THÈSE

présentée pour obtenir le titre de

DOCTEUR DE l'ÉCOLE NATIONALE SUPÉRIEURE DE L'AÉRONAUTIQUE ET DE L'ESPACE

Spécialité : Informatique

par

Cédric Pralet

Un cadre algébrique général pour représenter et résoudre des problèmes de décision séquentielle avec incertitudes, faisabilités et utilités

---

Thèse présentée devant le jury composé de:

Malik Ghallab	**LAAS-CNRS, Toulouse**	Examinateur
Patrice Perny	**LIP6, Paris**	Rapporteur
Francesca Rossi	**Université de Padoue (Italie)**	Rapporteur
Thomas Schiex	**INRA, Toulouse**	Directeur de thèse
Gérard Verfaillie	**ONERA, Toulouse**	Directeur de thèse
Nic Wilson	**4C, Cork (Irlande)**	Examinateur

Thèse préparée au LAAS-CNRS et à l'INRA Toulouse

Table des matières

Note de lecture

Ce document est un résumé d'une thèse écrite en anglais. Pour une vision complète et formelle du travail réalisé, nous conseillons au lecteur de se référer à la version anglaise.

Remerciements

Merci tout d'abord à Elyssa de m'avoir toujours supporté (dans les deux sens du terme) pendant ma thèse. Cette thèse est un peu la tienne. Merci aussi à ma famille pour son soutien. Je tiens également à remercier les personnes suivantes, tant sur le plan scientifique que sur le plan humain :

— Thomas Schiex et Gérard Verfaillie, mes deux directeurs de thèse, pour leur disponibilité, l'excellence de leur encadrement, leur ouverture d'esprit, et leur soutien. Merci notamment pour le caractère scientifiquement stimulant de nos réunions, qui, de mon point de vue, ont fait du travail de recherche un pur plaisir.

— Francesca Rossi, de l'université de Padoue, et Patrice Perny, de l'université Paris 6, qui m'ont fait l'honneur de s'intéresser à mon travail en acceptant d'être rapporteurs de cette thèse.

— Malik Ghallab, directeur du LAAS-CNRS, et Nic Wilson, chercheur au Cork Constraint Computation Center, pour avoir accepté de participer à mon jury de thèse. Merci sincèrement à Nic de m'avoir invité à présenter mes travaux à un workshop ECAI'06. Je lui suis réellement reconnaissant de cette belle opportunité.

— Aux membres de mon "comité de thèse" réunis à l'issue de mes premières et deuxièmes années de thèse : Rachid Alami du LAAS-CNRS, Jean-Loup Farges de l'ONERA Toulouse, Jérôme Lang de l'IRIT, et Régis Sabbadin de l'INRA Toulouse. Merci pour leur lecture attentive de mes rapports d'avancement et pour les discussions que j'ai pu avoir avec eux par la suite.

— Plus généralement, merci aux personnes du groupe RIA du LAAS-CNRS et aux personnes de l'INRA pour la bonne ambiance de travail dont j'ai pu bénéficier.

Introduction

Au cours des dernières décennies, de nombreux formalismes ont été développés pour représenter et résoudre des problèmes de décision pouvant correspondre à des problèmes de planification d'actions (comme en ordonnancement de tâches ou en allocation de ressources) ou à des problèmes de recherche d'explications (comme en diagnostic ou en suivi de situation). Ces problèmes peuvent être plus ou moins complexes suivant les paramètres qu'ils intègrent :

1. L'évolution de l'environnement peut être déterministe ou non et on peut avoir des mesures d'incertitudes, appelées *plausibilités*, concernant l'état du monde.
2. Certaines actions peuvent être *faisables* uniquement si certaines préconditions sont satisfaites.
3. Les différents états de l'environnement et les diverses décisions possibles n'ont généralement pas la même valeur du point de vue des décideurs : des préférences peuvent être exprimées pour modéliser des coûts, des gains, des risques, des degrés de satisfaction, des exigences dures... Ces préférences sont appelées ici des *utilités*.
4. Le processus décisionnel peut être *séquentiel*, c'est-à-dire qu'il peut y avoir plusieurs étapes de décision. Certaines informations peuvent être observées entre deux étapes de décision, à l'instar des échecs où deux joueurs jouent à tour de rôle, et où chaque coup est joué après observation du dernier coup adverse.
5. Le problème peut faire intervenir plusieurs agents collaboratifs ou antagonistes. Certaines décisions peuvent être non *contrôlables* par un agent donné.

Cette thèse considère des formes générales de problèmes de décision faisant intervenir tous ces aspects. Plus précisément, étant données les plausibilités sur l'état de l'environnement, les contraintes de faisabilité sur les décisions, les utilités définissant des préférences et la succession des différentes étapes de décision, le but est de fournir à un agent décideur des règles de décision optimales pour les décisions qu'il contrôle, ceci en fonction de l'environnement et des autres agents.

De nombreux formalismes classiques existent pour résoudre des problèmes inclus dans cette classe de problèmes. Parmi ces formalismes, on peut citer :

— le cadre des problèmes de satisfiabilité d'une formule logique propositionnelle (SAT) et ses extensions permettant de prendre en compte un contexte non déterministe (*Quantified Boolean Formulae, QBF*) ou stochastique (Stochastic Satisfiability [62]) ;

— le cadre très proche des problèmes de satisfaction de contraintes (*Constraint Satisfaction Problems*, CSP [63]) et ses extensions permettant de prendre en compte des préférences (*Valued* ou *Semiring-based CSP* [11]) ou un contexte non déterministe (*Quantified CSP* [13], *Mixed CSP* [38]) ou stochastique (*Stochastic CSP* [107]) ;

— le cadre de la représentation de l'incertain, incluant les *réseaux bayésiens* [73], les champs de Markov (*Markov random fields* [18]), les graphes chaînés [42], ainsi que leurs extensions permettant de prendre en compte des contraintes (*Hybrid* et *Mixed networks* [29, 30]), ou alors des décisions, des utilités et/ou des faisabilités (*Influence diagrams* [48, 52, 101, 71, 51], *valuation networks* [98, 100, 34]) ;
— le cadre de la planification (*STRIPS* planning [40, 44], PDDL [65]) et ses extensions permettant de prendre en compte l'incertitude sur l'état courant et sur les effets des actions (*Conformant Planning* [46], *Probabilistic Planning* [58]) ;
— le cadre enfin des processus décisionnels markoviens (*Markov Decision Processes* [86]), avec ses extensions permettant de prendre en compte un contexte d'observabilité partielle (*Partially Observable MDP* [68]), des incertitudes non probabilistes [91, 74], ou encore la structure des états (*Factored MDP* [16]).

Au-delà de leurs nombreuses différences, ces cadres possèdent d'importantes similitudes :

— ils utilisent des variables à domaine souvent fini pour représenter soit l'état de l'environnement (*variables d'environnement*), soit les décisions d'un ou plusieurs agents (*variables de décision*) ;
— ils mettent en jeu des *fonctions locales* qui peuvent représenter des *faisabilités* pesant sur les variables de décision (par exemple, des pré-conditions d'actions), des *plausibilités* portant sur les variables d'environnement (par exemple, des distributions de probabilité conditionnelles) ou encore des *utilités*, fonctions de diverses variables (par exemple, des coûts ou des degrés de satisfaction) ;
— ils font appel à divers opérateurs, soit pour *agréger* les fonctions locales (par exemple, le $\wedge$ logique pour agréger les faisabilités, le $\times$ pour agréger les probabilités, le $+$ pour agréger les utilités additives), soit pour *synthétiser* une information globale (par exemple, le $\vee$ logique pour décider d'une faisabilité, le $+$ pour calculer une distribution de probabilité marginale, le max ou le min pour sélectionner une décision optimale).

