sur une élicitation par réduction systématique de

l'espace des paramètres ; à chaque fois qu'une question est posée au décideur, sa réponse est formalisée en une contrainte qui exclut toutes les valeurs de paramètres qui ne sont pas compatibles avec la réponse donnée. Ces méthodes sont très efficaces en pratique car elles permettent de déterminer une solution optimale malgré une connaissance des valeurs des paramètres encore très imprécise.

- *L'élicitation de modèles complexes en présence de réponses incohérentes* : l'inconvénient des méthodes procédant par réductions successives de l'espace des paramètres est l'aspect irréversible dans l'ajout des contraintes induites par les réponses du décideur. En effet, cet aspect rend ces méthodes très sensibles à la présence d'erreurs (par rapport à son propre système de valeurs) ou d'incohérences dans les réponses du décideur. Nous montrons à l'aide d'un exemple très simple, en début de chapitre 4, que la recommandation faite peut être de très mauvaise qualité lorsque le décideur présente des erreurs/incohérences dans ses réponses. Or, il peut arriver dans des situations réelles que le décideur se trompe en répondant à une question du fait de sa difficulté (par exemple, à cause de solutions à comparer trop proches ou, à l'inverse, trop différentes) ; il peut également arriver que ses préférences évoluent au cours du temps, au fur et à mesure qu'il découvre de nouvelles alternatives. Dans ces cas là, des méthodes d'élicitation prenant en compte les réponses du décideur de manière trop stricte et irréversible comme celles des chapitres 2 et 3 ne conviennent pas. D'autres part, les questions posées dans ces méthodes ne laissent pas au décideur l'opportunité de revenir sur une préférence qu'il aurait déclarée précédemment, ce qui écarte toute possibilité de rectifier le modèle.

Nous nous sommes alors naturellement intéressés à la conception de méthodes d'élicitation incrémentales permettant de prendre en compte la possibilité d'incohérences dans les réponses du décideur. Dans un premier temps, notre intérêt s'est porté sur la conception de telles méthodes pour la prise de décision sur domaine explicite. La prise en compte des incohérences n'est déjà pas évidente dans ce cas, car nos méthodes doivent pouvoir permettent de prendre en compte les réponses du décideur pour réduire l'incertitude concernant la valeur des paramètres, mais sans trop impacter la qualité de la recommandation en cas d'erreurs. Nous proposons dans les chapitres 4 et 5 deux méthodes d'élicitation des paramètres de modèles complexes (OWA et intégrale de Choquet) prenant en compte la présence d'erreurs éventuelles de deux manières différentes : la première passe par la gestion d'une distribution de probabilité continue associée à l'espace des paramètres, alors que la seconde passe par la gestion d'une distribution de probabilité discrète. À chaque fois que l'on pose une question au décideur, sa réponse permet de mettre à jour la distribution de probabilité de manière à augmenter la probabilité des valeurs de paramètres compatibles avec l'information préférentielle induite par cette réponse. Ainsi, aucune valeur de paramètres n'est exclue de l'espace des paramètres de manière définitive à la suite d'une interaction avec le décideur.

- *L'élicitation incrémentale en présence de réponses incohérentes pour la résolution de problèmes définis sur domaine combinatoire* : l'adaptation des méthodes introduites dans les chapitres 4 et 5 au domaine combinatoire n'est pas évidente car (entre autres) la stratégie du choix de la question repose sur l'énumération implicite de toutes les questions possibles afin de déterminer *la* question *la plus informative*. Effectuer ces comparaisons lorsque l'espace des solutions est défini en compréhension et qu'il possède une structure combinatoire peut nécessiter un effort de calcul prohibitif. Dans le chapitre 4, nous introduisons un nouvel algorithme permettant de surmonter cette difficulté à l'aide d'une

méthode de génération de contraintes et de variables permettant de comparer uniquement un sous-ensemble de solutions réalisables. Dans ce chapitre, nous illustrons notre méthode en utilisant un modèle linéaire : la somme pondérée. Notons tout de même que ces méthodes peuvent s'appliquer à des modèles non-linéaires possédant une linéarisation efficace. En pratique, l'élicitation de modèles non-linéaires peut engendrer des temps de calculs plus longs car les linéarisations de ces modèles entraînent l'introduction de variables et de contraintes additionnelles

À la suite de cette thèse, plusieurs pistes de recherche peuvent être envisagées. On présente ici quelques perspectives qui nous semblent pertinentes :

- *La détermination d'une borne théorique sur le nombre de questions posées* : tout au long de cette thèse, nous avons pu montrer l'efficacité empirique de nos méthodes de résolution en termes de qualité de la recommandation faite et du nombre de questions nécessaires à la détermination de cette recommandation. Néanmoins, nous n'avons (souvent) pas abordé la question de la détermination d'une borne supérieure *efficace* sur le nombre de questions posées (en fonction de la taille de l'instance et de la dimension du paramètre du modèle considéré). Or, il peut être pratique pour le décideur de savoir, à l'avance, quelle sera la longueur maximale du questionnaire. Nous avons pu déterminer, dans le chapitre 3, un questionnaire particulier permettant de borner supérieurement le nombre de questions nécessaires à la détermination d'une recommandation δ-optimale, pour un seuil d'approximation $\delta > 0$ fixé. Néanmoins, les essais numériques montrent que cette stratégie est en pratique moins informative que la stratégie CSS [Wang et Boutilier, 2003] que nous utilisons principalement dans cette thèse. Cependant, il n'existe pas, à notre connaissance, de borne théorique efficace connue. Il existe dans la littérature des travaux visant à déterminer des questionnaires permettant d'optimiser le nombre de questions posées, e.g., [Gilbert *et al.*, 2015], mais les résultats théoriques bornant supérieurement le nombre de questions sont beaucoup moins nombreux. Nous pourrions également nous poser la question du nombre minimum de questions à poser afin de déterminer une solution préférée. Lorsque les questions posées sont des questions de comparaisons par paires, une borne pourrait, par exemple, être obtenue par le calcul du nombre de faces minimum d'un polyèdre d'optimalité associé à une solution réalisable. Néanmoins, cette solution n'est pas facilement envisageable sur domaine combinatoire.

- *La détermination d'une garantie théorique sur la qualité de la recommandation* : les travaux que nous avons introduits dans les chapitres 2 et 3 offrent la possibilité de borner le regret réel[4] de la recommandation faite par l'algorithme. Par contre, nous n'avons pas pu établir de telles bornes pour les méthodes introduites dans les chapitres 4 et 5. En effet, lorsque l'élicitation des préférences est fondée sur la mise à jour d'une distribution de probabilité, il semble être très difficile de déterminer des bornes théoriques sur la qualité de la recommandation dans la mesure où l'on considère que des incohérences peuvent être présentes dans les réponses du décideur. Néanmoins, il serait intéressant de pouvoir déterminer, par exemple, la probabilité que la recommandation soit optimale après avoir posé q questions (où q est préalablement fixé), ou encore, la probabilité que le regret réel de la recommandation soit d'au plus δ ($\delta > 0$) après avoir posé q questions. Il existe

4. En notant f_ω le modèle de décision, $\hat{\omega}$ le paramètre (caché) modélisant les préférences du décideur et $\hat{x}$ la solution $f_{\hat{\omega}}$-optimale, on rappelle que le regret réel de la recommandation x^* correspond à la distance $f_{\hat{\omega}}(\hat{x}) - f_{\hat{\omega}}(x^*)$ (en maximisation).

des résultats proches ; par exemple, Nowak [2009] établit, dans un cadre de classification binaire, une borne sur la probabilité que la fonction de classification trouvée soit erronée. Ce résultat est très intéressant puisque la borne décroît lorsque le nombre de questions augmente ; néanmoins, il ne s'adapte pas directement à notre stratégie du choix de la question car l'auteur utilise un questionnaire très particulier.

- *La détermination de la stratégie du choix de la question* : les travaux présentés dans cette thèse sont essentiellement fondés sur l'utilisation (ou l'adaptation) du critère de choix du minimax regret [Savage, 1954] et de la stratégie CSS [Wang et Boutilier, 2003] pour le choix de la question à poser. L'utilisation des regrets offre de nombreux avantages tels que (dans le cas déterministe) le fait de définir une borne sur le regret réel, le fait que cette borne décroît au fur et à mesure des questions posées (ce qui permet de définir un critère d'arrêt), ou encore le fait de poser des questions qui ne peuvent pas mener à un ensemble de paramètres vide (cf. chapitre 2 pour plus de détails). Néanmoins, elle traduit une vision relativement pessimiste consistant à vouloir se prémunir contre un décideur qui aurait toujours la préférence "la plus éloignée" de celle correspondant à la recommandation faite (principe du regret maximum). De plus, les questions posées à l'aide de cette stratégie peuvent parfois être difficiles à répondre pour le décideur car les solutions sont trop proches, ou à l'inverse, trop différentes. On pourrait alors vouloir déterminer un autre critère de choix ainsi qu'une autre stratégie du choix de la question ; il existe d'ailleurs plusieurs autres stratégies dans la littérature (par exemple, [Chajewska *et al.*, 2000; Boutilier, 2002]), mais elles ne possèdent pas forcément les mêmes avantages que la stratégie CSS. Néanmoins, utiliser autre chose que les regrets maximums pour déterminer une recommandation ou une question à poser entraîne la possibilité de perdre la borne (décroissante) du regret réel de la recommandation. Une solution permettant d'éviter le côté pessimiste du critère de choix fondé sur le minimax regret ainsi que de la stratégie CSS, tout en gardant un maximum de leurs avantages, pourrait être de remplacer les regrets maximums par des regrets moyens (dans le même esprit que les regrets espérés), puis d'utiliser le regret maximum pour déterminer le regret réel d'une recommandation (puisque la moyenne ne peut pas donner une borne de ce regret).