Ils peuvent donc tous être vus comme des *modèles graphiques* dans la mesure où ils reposent tous, implicitement ou explicitement, sur un *hyper-graphe* de fonctions locales entre variables à domaine fini. Les différences entre eux tiennent essentiellement à ce que représentent variables et fonctions, ainsi qu'aux opérateurs d'agrégation et de synthèse utilisés.

Cette thèse montre qu'il est possible de les rassembler dans un même cadre générique, graphique et algébrique : *graphique* pour respecter leur nature graphique et *algébrique* pour abstraire les opérateurs d'agrégation et de synthèse utilisés et ne plus considérer que des opérateurs abstraits dotés de certaines propriétés algébriques. La variété des cadres visés, des problèmes de satisfiabilité aux processus décisionnels markoviens, peut cependant faire penser qu'une telle unification est hors d'atteinte ou que le résultat en serait un monstre incompréhensible et ingérable. Cette thèse montre qu'il n'en est rien et que tous ces cadres sont de fait suffisamment proches pour être, grâce à une approche algébrique, réunis dans un seul, dénommé PFU, et dont les composants et les propriétés peuvent être décrits de manière relativement simple et compacte.

Cette thèse montre également qu'il est possible de ramener de nombreux problèmes de décision à des calculs de séquences d'*éliminations* de variables sur une *combinaison* de fonctions locales.

Motivations Construire un cadre générique pour représenter et résoudre des problèmes de décision variés est utile pour diverses raisons :

— *Unification et meilleure compréhension des formalismes existants* : construire un cadre générique présente tout d'abord un intérêt théorique et pédagogique. Cette démarche peut permettre de mieux comprendre des relations souvent ignorées entre des cadres spécifiques développés par des communautés qui souvent se méconnaissent ;

— *Expressivité accrue* : un cadre générique peut permettre, par la variété de la structure proposée, de considérer des nouveaux cadres spécifiques non encore explorés ;

— *Intérêt algorithmique* : il devrait être possible de définir des algorithmes de résolution génériques, dont on sait qu'ils se révèlent souvent aussi performants, sinon plus, que les algorithmes spécifiques développés dans tel ou tel cadre au prix d'efforts non négligeables. Cet objectif est dans son esprit relié à une démarche globale d'identification d'approches algorithmiques communes développées pour résoudre différents problèmes d'intelligence artificielle. Il peut également permettre à un cadre donné de bénéficier des avancées algorithmiques réalisées dans d'autres cadres.

Organisation de la thèse Cette thèse est découpée en deux parties :

1. La première partie se concentre sur des aspects représentation de la connaissance. Elle introduit un nouveau cadre général de représentation pour la décision séquentielle avec incertitudes, faisabilités et utilités.

   Après avoir défini certaines notations et certaines notions (chapitre 1), nous commençons par montrer informellement au chapitre 2, via un catalogue de formalismes existants, pourquoi et comment un cadre générique peut être construit.

   Ce cadre générique, appelé le cadre Plausibilité-Faisabilité-Utilité (PFU), est ensuite formellement introduit en trois temps :

   — Des structures algébriques permettant d'exprimer des formes générales d'incertitudes, de faisabilités et d'utilités sont tout d'abord définies au chapitre 3. Ces structures spécifient comment combiner et synthétiser des informations.

   — Sur ces structures algébriques, nous introduisons au chapitre 4 une forme de modèle graphique faisant intervenir des variables et un réseau de fonctions locales entre ces variables.

   — Enfin, la notion de requête est introduite au chapitre 5. Les requêtes permettent de formuler des problèmes de décision variés sur un réseau de fonctions locales.

2. La second volet de cette thèse a pour objet la définition d'algorithmes génériques permettant de répondre à des requêtes définies dans le cadre PFU.

   — Les premiers algorithmes génériques présentés au chapitre 6 sont des algorithmes de recherche arborescente et d'élimination de variables qui essaient d'exploiter au mieux le fait que les informations sont exprimées par des fonctions *locales*. Leur complexité est fonction d'un paramètre appelé largeur induite contrainte.

   — Des techniques plus sophistiquées analysant la structure d'une requête de manière plus fine sont ensuite introduites au chapitre 7. Cette analyse structurelle nous conduit à une architecture de calcul générale, appelée l'architecture des DAG de clusters multi-opérateurs (DAG = Directed Acyclic Graph). Cette architecture exprime de manière

explicite une décomposition des calculs à réaliser pour répondre à une requête.

— Partant de cette architecture, le chapitre 8 définit des algorithmes de recherche arborescente structurée qui peuvent être plus ou moins sophistiqués suivant s'ils utilisent des techniques de mémorisation ou des bornes pour élaguer l'espace de recherche.

— Enfin, le chapitre 9 présente très brièvement un outil de résolution générique permettant de répondre à des requêtes sur un réseau PFU. Cet outil prouve notamment que le cadre développé n'est pas juste une abstraction.

Première partie

Un nouveau cadre générique de représentation de problèmes de décision : le cadre PFU

Chapitre 1

Notations et définitions

Ce court chapitre introduit quelques objets mathématiques utilisés intensivement par la suite. Nous manipulons notamment les notions de variables, domaines, fonctions locales, modèles graphiques, opérateurs de combinaison, opérateurs d'élimination, règles de décision et certains éléments de vocabulaire relatifs aux graphes. Certaines de ces notions sont illustrées par un exemple jouet qui précise également ce que nous entendons par "plausibilité", "faisabilité", "utilité", "observabilité partielle" ou "controlabilité".

1.1 Quelques définitions

Définition 1.1. *Le* domaine *de valeurs d'une variable x est noté $dom(x)$ et pour tout $a \in dom(x)$, (x, a) représente l'affectation de x avec la valeur a.*

Par extension, étant donné un ensemble de variables S, nous notons $dom(S)$ le produit cartésien des domaines des variables de S, c'est-à-dire $dom(S) = \prod_{x \in S} dom(x)$. Un élément $A \in dom(S)$ est appelé une affectation de S. [1]

Si A_1, A_2 sont deux affectations d'ensembles disjoints S_1, S_2, alors la concaténation de A_1 et A_2, notée $A_1.A_2$, est l'affectation de $S_1 \cup S_2$ dans laquelle les variables de S_1 prennent la même valeur que dans A_1 et les variables de S_2 prennent la même valeur que dans A_2.

Si A est une affectation d'un ensemble de variables S, alors la projection de A sur un ensemble de variables S', notée $A^{\downarrow S'}$, est l'affectation de $S \cap S'$ donnant à chaque variable la même valeur que dans A.