- *L'élicitation du modèle de décision* : dans cette thèse, nous supposons que nous disposons de suffisamment d'informations concernant les préférences du décideur pour pouvoir déterminer le modèle de décision qui convient le mieux pour les modéliser. Or, cette hypothèse n'est pas toujours vraie : il peut arriver que le décideur ne nous apporte pas suffisamment d'informations ou que ses préférences évoluent au cours du temps. Par exemple, le décideur peut, dans un premier temps, se soucier uniquement de la notion d'équilibre des performances sur les différents objectifs, on opte alors pour une modélisation à l'aide d'un OWA. Puis, au fur et à mesure qu'il découvre de nouvelles alternatives (grâce aux questions que l'on pose), il réalise qu'il associe une certaine importance à une (ou plusieurs) coalition(s) d'objectifs : une intégrale de Choquet s'avérerait alors plus pertinente, mais le modèle est déjà fixé et nos méthodes ne permettent pas de le changer en cours de résolution. Une première solution serait de toujours prendre le modèle le plus général possible, mais si le modèle est trop sophistiqué pour les préférences du décideur, ce choix peut entraîner des difficultés computationnelles inutiles. On peut alors imaginer des approches ou le *modèle* ainsi que ses *paramètres* sont à éliciter. Le modèle courant avec la richesse descriptive qui l'accompagne doivent alors être adaptés en fonction de leur cohérence avec les réponses du décideur. Or, la détection d'incohérences n'est pas évidente selon la stratégie du choix

de la question à poser ; en utilisant la stratégie CSS, on ne donne jamais au décideur l'opportunité de se contredire. On se pose alors naturellement la question de la détermination de ce que l'algorithme doit faire, lors de l'acquisition d'une nouvelle information préférentielle, entre mise à jour de la connaissance des valeurs des paramètres, et remise en question du modèle. Notons que dans ce type d'approches, un aspect très difficile à prendre en compte serait alors la présence d'erreurs ou d'incohérences (si le questionnaire les autorise) dans les réponses du décideur. En effet, il est, dans ce cas, difficile d'interpréter une erreur ou une incohérence détectée : on peut difficilement dire si l'incohérence provient d'un choix de modèle de décision inapproprié, ou si elle correspond à une erreur dans le jugement du décideur.

Ces perspectives semblent être des pistes de recherche intéressantes à explorer, à court et moyen terme, dans la continuité de cette thèse. À plus long terme, d'autres perspectives, plus éloignées du travail fait durant cette thèse, pourraient être envisagées. On pourrait, par exemple, s'intéresser à l'élicitation de préférences non-transitives, ce qui nécessiterait d'aborder la modélisation ainsi que l'élicitation d'une manière complètement différente.

Annexe A

Opérations sur des distributions Gaussiennes

Cette annexe rappelle des propriétés concernant la somme et le produit de deux distributions Gaussiennes qu'on utilise dans la preuve de la proposition 4.2 :

A.1 Produit de deux distributions Gaussiennes multivariées

Soit X une variable aléatoire multidimensionnelle, et soient $\mathcal{N}_X(\mu_1, S_1)$ et $\mathcal{N}_X(\mu_2, S_2)$ deux distributions Gaussiennes multivariées (portant sur X) de moyennes μ_1 et μ_2 (respectivement) et de matrices de variance-covariance S_1 et S_2 (respectivement). Le produit des deux distributions Gaussiennes multivariées mène à une troisième distribution qui est également une Gaussienne multivariée [Ahrendt, 2005] :

$$\mathcal{N}_X(\mu_1, S_1) \times \mathcal{N}_X(\mu_2, S_2) \propto \mathcal{N}_X(\mu_3, S_3)$$

avec

$$\begin{aligned} S_3 &= (S_1^{-1} + S_2^{-1})^{-1} \\ \mu_3 &= S_3(S_1^{-1}\mu_1 + S_2^{-1}\mu_2) \end{aligned}$$

On utilise cette propriété dans la preuve de la proposition 4.2 afin de déterminer la densité de la distribution conditionnelle $\omega|z_j, r^{(i)}$ qui s'obtient par le produit des densités conditionnelles $z_j|\omega, r^{(i)}$ et $\omega|r^{(i)}$, que l'on réécrit comme deux distributions Gaussiennes de la variable ω :

$$p(\omega|z_j, r^{(i)}) \propto \mathcal{N}(\hat{\omega}, \sigma^2(d^{(i)}{d^{(i)}}^T)^{-1}) \times \mathcal{N}(\mu_k, \Sigma_k) \times \mathbb{1}_{z_j, r^{(i)}} \times \mathbb{1}_{\omega \in \mathcal{W}}$$

Ainsi $\omega|z_j, r^{(i)} \sim \mathcal{N}(\mu^j, S)$ avec :

$$\begin{aligned} S^{-1} &= \frac{1}{\sigma^2} d^{(i)}{d^{(i)}}^T + S_k^{-1} \\ \mu^j &= S(\frac{1}{\sigma^2}{d^{(i)}}^T z_j + S_k^{-1}\mu_k) \end{aligned}$$

A.2 Somme de deux distributions Gaussiennes multivariées

Soient X_1 et X_2 deux variables aléatoires multidimensionnelles suivant chacune une loi Gaussienne multivariée. On note $X_1 \sim \mathcal{N}(\mu_1, S_1)$ et $X_2 \sim \mathcal{N}(\mu_2, S_2)$ où μ_i est le vecteur moyenne de la densité de probabilité associée à X_i et S_i est la matrice de variance covariance associée à X_i. Si les deux variables X_1 et X_2 sont indépendantes, alors leur somme représente également une variable aléatoire Gaussienne multivariée [Ahrendt, 2005] :

$$\mathcal{N}(\mu_1, S_1) + \mathcal{N}(\mu_2, S_2) \propto \mathcal{N}(\mu_3, S_3)$$

avec

$$\begin{aligned} S_3 &= S_1 + S_2 \\ \mu_3 &= \mu_1 + \mu_2 \end{aligned}$$

On utilise cette propriété dans la preuve de la proposition 4.2 afin de déterminer la densité de la distribution p_{k+1} qui s'obtient par la somme des densités Gaussiennes $\omega|z_j, r^{(i)}$, pour tout $j \in \{1, \ldots, m\}$.

Bibliographie

Ahrendt, P. (2005). The multivariate gaussian probability distribution. *Technical University of Denmark, Tech. Rep.* (Citée pages 203 et 204.)

Albert, J. H. et Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *Journal of the American statistical Association*, 88(422):669–679. (Citée pages 138 et 143.)

Amit, D. J., Wong, K. M. et Campbell, C. (1989). Perceptron learning with sign-constrained weights. *Journal of Physics A : Mathematical and General*, 22(12):2039. (Citée page 46.)

Angilella, S., Corrente, S. et Greco, S. (2015). Stochastic multiobjective acceptability analysis for the choquet integral preference model and the scale construction problem. *European Journal of Operational Research*, 240(1):172–182. (Citée pages 48 et 177.)

Benabbou, N., Leroy, C. et Lust, T. (2020). An interactive regret-based genetic algorithm for solving multi-objective combinatorial optimization problems. *In Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*. (Citée pages 13 et 63.)

Benabbou, N. et Perny, P. (2015a). Combining preference elicitation and search in multiobjective state-space graphs. *In Proceedings of IJCAI-15*, pages 297–303. (Citée pages 13, 56 et 62.)

Benabbou, N. et Perny, P. (2015b). Incremental weight elicitation for multiobjective state space search. *In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1093–1099. (Citée pages 12, 13, 55, 62, 65, 66 et 67.)

Benabbou, N. et Perny, P. (2015c). On possibly optimal tradeoffs in multicriteria spanning tree problems. *In International Conference on Algorithmic DecisionTheory*, pages 322–337. Springer. (Citée pages 13 et 55.)

Benabbou, N. et Perny, P. (2016). Solving multi-agent knapsack problems using incremental approval voting. *In Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 1318–1326. (Citée page 56.)

Benabbou, N. et Perny, P. (2017). Adaptive elicitation of preferences under uncertainty in sequential decision making problems. *In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*. (Citée pages 13, 54 et 62.)

Benabbou, N., Perny, P. et Viappiani, P. (2014). Incremental elicitation of choquet capacities for multicriteria decision making. *In ECAI*, pages 87–92. (Citée page 54.)

Benabbou, N., Perny, P. et Viappiani, P. (2017). Incremental elicitation of choquet capacities for multicriteria choice, ranking and sorting problems. *Artificial Intelligence*, 246:152–180. (Citée pages 54 et 62.)

Benade, G., Nath, S., Procaccia, A. D. et Shah, N. (2017). Preference elicitation for participatory budgeting. *In Proceedings of AAAI*, pages 376–382. (Citée pages 56 et 63.)

Benayoun, R., De Montgolfier, J., Tergny, J. et Laritchev, O. (1971). Linear programming with multiple objective functions : Step method (stem). *Mathematical programming*, 1(1):366–375. (Citée page 50.)

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer. (Citée pages 58, 148 et 165.)

Boland, N., Domínguez-Marín, P., Nickel, S. et Puerto, J. (2006). Exact procedures for solving the discrete ordered median problem. *Computers & Operations Research*, 33(11):3270–3300. (Citée page 41.)

Bourdache, N. et Perny, P. (2017). Anytime algorithms for adaptive robust optimization with owa and wowa. *In International Conference on Algorithmic DecisionTheory*, pages 93–107. Springer. (Citée page 61.)

Bourdache, N. et Perny, P. (2019). Active preference learning based on generalized gini functions : Application to the multiagent knapsack problem. *In Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7741–7748. (Citée pages 61 et 117.)

Bourdache, N., Perny, P. et Spanjaard, O. (2019a). Active preference elicitation by bayesian updating on optimality polyhedra. *In International Conference on Scalable Uncertainty Management*, pages 93–106. Springer. (Citée page 175.)

Bourdache, N., Perny, P. et Spanjaard, O. (2019b). Incremental elicitation of rank-dependent aggregation functions based on bayesian linear regression. *In Proceedings of IJCAI'19*, pages 2023–2029. (Citée page 135.)

Bourdache, N., Perny, P. et Spanjaard, O. (2020). Bayesian preference elicitation for multiobjective combinatorial optimization. *arXiv preprint arXiv :2007.14778*. (Citée page 135.)

Boutilier, C. (2002). A pomdp formulation of preference elicitation problems. *In AAAI/IAAI*, pages 239–246. Edmonton, AB. (Citée pages 54 et 200.)