Définition 1.2. *(Fonction locale et portée d'une fonction locale) Une* fonction locale *est un couple (S, φ) tel que S est un ensemble de variables et φ est une fonction associant à chaque élément de $dom(S)$ un élément dans un ensemble E donné.*

Par la suite, nous considérons souvent que l'ensemble de variables S est implicite. Ainsi, une fonction locale (S, φ) peut être notée simplement φ. L'ensemble de variables S est appelée la portée *de φ et est noté $sc(\varphi)$ (sc comme "scope"). Si A est une affectation d'un sur-ensemble de $sc(\varphi)$, alors $\varphi(A)$ vaut $\varphi(A^{\downarrow sc(\varphi)})$.*

1. Techniquement, une affectation de $S = \{x_1, \ldots, x_k\}$ devrait être un ensemble de paires variable-valeur $\{(x_1, a_1), \ldots, (x_k, a_k)\}$. Nous supposons ici que les variables sont implicites lorsqu'un tuple de valeurs $(a_1, \ldots, a_k) \in dom(S)$ est utilisé.

Par exemple, une fonction locale φ associant à chaque affectation de $sc(\varphi)$ un élément dans le treillis booléen $\mathbb{B} = \{t, f\}$ est analogue à une contrainte décrivant le sous-ensemble des affectations de $sc(\varphi)$ qui satisfont la contrainte en question.

A partir de la notion de fonction locale, la notion de modèle graphique peut être définie :

Définition 1.3. *(Modèle graphique) Un* modèle graphique *est un couple* (V, Φ) *tel que* $V = \{x_1, \ldots, x_n\}$ *est un ensemble fini de variables et* $\Phi = \{\varphi_1, \ldots, \varphi_m\}$ *est un ensemble fini de fonctions locales dont la portée est incluse dans* V*.*

Le terme modèle *graphique* est utilisé simplement car un ensemble de fonctions locales peut être représenté par un hypergraphe dont les hyper-arêtes sont les portées des fonctions locales. Comme nous le verrons, cet hypergraphe représente une certaine forme d'indépendance conditionnelle et induit des paramètres influençant la complexité algorithmique. La définition adoptée ici généralise la définition classique utilisée en statistique selon laquelle un modèle graphique est un graphe (orienté ou non) dont les nœuds représentent des variables aléatoires et dont la structure modélise des relations d'indépendances conditionnelles probabilistes.

Les fonctions locales d'un modèle graphique expriment de manière compacte une fonction globale portant sur toutes les variables du modèle graphique. Cette fonction globale est obtenue en agrégeant toutes les fonctions locales. Par exemple, un réseau bayésien [73] représente une distribution de probabilité jointe globale $P_{x,y,z}$ sous la forme d'un produit de fonctions locales qui peuvent être les fonctions locales de l'ensemble $\{P_x, P_{y|x}, P_{z|x}\}$.

Afin de raisonner sur un modèle graphique (V, Φ), il est nécessaire de pouvoir synthétiser l'information qu'il exprime sur un sous-ensemble des variables de V. Par exemple, pour calculer une distribution de probabilité marginale $\mathcal{P}_{y,z}$ à partir du réseau bayésien précédent, nous devons calculer la quantité : $\sum_x P_{x,y,z} = \sum_x (P_x \times P_{y|x} \times P_{z|x})$. Partant d'une information portant sur $\{x, y, z\}$, l'opérateur $\sum$ permet d'obtenir une information sur $\{y, z\}$ en "éliminant" la variable x. Les opérateurs utilisés pour combiner des fonctions locales sont appelés des opérateurs de *combinaison* et les opérateurs utilisés pour synthétiser des informations sont appelés des opérateurs d'*élimination*.

Définition 1.4. *(Combinaison) Soit* φ_1*,* φ_2 *deux fonctions locales à valeurs dans* E_1 *et* E_2 *respectivement. Soit* $\otimes : E_1 \times E_2 \to E$ *un opérateur binaire. La combinaison de* φ_1 *et* φ_2*, notée* $\varphi_1 \otimes \varphi_2$*, est la fonction locale à valeurs dans* E *dont la portée est* $sc(\varphi_1) \cup sc(\varphi_2)$*, et qui satisfait, pour toute affectation* A *de cette portée,* $(\varphi_1 \otimes \varphi_2)(A) = \varphi_1(A) \otimes \varphi_2(A)$*. L'opérateur* $\otimes$ *est appelé un* opérateur de combinaison *de* φ_1 *et* φ_2*.*

Définition 1.5. *(Elimination) Soit* φ *une fonction locale à valeurs dans* E*. Soit op un opérateur associatif et commutatif sur* E*. L'élimination d'une variable* x *sur* φ *avec un opérateur op est la fonction locale de portée* $sc(\varphi) - \{x\}$ *qui satisfait, pour toute affectation* A *de cette portée,* $(op_x\, \varphi)(A) = op_{a \in dom(x)}\, \varphi(A.(x, a))$*. L'opérateur op est dans ce cas appelé* opérateur d'élimination *de la variable* x

De manière analogue, l'élimination d'un ensemble de variables $S = \{x_1, \ldots, x_k\}$ *sur* φ *est une fonction locale de portée* $sc(\varphi) - S$ *definie par* $(op_S\, \varphi)(A) = op_{A' \in dom(S)}\, \varphi(A.A')$*.*

Ainsi, dans l'expression $\sum_x (P_x \times P_{y|x} \times P_{z|x})$, des fonctions locales sont agrégées via l'opérateur de combinaison $\otimes = \times$ et l'information est synthétisée via une élimination de x avec l'opérateur

d'élimination $+$. Dans toute la suite de la partie I, $\otimes$ représente des opérateurs de combinaison et $\oplus$ représente des opérateurs d'élimination. Notons que la notion d'opérateur de combinaison ou d'élimination n'est pas une propriété intrinsèque d'un opérateur donné. Elle dépend de l'usage qui est fait de cet opérateur : par exemple, l'opérateur $+$ a le statut d'opérateur de combinaison s'il est utilisé pour agréger des gains et des coûts alors qu'il a le statut d'opérateur d'élimination s'il sert à calculer une distribution de probabilité marginale.

Dans certains cas, l'élimination d'un ensemble de variables S avec un opérateur op sur une fonction locale φ doit être réalisée uniquement sur un sous-ensemble de $dom(S)$ contenant les affectations qui satisfont une certaine propriété représentée par une fonction booléenne F. Nous devons alors calculer, pour chaque affectation $A \in dom(sc(\varphi)-S)$, la quantité $op_{A' \in dom(S), F(A')=t}\, \varphi(A.A')$. Pour des raisons de simplicité et d'homogénéité et pour n'utiliser que des éliminations sur $dom(S)$, on peut de manière équivalente tronquer la fonction φ pour que les éléments de $dom(S)$ violant la propriété définie par F soient associés à un élément spécial (noté $\Diamond$) qui est lui-même un élément neutre de op.

Définition 1.6. *(Opérateur de troncature) L'*élément infaisable $\Diamond$ *est un nouvel élément spécial et tout opérateur d'élimination op est étendu de manière à satisfaire* $op(\Diamond, e) = op(e, \Diamond) = e$ *pour tout élément e du domaine de définition de op.*

Soit $\{t, f\}$ *le treillis booléen. Pour chaque booléen b et chaque élément e, nous définissons l'opérateur* $\star$ *tel que* $b \star e$ *vaut e si* $b = t$ *et* $\Diamond$ *sinon.* $\star$ *est appelé l'opérateur de troncature.*

Etant donnée une fonction locale booléenne F, l'élément infaisable $\Diamond$ et l'opérateur de troncature $\star$ permettent d'écrire des quantités telles que $op_{A' \in dom(S), F(A')=t}\, \varphi$ sous la forme $op_S\, (F \star \varphi)$. Dans cette dernière forme, une élimination est réalisée sur tout le domaine des variables et la fonction locale F a le même statut que la fonction locale φ (c'est pourquoi nous disons que $\star$ et $\Diamond$ permettent d'utiliser des notations plus simples et plus homogènes).