Boutilier, C., Patrascu, R., Poupart, P. et Schuurmans, D. (2006a). Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, 170(8-9):686–713. (Citée pages 13, 54, 55, 78, 80 et 118.)

Boutilier, C., Patrascu, R., Poupart, P. et Schuurmans, D. (2006b). Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, 170(8-9):686–713. (Citée pages 62, 142 et 181.)

Boutilier, C., Regan, K. et Viappiani, P. (2009). Online feature elicitation in interactive optimization. *In Proceedings of the 26th Annual International Conference on Machine Learning*, pages 73–80. (Citée page 54.)

Branke, J., Corrente, S., Greco, S., Słowiński, R. et Zielniewicz, P. (2016). Using choquet integral as preference model in interactive evolutionary multiobjective optimization. *European Journal of Operational Research*, 250(3):884–901. (Citée page 57.)

Braziunas, D. et Boutilier, C. (2007). Minimax regret based elicitation of generalized additive utilities. *In Proc. of UAI-07*, pages 25–32. (Citée pages 13, 54, 62 et 80.)

Bremner, D., Fukuda, K. et Marzetta, A. (1998). Primal—dual methods for vertex and facet enumeration. *Discrete & Computational Geometry*, 20(3):333–357. (Citée page 122.)

Brero, G., Lubin, B. et Seuken, S. (2018). Combinatorial auctions via machine learning-based preference elicitation. *In Proceedings of IJCAI'18*, pages 128–136. (Citée pages 13 et 62.)

Chajewska, U., Koller, D. et Parr, R. (2000). Making rational decisions using adaptive utility elicitation. *In Proceedings of AAAI-00*, pages 363–369. (Citée pages 13, 54, 57, 62, 138 et 200.)

Charnetski, J. R. et Soland, R. M. (1978). Multiple-attribute decision making with partial information : The comparative hypervolume criterion. *Naval Research Logistics Quarterly*, 25(2):279–288. (Citée pages 47, 176 et 179.)

Chassein, A. et Goerigk, M. (2015). Alternative formulations for the ordered weighted averaging objective. *Information Processing Letters*, 115(6-8):604–608. (Citée page 41.)

Chateauneuf, A. (1991). On the use of capacities in modeling uncertainty aversion and risk aversion. *Journal of mathematical Economics*, 20(4):343–369. (Citée page 39.)

Chateauneuf, A. et Tallon, J.-M. (2002). Diversification, convex preferences and non-empty core in the choquet expected utility model. *Economic Theory*, 19(3):509–523. (Citée page 38.)

Chong, K.-M. (1976). An induction theorem for rearrangements. *Canadian Journal of Mathematics*, 28(1):154–160. (Citée page 22.)

Choquet, G. (1954). Theory of capacities. *In Annales de l'institut Fourier*, volume 5, pages 131–295. (Citée page 37.)

Chu, W. et Ghahramani, Z. (2005). Preference learning with gaussian processes. *In Proceedings of the 22nd international conference on Machine learning*, pages 137–144. (Citée page 138.)

Cortes, C. et Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297. (Citée page 47.)

Duchi, J., Hazan, E. et Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7). (Citée page 49.)

E Costa, C. A. B. et Vansnick, J.-C. (1995). General overview of the macbeth approach. *In Advances in multicriteria analysis*, pages 93–100. Springer. (Citée pages 44 et 48.)

Eric, B., Freitas, N. D. et Ghosh, A. (2008). Active preference learning with discrete choice data. *In Advances in neural information processing systems*, pages 409–416. (Citée page 138.)

Frean, M. (1992). A" thermal" perceptron learning rule. *Neural Computation*, 4(6):946–957. (Citée page 49.)

French, S. (1986). *Decision theory : an introduction to the mathematics of rationality*. Halsted Press. (Citée page 54.)

Fukuda, K. et Prodon, A. (1996). Double description method revisited. *In Comb. and Computer Science*, pages 91–111. Springer. (Citée page 127.)

Galand, L. (2008). *Méthodes exactes pour l'optimisation multicritère dans les graphes : recherche de solutions de compromis*. Thèse de doctorat, Paris 6. (Citée page 84.)

Galand, L. et Perny, P. (2006). Search for compromise solutions in multiobjective state space graphs. *In Proceedings of the Seventeenth European Conference on Artificial Intelligence ECAI 2006)*, pages 93–97. (Citée pages 41, 42 et 85.)

Galand, L. et Perny, P. (2007). Search for choquet-optimal paths under uncertainty. *In UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, July 19-22, 2007*, pages 125–132. (Citée pages 41 et 42.)

Galand, L., Perny, P. et Spanjaard, O. (2010). A branch and bound algorithm for choquet optimization in multicriteria problems. *In Multiple criteria decision making for sustainable energy and transportation systems*, pages 355–365. Springer. (Citée pages 38, 41 et 42.)

Galand, L. et Spanjaard, O. (2012). Exact algorithms for owa-optimization in multiobjective spanning tree problems. *Computers & Operations Research*, 39(7):1540–1554. (Citée pages 12, 84 et 107.)