Lorsqu'un problème de décision est résolu, l'objectif est souvent d'obtenir des *règles de décision* indiquant des décisions à prendre en fonction des informations disponibles :

Définition 1.7. *(Règle de décision, politique) Une* règle de décision *pour une variable x sachant un ensemble de variables* S' *est une fonction* $\delta : dom(S') \rightarrow dom(x)$ *associant à chaque affectation de* S' *une valeur du domaine de x. Par extension, une règle de décision pour un ensemble de variables S sachant un ensemble de variables* S' *est une fonction* $\delta : dom(S') \rightarrow dom(S)$*. Un ensemble de règles de décision est appelé une* politique.

Des exemples de règles de décision sont les règles décision qui sont optimales du point de vue d'un certain critère décision. Par exemple, si l'on travaille sur un ensemble totalement ordonné et si l'on doit effectuer le calcul $\sum_{S'} \max_S \varphi$ avec φ une fonction locale, une règle de décision optimale $\delta : dom(S') \rightarrow dom(S)$, qui vérifie $\varphi(A.\delta(A)) \succeq \varphi(A.A')$ pour tout $(A, A') \in dom(sc(\varphi)-S) \times dom(S)$, peut être obtenue en utilisant argmax.

Quelques définitions sur les graphes

Définition 1.8. $\mathcal{G} = (V, H)$ *est un hypergraphe si et seulement si* V *est un ensemble de variables et* H *est un ensemble d'hyper-arêtes sur* V*, i.e. un sous-ensemble de* 2^V*.*

Définition 1.9. *Un graphe $G = (V, E)$ est un arbre si et seulement si G est un graphe connexe, non orienté, et sans cycle. G est un arbre enraciné si et seulement si G est un graphe connexe, orienté et sans cycle. La racine de l'arbre est alors l'unique sommet du graphe sans parent.*

Définition 1.10. *(Graphe acyclique orienté (Directed Acyclic Graph, DAG)) Un graphe orienté G est un DAG si et seulement si G ne contient pas de cycle orienté. Lorsque des variables sont associées aux sommets du graphe, on note $pa_G(x)$ l'ensemble des parents d'une variable x dans G.*

Enfin, le cardinal d'un ensemble fini Γ est noté $|\Gamma|$.

1.2 Un exemple illustratif

Un exemple jouet a été bâti pour donner une vision concrète des notions de "plausibilités", "faisabilités", "utilités", "observabilité", "variable de décision", "variable d'environnement" ou encore de "contrôlabilité". Cet exemple illustre également concrêtement comment variables et fonctions locales peuvent exprimer une information globale de manière compacte. Il introduit enfin le lien entre problèmes de décision et séquences d'éliminations de variables sur des combinaisons de fonctions locales

Exemple *Jean a trois portes en face de lui : A, B, et C de gauche à droite. Derrière l'une de ces portes, se trouve un trésor et derrière une autre, un gangster. Jean doit décider quelle porte ouvrir. Il sait qu'il gagnera 10 000€ s'il ouvre la porte où se trouve le trésor, mais qu'il devra payer 4 000€ s'il ouvre la porte où se trouve le gangster.*

Modélisation Pour modéliser ce problème, nous introduisons trois variables : (1) deux variables représentant l'environnement de Jean, l'une notée *tr* pour représenter la porte du trésor et l'autre notée *ga* pour représenter la porte du gangster ; (2) une variable notée *do* (comme "door") représentant la décision de Jean. Ces trois variables ont toutes le même domaine de valeur $\{A, B, C\}$. Les variables de décision correspondent aux variables dont la valeur est choisie directement par un agent, alors que les variables d'environnement correspondent aux variables dont la valeur n'est pas choisie directement par un agent.

Pour modéliser les coûts et les gains possibles, nous introduisons deux fonctions locales d'utilité : une première U_1 qui exprime que si Jean ouvre la porte du trésor, il gagne 10 000€ (contrainte souple $do = tr$ de poids 10 000, qui renvoie son poids si elle est satisfaite et 0 sinon) et une seconde U_2 qui exprime que si Jean ouvre la porte du gangster, il paye 4 000€ (contrainte souple $do = ga$ de poids -4 000, qui renvoie de même son poids si elle est satisfaite et 0 sinon). Une contrainte souple peut aussi être appelée une fonction de coût.

Requête Quelle est ou quelles sont la(les) décision(s) de Jean qui maximise(nt) son utilité si le gangster est derrière la porte A et le trésor est derrière la porte C ? La réponse est évidemment la décision (do, C).

Ajout d'incertitudes

Dans les problèmes réels, l'environnement peut ne pas être complètement connu : il peut exister des incertitudes sur l'environnement, appelées ici des plausibilités, et certaines observations de l'environnement incertain peuvent éventuellement être réalisées.

Example *Le trésor et le gangster ne sont pas derrière la même porte et toutes les situations possibles sont équiprobables. Jean travaille en équipe avec Pierre et chacun peut choisir une porte où écouter et ainsi tenter de repérer le gangster. On suppose que la probabilité d'entendre le gangster est de* 0.8 *si on écoute à la porte derrière laquelle il se trouve, de* 0.4 *si on écoute à une porte adjacente et de* 0 *sinon.*

Nouvelle modélisation Pour modéliser ces nouveaux éléments, nous introduisons quatre variables supplémentaires :

— deux variables de décisions li_J et li_P (li comme "listen") de domaine $\{A, B, C\}$ qui représentent respectivement les portes auxquelles Jean et Pierre écoutent ;

— deux variables d'environnement he_J et he_P (he comme "hear") de domaine $\{yes, no\}$ qui représentent respectivement le fait que Jean ou Pierre entend le gangster ou non.

Nous introduisons également des *fonctions locales de plausibilité* :

— $P_1 : ga \neq tr$ et $P_2 = 1/6$, qui représentent la distribution de probabilité sur les positions du gangster et du trésor

— $P_3 = P_{he_J \mid li_J, ga}$, qui spécifie la probabilité que Jean entende du bruit sachant la porte à laquelle il écoute et la porte derrière laquelle se trouve le gangster ;

— P_4, qui correspond de manière analogue à la distribution conditionnelle $P_{he_P \mid li_P, ga}$.

Ces fonctions locales de plausibilité satisfont implicitement certaines *conditions de normalisation*. D'une part, étant donné que le gangster et le trésor se trouvent quelque part, on peut écrire $\sum_{ga,tr} (P_1 \times P_2) = 1$. D'autre part, étant donné que Jean et Pierre entendent quelque chose ou non, nous avons $\sum_{he_J} P_3 = 1$ et $\sum_{he_P} P_4 = 1$. Ces normalisations traduisent le fait que la disjonction de toutes les situations possibles est certaine.

Requêtes associées Quelle est ou quelle sont la(les) décision(s) qui maximisent l'utilité espérée, si on suppose que Jean et Pierre choisissent chacun une porte où écouter et ensuite Jean choisit une porte à ouvrir en fonction de ce qui a été entendu ?

Pour répondre à une telle requête, une approche classique consiste à construire un *arbre de décision*. Dans cet arbre, les variables sont considérées dans un ordre cohérent avec l'ordre des décisions et des observations, par exemple dans l'ordre $li_J \rightarrow li_P \rightarrow he_J \rightarrow he_P \rightarrow do \rightarrow ga \rightarrow tr$. Tout nœud n de cet arbre correspond à une variable x et toute arête de n vers un nœud fils correspond à une affectation (x, a) de x. Si x est une variable d'environnement, cette arête est de plus pondérée par la probabilité $P((x, a) \mid A)$, où A est l'affectation associée au chemin de la racine à n. Toute feuille de cet arbre correspond à une affectation A de l'ensemble des variables et son utilité est l'utilité globale $(U_1 + U_2)(A)$ associée à A. L'utilité d'un nœud de décision est l'utilité optimale de ses nœuds fils, avec possibilité de mémoriser la(les) décision(s) correspondante(s). L'utilité d'un nœud d'environnement est la somme des utilités de ses nœuds fils, pondérées par le poids des arêtes associées. L'utilité espérée associée à la requête est celle du nœud racine. Il est

cependant prouvé [79] que cette approche à base d'arbres de décisions est équivalente au calcul de la formule suivante qui ne fait intervenir que les probabilités présentes dans la définition du problème, et pas des probabilités de type $P((x,a) \mid A)$ dont le calcul est potentiellement complexe :

$$\max_{li_J, li_P} \sum_{he_J, he_P} \max_{do} \sum_{ga,tr} ((\prod_{i\in[1,4]} P_i) \times (\sum_{i\in[1,2]} U_i))$$

Cet exemple montre que raisonner à partir d'arbres de décision est équivalent à effectuer une séquence d'éliminations de variables sur une combinaison de fonctions locales. Des règles de décision optimales peuvent être mémorisées en utilisant un argmax pendant les calculs.

D'autres scénarios correspondent à d'autres séquences d'éliminations :

— Si Jean pense que Pierre est un traître et s'il le laisse choisir une porte à ouvrir en premier (attitude pessimiste vis-à-vis de Pierre), la séquence d'éliminations devient

$$\min_{li_P} \max_{li_J} \sum_{he_J, he_P} \max_{do} \sum_{ga,tr}$$

Cette séquence élimine li_P avec l'opérateur min.

— Si Pierre ne dit même pas à Jean ce qu'il a entendu, ou autrement si Jean n'observe pas la valeur de la variable he_P, alors la séquence d'éliminations devient

$$\min_{li_P} \max_{li_J} \sum_{he_J} \max_{do} \sum_{he_P} \sum_{ga,tr}$$

c'est-à-dire que l'élimination $\sum_{he_P}$ est placée à droite de l'élimination $\max_{do}$.

Ajout de faisabilités

Il se peut que certaines préconditions soient requises pour que certaines décisions soient faisables. Par exemple, si deux joueurs acceptent de respecter les règles des échecs, alors un coup est dit faisable s'il respecte ces règles. Notons ici que ce qui est infaisable est différent de ce qui est inacceptable, puisque par exemple aucun des joueurs ne peut jouer un coup impossible alors que chacun d'entre eux peut éventuellement jouer un coup inacceptable pour son adversaire en le mettant échec et mat.

Exemple *Jean et Pierre ne peuvent pas écouter à la même porte, et la porte A est fermée.*

Modélisation Ces contraintes sur les décisions peuvent être modélisées en utilisant deux fonctions locales de faisabilité : une première F_1 qui exprime que Jean et Pierre ne peuvent pas écouter à la même porte (contrainte dure $li_J \neq li_P$) et une seconde F_2 qui exprime que la porte C ne peut pas être ouverte (contrainte dure $do \neq C$). De la même façon qu'avec les plausibilités, ces contraintes sont supposées exprimer des distributions de faisabilité normalisées ($\vee_{li_J, li_P} F_1 = t$ et $\vee_{do} F_2 = t$) pour traduire le fait qu'une décision est toujours possible, même s'il s'agit de ne rien faire.

Avec ces données supplémentaires, la procédure classique à base d'arbre de décision est équivalente au calcul de la quantité suivante :

$$\min_{li_P} \max_{li_J} \sum_{he_J} \max_{do} \sum_{he_P} \sum_{ga,tr} ((\bigwedge_{i\in[1,2]} F_i) \star (\prod_{i\in[1,4]} P_i) \times (\sum_{i\in[1,2]} U_i))$$

dans laquelle l'opérateur de troncature $\star$ permet de ne pas prendre en compte les décisions infaisables. La forme obtenu est à nouveau une séquence d'éliminations de variables sur une combinaison de fonctions locales.

Au terme de cet exemple, nous voyons que les données du problème peuvent être exprimées par l'intermédiaire de variables et de fonctions locales qui forment un modèle graphique *composite* constitué d'un DAG représentant des conditions de normalisation sur les plausibilités et les faisabilités (figure 1.1(a)), [2] et d'un réseau de fonctions locales (figure 1.1(b)). Ce réseau fait intervenir plusieurs types de variables (variables de décision et variables d'environnement) et plusieurs types de fonctions locales (fonctions locales de plausibilité, de faisabilité et d'utilité).

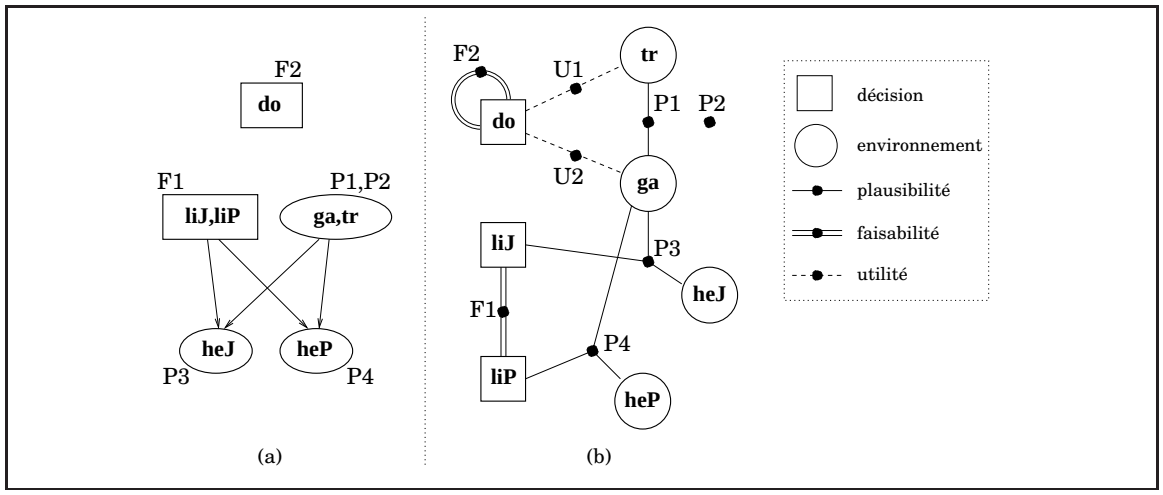

Figure 1.1: Modèle graphique composite (a) DAG représentant des conditions de normalisation ; (b) Réseau de fonctions locales.

En résumé, l'exemple introduit ici illustre la notion de problème de décision séquentielle faisant intervenir des plausibilités, des faisabilités et des utiités. Cette notion est fortement utilisée dans les chapitres à venir.