Gelain, M., Pini, M. S., Rossi, F., Venable, K. B. et Walsh, T. (2010). Elicitation strategies for soft constraint problems with missing preferences : Properties, algorithms and experimental studies. *Artificial Intelligence*, 174(3-4):270–294. (Citée pages 13 et 55.)

Geoffrion, A. M., Dyer, J. S. et Feinberg, A. (1972). An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Management science*, 19(4-part-1):357–368. (Citée page 51.)

Gilbert, H., Spanjaard, O., Viappiani, P. et Weng, P. (2015). Reducing the number of queries in interactive value iteration. *In Proceedings of ADT-15*, pages 139–152. (Citée page 199.)

Grabisch, M. *et al.* (1996). The application of fuzzy integrals in multicriteria decision making. *European journal of operational research*, 89(3):445–456. (Citée page 37.)

Grabisch, M. et Labreuche, C. (2010). A decade of application of the choquet and sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 175(1): 247–286. (Citée pages 12 et 38.)

Grabisch, M., Marichal, J., Mesiar, R. et Pap, E. (2009). *Aggregation Functions*. Camb. U. Press. (Citée page 150.)

Greco, S., Mousseau, V. et Słowiński, R. (2009). The possible and the necessary for multiple criteria group decision. *In International Conference on Algorithmic Decision-Theory*, pages 203–214. Springer. (Citée pages 12 et 43.)

Guillot, P.-L. et Destercke, S. (2019). Preference elicitation with uncertainty : Extending regret based methods with belief functions. *In International Conference on Scalable Uncertainty Management*, pages 289–309. Springer. (Citée page 58.)

Guo, S. et Sanner, S. (2010). Multiattribute bayesian preference elicitation with pairwise comparison queries. *In International Symposium on Neural Networks*, pages 396–403. Springer. (Citée pages 13 et 58.)

Ha, V. et Haddawy, P. (1997). Problem-focused incremental elicitation of multi-attribute utility models. *In Proceedings of UAI-97*, pages 215–222. Morgan Kaufmann Publishers Inc. (Citée page 62.)

Heskes, T. (2000). The use of being stubborn and introspective. *In Prerational Intelligence : Adaptive Behavior and Intelligent Systems Without Symbols and Logic, Volume 1, Volume 2 Prerational Intelligence : Interdisciplinary Perspectives on the Behavior of Natural and Artificial Systems, Volume 3*, pages 1184–1200. Springer. (Citée page 49.)

Hong, C. S., Karni, E. et Safra, Z. (1987). Risk aversion in the theory of expected utility with rank dependent probabilities. *Journal of Economic theory*, 42(2):370–381. (Citée page 36.)

Houlsby, N., Huszar, F., Ghahramani, Z. et Hernández-Lobato, J. M. (2012). Collaborative gaussian processes for preference learning. *In Advances in neural information processing systems*, pages 2096–2104. (Citée page 59.)

Jacquet-Lagreze, E. et Siskos, J. (1982). Assessing a set of additive utility functions for multicriteria decision-making, the uta method. *European journal of operational research*, 10(2):151–164. (Citée pages 43 et 48.)

Jiménez, V. M. et Marzal, A. (2003). A lazy version of Eppstein's k shortest paths algorithm. *In International Workshop on Experimental and Efficient Algorithms*, pages 179–191. Springer. (Citée pages 88, 98 et 100.)

Kaddani, S., Vanderpooten, D., Vanpeperstraete, J.-M. et Aissi, H. (2017). Weighted sum model with partial preference information : Application to multi-objective optimization. *European Journal of Operational Research*, 260(2):665–679. (Citée page 62.)

Konczak, K. et Lang, J. (2005). Voting procedures with incomplete preferences. *In Proc. IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, volume 20. Citeseer. (Citée page 12.)

Kouvelis, P. et Yu, G. (2013). *Robust discrete optimization and its applications*, volume 14. Springer Science & Business Media. (Citée pages 93, 97 et 112.)

Lahdelma, R., Hokkanen, J. et Salminen, P. (1998). SMAA - Stochastic Multiobjective Acceptability Analysis. *European J. of Operational Research*, 106(1):137 – 143. (Citée pages 47 et 176.)

Leclercq, F. (2018). Bayesian optimization for likelihood-free cosmological inference. *Physical Review D*, 98(6):063511. (Citée page 59.)

Lesca, J. et Perny, P. (2010). Lp solvable models for multiagent fair allocation problems. *In ECAI*, volume 2010, pages 393–398. (Citée pages 12 et 41.)

Li, D. (1996). Convexification of a noninferior frontier. *Journal of Optimization Theory and Applications*, 88(1):177–196. (Citée page 188.)

Lokman, B., Köksalan, M., Korhonen, P. J. et Wallenius, J. (2018). An interactive approximation algorithm for multi-objective integer programs. *Computers & Operations Research*, 96:80–90. (Citée page 51.)

Marichal, J.-L. et Roubens, M. (2000). Determination of weights of interacting criteria from a reference set. *European journal of operational Research*, 124(3):641–650. (Citée page 48.)

Marshall, A. W., Olkin, I. et Arnold, B. C. (1979). *Inequalities : theory of majorization and its applications*, volume 143. Springer. (Citée page 22.)