2. Si P est l'ensemble des fonctions locales de plausibilité associées à un nœud du DAG étiqueté par un ensemble de variables S, cela signifie que $\sum_S (\prod_{P_i \in P} P_i) = 1$. De même, si F est l'ensemble des fonctions locales de faisabilités associées à un nœud du DAG étiqueté par un ensemble de variables S, cela signifie que $\vee_S(\wedge_{F_i \in F} F_i) = t$.

Chapitre 2

Cadres existants

L'étape prélable à la définition d'un cadre générique pour la décision est une étape de compréhension et d'analyse des formalismes existants. Ces formalismes, issus d'efforts non négligeables réalisés au sein de plusieurs communautés, sont nombreux. Ils peuvent présenter des capacités de représentation plus ou moins sophistiquées. Certains peuvent modéliser des préférences alors que d'autres sont adaptés uniquement pour modéliser des exigences dures. Certains peuvent modéliser des incertitudes alors que d'autres ne le peuvent pas. Certains peuvent modéliser des problèmes de décision séquentielle, d'autres ne le peuvent pas.

Ce chapitre présente un catalogue **non-exhaustif** de certains de ces formalismes. Ce catalogue présente deux caractéristiques principales :

— Il est incrémental, dans le sens où il montre comment des cadres de base que sont le cadre des problèmes de satisfiabilité d'une formule logique propositionnelle (SAT), les problèmes de satisfaction de contraintes [63], les réseaux bayésiens [73], la planification classique [40, 44] ou les processus décisionnels markoviens [86, 68] ont été étendus pour intégrer la notion d'incertitude pour certains ou la notion de préférence et de décision pour d'autres.

— Il analyse les similitudes et les différences entre les formalismes existants en termes de représentation de la connaissance. Cette analyse tend à montrer que beaucoup de formalismes raisonnent à partir de variables et de fonctions locales entre variables et que les problèmes de décision qu'ils permettent de formuler peuvent être réduits à des calculs de séquences d'éliminations de variables sur une combinaison de fonctions locales.

Se référer à la version anglaise de la thèse pour une description plus précise et plus complète des formalismes existants.

2.1 Des CSP aux MDP algèbriques

Le cadre des problèmes de satisfaction de contraintes (*Constraint Satisfaction Problems*, CSP [63]) ou celui de la satisfiabilité d'une formule logique propositionnelle (SAT) peuvent être vus comme des modèles graphiques (V, Φ) dans lesquels l'ensemble des fonctions locales Φ contient des contraintes ou des clauses associant la valeur vrai ou faux à chaque affectation de leur portée. Une requête classique sur les CSP consiste à trouver une affectation des variables qui satisfait toutes les contraintes. En termes purement algébriques, cette requête peut être vue comme une requête d'optimisation

binaire s'écrivant sous la forme (en supposant $f \prec t$) :

$$\max_{V}(\bigwedge_{\varphi\in\Phi}\varphi) \tag{2.1}$$

Si la quantité précédente vaut t, alors le CSP est dit cohérent et une affectation optimale de V obtenue en utilisant argmax définit une solution, c'est-à-dire une affectation des variables qui satisfait toutes les contraintes. Si la quantité précédente vaut f, alors le CSP est dit incohérent et aucune affectation de V ne satisfait toutes les contraintes. Dans l'équation 2.1, les fonctions locales sont combinées par un $\wedge$ logique et les variables sont toutes éliminées par un max.

En remplaçant les contraintes dures par des contraintes souples pour lesquelles un coût est payé en cas de violation de la contrainte et en remplaçant $\wedge$ par un opérateur de combinaison $\otimes$ qui peut valoir min, max, $+$, $\times\ldots$ nous obtenons la forme algébrique associée à la recherche d'une solution pour un CSP valué [94] ou un *semiring-based CSP* [9, 10] totalement ordonné.

Dans une autre direction, un *réseau bayésien* [73] est aussi un modèle graphique (V, Φ), dans lequel les fonctions locales sont des distributions de probabilité conditionnelles : $\Phi = \{P_{x\,|\,pa(x)}\,,\ x \in V\}$ où $pa(x)$ correspond à l'ensemble des parents de la variable x dans un graphe acyclique orienté (DAG) associé au réseau bayésien. Un tel réseau représente de façon concise une distribution de probabilité jointe P_V sur toutes les variables, qui s'écrit sous la forme $P_V = \prod_{x\in V} P_{x\,|\,pa(x)}$, de la même manière qu'un CSP représente une contrainte globale sur toutes les variables comme une conjonction de contraintes locales. Les requêtes susceptibles d'être formulées sur un réseau bayésien sont multiples. On peut par exemple chercher la distribution de probabilité marginale sur une variable $y \in V$. Algébriquement parlant, le calcul associé à cette requête est

$$P_y = \sum_{V-\{y\}}(\prod_{x\in V} P(x\,|\,pa(x))) \tag{2.2}$$

Dans ce cas, les fonctions locales sont combinées par un produit et toutes les variables, hormis y, sont éliminées par une somme.

Utiliser des distributions de probabilité conditionnelles locales n'est pas l'unique moyen d'exprimer une distribution de probabilité globale sous une forme factorisée. En effet, dans des domaines tels que le traitement d'images ou les neurosciences, d'autres modèles sont utilisés, parmi lesquels on trouve notamment le formalisme des champs de Markov (*Markov Random Fields* [18]). Dans un champ de Markov, une distribution de probabilité jointe globale P_V se factorise sous une forme appelée distribution de Gibbs, c'est-à-dire sous la forme $P_V = 1/Z \cdot \prod_{\varphi\in\Phi} e^{-\beta_\varphi \cdot \varphi}$ avec Z une constante de normalisation, Φ un ensemble de fonctions locales appelées des potentiels et β_φ une constante associée à chaque $\varphi \in \Phi$. La portée des potentiels $\varphi \in \Phi$ est définie par les cliques d'un graphe non orienté associé au champ de Markov et les potentiels ne correspondent pas à des distributions de probabilité conditionnelles locales.

Etant donné un champ de Markov modélisant P_V, une affectation la plus probable de V peut être déterminée en calculant :

$$\max_{V}(\frac{1}{Z} \times \prod_{\varphi\in\Phi} exp(-\beta_\varphi \cdot \varphi)) \tag{2.3}$$

Les fonctions locales sont combinées par un produit et des éliminations avec max sont réalisées. Intrinsèquement, un champ de Markov exploite des propriétés d'indépendance dans un graphe non orienté alors qu'un réseau bayésien exploite des propriétés d'indépendance dans un graphe orienté. Les deux modèles peuvent fournir des résultats très différents en terme de taille de la portée des fonctions locales utilisées. *Généralement*, les réseaux bayésiens sont plus utilisés pour modéliser des relations de causalité alors que les champs de Markov sont plutôt utilisés pour modéliser des corrélations spatiales.

Les réseaux bayésiens et les champs de Markov sont unifiés par le formalisme des graphes chaînés (*Chain graphs* [42]). Un graphe chaîné utilise un graphe qui contient à la fois des liens orientés et des liens non orientés, tels que les cycles dans ce graphe impliquent uniquement des liens non orientés. L'ensemble $\mathcal{C}$ des composantes connexes obtenues lorsque les liens orientés sont supprimés est appelé l'ensemble des composantes du graphe chaîné. Un graphe chaîné peut alors être vu comme un DAG dont les sommets correspondent aux composantes de $\mathcal{C}$.

Un graphe chaîné représente une distribution de probabilité jointe P_V sous la forme factorisée $P_V = \prod_{c \in \mathcal{C}} P_{c \mid pa(c)}$, chaque distribution de probabilité conditionnelle $P_{c \mid pa(c)}$ étant elle-même exprimée comme dans un champ de Markov sous une forme du type $P_{c \mid pa(c)} = \frac{1}{Z_{pa(c)}} \prod_{\varphi \in \Phi_c} e^{-\beta_\varphi \cdot \varphi}$.

Les équations 2.1, 2.2 et 2.3 utilisent un seul opérateur de combinaison et un seul opérateur d'élimination. Dans d'autres cas, plusieurs opérateurs de combinaison et/ou plusieurs opérateurs d'élimination peuvent être utilisés.

Prenons l'exemple des CSP stochastiques (*stochastic CSP* [107]). Les CSP stochastiques étendent les CSP classiques en ajoutant, en plus des variables de décision contrôlables, des variables dites contingentes qui sont non contrôlables et qui ne peuvent pas être influencées par les décisions prises. Cette dernière donnée est appelée l'hypothèse de contingence. Lorsque les variables contingentes sont mutuellement indépendantes, un ensemble P de distributions de probabilité unaires P_s sur chaque variable contingente s est disponible. Un ensemble de contraintes C est également défini, comme dans un CSP classique.

Considérons le scénario suivant : la valeur de deux variables de décision d_1 et d_2 doit être choisie, puis on observe la valeur d'une variable contingente s_1 et enfin on prend des décisions d_3 et d_4 (potentiellement en fonction de s_1) sans avoir observé une variable contingente s_2. Deux distributions de probabilité P_{s_1} et P_{s_2} sont définies sur s_1 et s_2 et certaines contraintes sont imposées sur les variables. Ces éléments définissent un CSP stochastique dit à deux étages (car il y a deux étages de décision). Chercher des règles de décision maximisant la probabilité que les contraintes soient satisfaites est équivalent à chercher des règles de décision optimales pour la quantité suivante :

$$\max_{d_1} \max_{d_2} \sum_{s_1} \max_{d_3} \max_{d_4} \sum_{s_2} (P(s_1) \times P(s_2)) \times (c_1 \times \ldots \times c_m) \tag{2.4}$$

Ainsi, toutes les fonctions locales sont combinées par un produit et des éliminations utilisant max pour les variables de décision et + pour les variables contingentes sont réalisées.

Un autre formalisme dans lequel plusieurs types d'opérateurs peuvent être utilisés est le formalisme des diagrammes d'influence [48]. Ce formalisme étend les réseaux bayésiens en leur ajoutant les notions de décision et d'utilité. Un diagramme d'influence met en jeu trois types de variables organisées dans une structure de DAG :

- des variables aléatoires ; l'ensemble des variables aléatoires est noté S et pour chaque variable $s \in S$, on spécifie une distribution de probabilité $P_{s \mid pa(s)}$ sur s sachant ses parents dans le DAG ;
- des variables de décision ; l'ensemble des variables de décision est noté $\{d_1, \ldots, d_q\}$, les indices représentant l'ordre dans lequel les décisions sont prises ; pour chaque variable de décision d, $pa(d)$ correspond à l'ensemble des variables dont la valeur est observée lorsque la décision d est prise ;
- des variables d'utilité ; l'ensemble des variables d'utilité est noté Γ ; on associe à chaque variable d'utilité u une fonction locale $U_{pa(u)}$ dont la portée est égale aux parents de u dans le DAG ; ces fonctions locales représentent une utilité globale $U_G = \sum_{u \in \Gamma} U_{pa(u)}$; les variables d'utilité doivent être des feuilles du DAG.

Le problème associé à un diagramme d'influence consiste à trouver des règles de décision maximisant l'utilité espérée. Algébriquement parlant, si on note I_0 l'ensemble des variables aléatoires observées avant de prendre la première décision, I_k l'ensemble des variables aléatoires dont la valeur est observée entre les décisions d_k et d_{k+1} et I_q l'ensemble des variables aléatoires dont la valeur n'est pas observée avant la dernière décision, alors des règles de décision optimales sont définies en utilisant argmax dans la quantité suivante :

$$\sum_{I_0} \max_{d_1} \sum_{I_1} \max_{d_2} \ldots \sum_{I_{q-1}} \max_{d_q} \sum_{I_q} ((\prod_{s \in S} P_{s \mid pa(s)}) \times (\sum_{u \in \Gamma} U_{pa(u)})) \tag{2.5}$$

Dans le calcul ci-dessus, nous combinons les probabilités par un produit, les utilités par une somme et les probabilités avec les utilités par un produit. Les variables de décision sont éliminées par un max et les variables aléatoires sont éliminées par un $+$. L'ordre dans lequel les variables sont éliminées est directement fonction de l'ordre dans lequel les décisions sont prises et les observations sont réalisées.

Il se peut également qu'intervienne un ensemble de fonctions locales de faisabilité décrivant quelles décisions sont possibles et que, comme dans les champs de Markov, les fonctions locales représentant les facteurs multiplicatifs d'une distribution de probabilité globale ne soient pas des distributions de probabilité conditionnelles. Dans ce cas, le formalisme des réseaux de valuation [98, 100, 34] peut être utilisé. Dans ce formalisme ou dans ses extensions, les calculs à effectuer pour trouver des règles de décisions optimales peuvent s'écrire sous la forme suivante :

$$\max_{d_1, d_2} \sum_{s_1} \max_{d_3, d_4} \sum_{s_2} \left(\left(\bigwedge_{F_i \in F} F_i \right) \star \left(\prod_{P_i \in P} P_i \right) \times \left(\sum_{U_i \in U} U_i \right) \right) \tag{2.6}$$

Les fonctions locales de faisabilité sont combinées par un $\wedge$ logique et combinées avec les autres fonctions locales en utilisant l'opérateur de troncature $\star$ (cf. définition 1.6 page 19). A nouveau, une séquence d'éliminations de variables est réalisée. De telles fonctions locales de faisabilités

peuvent également être utilisées pour modéliser des préconditions dans un problème de planification classique [40, 44].

Considérons maintenant le cadre des processus décisionnels markoviens (*Markov Decision Processes*, MDP [86, 68, 91]) à horizon fini. La présentation ci-dessous ne correspond pas à la présentation classique des MDP mais est une formulation équivalente.

Un MDP décrit l'évolution de l'environnement par pas de temps. A chaque pas de temps t sont associées une variable non déterministe s_t décrivant l'état de l'environnement à t et une variable de décision d_t décrivant une décision prise à t.