Martin, H. et Perny, P. (2020). New computational models for the choquet integral. *In 24th European Conference on Artificial Intelligence-ECAI 2020*. (Citée pages 12 et 41.)

Miranda, P., Combarro, E. F. et Gil, P. (2006). Extreme points of some families of non-additive measures. *European journal of operational research*, 174(3):1865–1884. (Citée page 172.)

Moulin, H. (1988). Axioms of cooperative decision making cambridge univ. (Citée page 21.)

Murty, K. G. (1968). An algorithm for ranking all the assignments in order of increasing costs. *Operations Research*, 16(3):682–687. (Citée pages 88 et 93.)

Nowak, R. (2009). Noisy generalized binary search. *In* Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I. et Culotta, A., éditeurs : *Advances in Neural Information Processing Systems 22*, pages 1366–1374. Curran Associates, Inc. (Citée pages 182 et 200.)

Ogryczak, W. et Śliwiński, T. (2003). On solving linear programs with the ordered weighted averaging objective. *European Journal of Operational Research*, 148(1):80–91. (Citée pages 12, 41, 120, 131, 157 et 173.)

Ogryczak, W. et Śliwiński, T. (2007). On optimization of the importance weighted owa aggregation of multiple criteria. *In International Conference on Computational Science and Its Applications*, pages 804–817. Springer. (Citée pages 12 et 41.)

Perny, P. (2000). Modélisation des préférences, agrégation multicritère et systèmes d'aide à la décision. *Habilitation à diriger les recherches*. (Citée pages 27 et 28.)

Perny, P., Viappiani, P. et Boukhatem, A. (2016). Incremental preference elicitation for decision making under risk with the rank-dependent utility model. *In Proceedings of UAI 2016*. (Citée pages 13, 62, 70 et 72.)

Quiggin, J. (1993). *Generalized Expected Utility Theory : The Rank Dependent Model*. Springer Science & Business Media. (Citée page 35.)

Ramsay, J. O. *et al.* (1988). Monotone regression splines in action. *Statistical science*, 3(4):425–441. (Citée page 70.)

Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. (Citée page 46.)

Roy, B. (1976). From optimisation to multicriteria decision aid : three main operational attitudes. *In Multiple criteria decision making*, pages 1–34. Springer. (Citée page 50.)

Roy, B. (1985). *Méthodologie multicritère d'aide à la décision*. Numéro BOOK. Economica. (Citée page 18.)

Salo, A. A. et Hamalainen, R. P. (2001). Preference ratios in multiattribute evaluation (prime)-elicitation and decision procedures under incomplete information. *IEEE Transactions on Systems, Man, and Cybernetics-Part A : Systems and Humans*, 31(6):533–545. (Citée pages 13 et 54.)

Sauré, D. (2016). Ellipsoidal methods for adaptive choice-based conjoint analysis. (Citée page 58.)

Sauré, D. et Vielma, J. P. (2019). Ellipsoidal methods for adaptive choice-based conjoint analysis. *Operations Research*, 67(2):315–338. (Citée pages 13, 58 et 138.)

Savage, L. J. (1954). The foundations of statistics. (Citée pages 44, 53, 79 et 200.)

Schmeidler, D. (1986). Integral representation without additivity. *Proceedings of the American mathematical society*, 97(2):255–261. (Citée page 37.)

Shivaswamy, P. et Joachims, T. (2015). Coactive learning. *Journal of Artificial Intelligence Research*, 53:1–40. (Citée page 49.)

Shorrocks, A. F. (1983). Ranking income distributions. *Economica*, 50(197):3–17. (Citée pages 21 et 22.)

Slowinski, R., Greco, S. et Mousseau, V. (2005). Multi-criteria ranking of a finite set of alternatives using ordinal regression and additive utility functions-a new utagms method. *In Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum fr Informatik. (Citée page 43.)

Stephen, I. (1990). Perceptron-based learning algorithms. *IEEE Transactions on neural networks*, 50(2):179. (Citée page 46.)

Steuer, R. E. et Choo, E.-U. (1983). An interactive weighted tchebycheff procedure for multiple objective programming. *Mathematical programming*, 26(3):326–344. (Citée page 50.)

Tanner, M. A. et Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *J. Am. Stat. Asso.*, 82(398):528–540. (Citée pages 138 et 144.)

Tehrani, A. F., Cheng, W., Dembczyński, K. et Hüllermeier, E. (2012). Learning monotone nonlinear models using the choquet integral. *Machine Learning*, 89(1-2):183–211. (Citée page 48.)

Teso, S., Passerini, A. et Viappiani, P. (2016). Constructive preference elicitation by setwise max-margin learning. *In Proceedings of IJCAI-16*, pages 2067–2073. (Citée pages 55, 81 et 138.)

Torra, V. (1997). The weighted owa operator. *International Journal of Intelligent Systems*, 12:153–166. (Citée pages 12 et 35.)

Vanderpooten, D. et Vincke, P. (1989). Description and analysis of some representative interactive multicriteria procedures. *In Models and Methods in Multiple Criteria Decision Making*, pages 1221–1238. Elsevier. (Citée page 51.)

Vendrov, I., Lu, T., Huang, Q. et Boutilier, C. (2020). Gradient-based optimization for bayesian preference elicitation. *In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020,*, pages 10292–10301. AAAI Press. (Citée pages 13, 58 et 182.)

Vincke, P. (1976). Une méthode interactive en programmation linéaire à plusieurs fonctions économiques. *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle*, 10(V2):5–20. (Citée page 51.)

Wakker, P. (1990). Under stochastic dominance choquet-expected utility and anticipated utility are identical. *Theory and Decision*, 29(2):119–132. (Citée page 39.)

Wang, T. et Boutilier, C. (2003). Incremental utility elicitation with the minimax regret decision criterion. *In Ijcai*, volume 3, pages 309–316. (Citée pages 13, 53, 54, 62, 78, 141, 199 et 200.)

Weng, P. et Zanuttini, B. (2013). Interactive value iteration for markov decision processes with unknown rewards. *In Proceedings of IJCAI-13*, pages 2415–2421. (Citée pages 55 et 62.)

White III, C. C., Sage, A. P. et Dozono, S. (1984). A model of multiattribute decision making and trade-off weight determination under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics*, 14(2):223–229. (Citée pages 13, 53 et 62.)

Yaari, M. E. (1987). The dual theory of choice under risk. *Econometrica*, 55:95–115. (Citée pages 34 et 36.)

Yager, R. R. (1998). On ordered weighted averaging aggregation operators in multicriteria decision making. *In IEEE Trans. Systems, Man and Cybern.*, volume 18, pages 183–190. (Citée pages 12 et 32.)

Zheng, J., Shen, F., Fan, H. et Zhao, J. (2013). An online incremental learning support vector machine for large-scale data. *Neural Computing and Applications*, 22(5):1023–1035. (Citée page 49.)

Zhou-Kangas, Y. et Miettinen, K. (2019). Decision making in multiobjective optimization problems under uncertainty : balancing between robustness and quality. *OR Spectrum*, 41(2):391–413. (Citée page 51.)

Zintgraf, L. M., Roijers, D. M., Linders, S., Jonker, C. M. et Nowé, A. (2018). Ordered preference elicitation strategies for supporting multi-objective decision making. *In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1477–1485. International Foundation for Autonomous Agents and Multiagent Systems. (Citée pages 59 et 138.)

Résumé

Les travaux effectués durant cette thèse s'inscrivent dans le cadre de la théorie de la décision algorithmique, domaine au carrefour de la théorie de la décision, de la recherche opérationnelle et de l'intelligence artificielle. Cette thèse vise à concevoir des méthodes d'optimisation interactive fondées sur l'élicitation incrémentale des préférences pour la prise de décision multicritère, multi-agents ou dans le risque. Nous nous intéressons plus précisément à l'élicitation incrémentale des paramètres de fonctions d'agrégation qui consiste à alterner questions préférentielles permettant de réduire l'incertitude concernant la valeur des paramètres modélisant les préférences particulières du décideur, et exploration de l'espace des solutions, jusqu'à pouvoir déterminer une recommandation de bonne qualité. L'intérêt d'alterner phases de questions et phases d'exploration est double : d'une part, les informations préférentielles récoltées durant une phase d'élicitation permettent de mieux focaliser la phase d'exploration suivante sur les solutions les plus intéressantes pour le décideur ; d'autre part, l'exploration de l'espace des solutions permet de guider le choix des questions de manière à ce qu'elles soient les plus informatives possible. Nous introduisons dans cette thèse des méthodes d'élicitation dans différents contextes. Dans un premier temps, nous nous intéressons à des fonctions d'agrégation non-linéaires pour modéliser les préférences du décideur sur un ensemble combinatoire d'alternatives. Nous nous intéressons ensuite à la conception de méthodes d'élicitation prenant en compte la possibilité de la présence d'incohérences dans les réponses du décideur, d'abord sur domaine explicite, puis sur domaine combinatoire. Les algorithmes introduits sont génériques et peuvent s'appliquer à différents problèmes de choix multi-objectifs.

Mots clés. Théorie de la décision algorithmique, élicitation incrémentale des préférences, optimisation combinatoire multi-objectifs, régression Bayésienne.

Abstract

This thesis work falls within the area of algorithmic decision theory, a research domain at the crossroad of decision theory, operations research and artificial intelligence. The aim is to produce interactive optimization methods based on incremental preference elicitation in decision problems involving several criteria, opinions of agents or scenarios. Preferences are represented by general decision models whose parameters must be adapted to each decision problem and each decision maker. Our methods interleave the elicitation of parameters and the exploration of the solution space in order to determine the optimal choice for the decision maker. The idea behind this is to use information provided by the elicitation to guide the exploration of the solution space and vice versa. In this thesis, we introduce new incremental elicitation methods for decision making in different contexts: first for decision making in combinatorial domains when the decision models are non-linear, and then in a setting where one takes into account the possibility of inconsistencies in the answers of te decision maker. All the algorithms that we introduce are general and can be applied to a wide range of multiobjective decision problems.

Keywords. Algorithmic decision theory, incremental preference elicitation, multiobjective combinatorial optimization, Bayesian regression.