Dans un MDP probabiliste, les incertitudes sur l'évolution de l'environnement sont décrites par des distributions de probabilité conditionnelles locales $P_{s_{t+1} \mid s_t, d_t}$ d'être dans l'état s_{t+1} à l'étape $t+1$ sachant l'état s_t à l'instant t et la décision d_t prise à t. Des préférences sur l'environnement et sur les décisions sont exprimées par l'intermédiaire de fonctions locales de récompense additives R_{s_t, d_t} associées à chaque instant t. A chaque instant, l'état de l'environnement s_t est connu avant de prendre la décision d_t. Pour des raisons de clarté, l'état s_1 est supposé connu. S'il y a T étapes de décision, alors trouver des règles de décisions optimales peut se faire en calculant

$$\max_{d_1} \sum_{s_2} \max_{d_2} \ldots \sum_{s_T} \max_{d_T} \left(\prod_{t \in [1,T-1]} P_{s_{t+1} \mid s_t, d_t} \right) \times \left(\sum_{t \in [1,T]} R_{s_t, d_t} \right) \tag{2.7}$$

pour obtenir l'utilité espérée optimale et en utilisant argmax pour définir une politique optimale : les probabilités sont combinées par un produit, les récomponses sont combinées par une somme, les probabilités sont combinées avec les récompenses en utilisant un produit, les variables de décisions sont éliminées par un max et les variables d'environnement sont éliminées par une somme.

Dans un MDP possibiliste, le même genre d'équation peut être obtenu. La différence est que les distributions de probabilité sont remplacées par des distributions de possibilité $\pi(s_{t+1} \mid s_t, d_t)$, les récompenses additives sont remplacées par des preférences $\mu(s_t, d_t)$ combinées par un min et les opérateurs de combinaison et d'élimination utilisés conduisent à la forme suivante, dans le cas d'un MDP possibiliste dit *pessimiste* :

$$\max_{d_1} \min_{s_2} \max_{d_2} \ldots \min_{s_T} \max_{d_T} \max \left(1 - \min_{t \in [1,T-1]} \pi_{s_{t+1} \mid s_t, d_t} \ , \ \min_{t \in [1,T]} \mu_{s_t, d_t} \right) \tag{2.8}$$

Les incertitudes sont combinées par un min, les utilités sont combinées par un min, une plausibilité p et une utilité u sont combinées par $\max(1-p, u)$, les variables de décision sont éliminées par un max et les variables d'environnement sont éliminées par un min.

Ces similarités entre MDP utilisant plusieurs modèles de plausibilité et d'utilité ont été exploitées pour définir les MDP algébriques [74], qui permettent d'exprimer des problèmes qui sont équivalents à calculer des quantités du type :

$$\max_{d_1} \oplus_u \max_{s_2} \max_{d_2} \ldots \oplus_u \max_{s_T} \max_{d_T} ((\bigotimes_{p \ t \in [1,T-1]} P_{s_{t+1} \mid s_t, d_t}) \otimes_{pu} (\bigotimes_{u \ t \in [1,T]} U_{s_t, d_t})) \tag{2.9}$$

où les plausibilités sont combinées par un opérateur abstrait $\otimes_p$, les utilités sont combinées par un opérateur $\otimes_u$, plausibilités et utilités sont combinées par un opérateur $\otimes_{pu}$ et où une séquence

d'éliminations de variables est effectuée (max sur les décisions et $\oplus_u$ sur les variables d'environnement).

2.2 Les trois éléments de base d'un cadre générique pour la décision séquentielle avec incertitudes, faisabilités et utilités

Les exemples donnés précédemment montrent que nombre de requêtes classiques formulées dans des cadres existants peuvent être ramenées à un calcul d'une séquence d'éliminations de variables sur une combinaison de fonctions locales. Ces cadres existants peuvent de surcroît tous être vus comme des modèles graphiques qui diffèrent principalement de par les opérateurs d'élimination et de combinaison utilisés et de par ce que les variables et les fonctions locales représentent.

C'est ce genre d'observations qui a conduit à la définition des MDP algébriques [74] ou des algèbres de valuation [97, 98, 56], ce dernier cadre étant un cadre algébrique générique au sein duquel le problème principal est de calculer une séquence d'éliminations de variables sur une combinaison de fonctions locales. Cependant, les algèbres de valuation ne font intervenir qu'un seul opérateur de combinaison, alors que plusieurs opérateurs peuvent être nécessaires pour agréger les différents types de fonctions locales d'un modèle graphique composite. En outre, les algèbres de valuation ne font intervenir qu'un seul type d'élimination, alors que plusieurs opérateurs d'élimination peuvent être nécessaires pour éliminer les différents types de variables. Dans les réseaux de valuation [100], les plausibilités représentent nécessairement des probabilités, et des minimisations ne peuvent pas être réalisées. Tous ces arguments justifient la nécessité d'introduire un cadre plus expressif.

Afin de couvrir les requêtes formulées dans divers formalismes, la forme générale à considérer est la suivante :

$$Sov\left(\left(\bigwedge_{F_i\in F} F_i\right) \star \left(\bigotimes_{P_i\in P}{}_p\ P_i\right) \otimes_{pu} \left(\bigotimes_{U_i\in U}{}_u\ U_i\right)\right) \tag{2.10}$$

où (1) $\wedge$, $\otimes_p$, $\otimes_u$ sont utilisés respectivement pour combiner les faisabilités locales, les plausibilités locales et les utilités locales, $\otimes_{pu}$ est utilisé pour combiner plausibilités et utilités, et l'opérateur de troncature $\star$ permet d'ignorer les décisions infaisables sans avoir à gérer des éliminations sur des domaines restreints ; (2) F, P, U sont des ensembles (éventuellement vides) de fonctions locales de faisabilité, de plausibilté et d'utilité respectivement ; (3) Sov est une séquence de couple opérateur-variable(s) qui indique comment les variables doivent être éliminées. Sov fait intervenir les opérateurs d'élimination min ou max sur les variables de décision et un opérateur $\oplus_u$ sur les variables d'environnement.

L'équation 2.10 a été obtenue de manière informelle, par analogie avec les cadres existants. Afin de donner une sémantique claire au calcul purement algébrique qu'elle exprime, il est nécessaire de définir trois éléments principaux :

1. Nous devons définir les divers opérateurs de combinaison $\otimes_p$, $\otimes_u$, $\otimes_{pu}$ ainsi que l'opérateur $\oplus_u$ utilisé pour éliminer les variables d'environnement. Un opérateur d'élimination noté $\oplus_p$

permettant de synthétiser certaines informations sur les plausibilités sera également introduit. Ces divers opérateurs définissent la *structure algébrique* du cadre PFU. Naturellement, certaines propriétés algébriques seront requises. Sémantiquement parlant, ces opérateurs définissent le modèle de plausibilité/utilité.

2. Nous devons ensuite exprimer des informations sous la forme d'un modèle graphique faisant intervenir un ensemble de variables et des ensembles de fonctions locales exprimant des plausibilités, des faisabilités et des utilités (ensembles P, F, U). Ces éléments définiront des *réseaux PFU*. Nous devons également justifier la possibilité d'exprimer de la connaissance sous une telle forme factorisée, notamment via la notion d'indépendance conditionnelle.
3. Enfin, pour formuler des problèmes de décision, nous devons définir la notion de *requêtes sur des réseaux PFU* en introduisant une séquence Sov de couple opérateur-variable(s) appliquée à la combinaison de toutes les fonctions locales, comme dans l'équation 2.10. Ces requêtes doivent permettrent de modéliser des situations variées en termes d'observabilité et de contrôlabilité. Il est également nécessaire de montrer pourquoi le calcul de quantités du type de celle donnée à l'équation 2.10 est intéressant d'un point de vue sémantique, en comparant cette équation avec une approche classique à base d'arbres de décision.

2.3 Résumé

Ce chapitre a montré de manière informelle que de nombreuses requêtes formulées dans des formalismes variés raisonnant sur des plausibilités et/ou des faisabilités et/ou des utilités pouvaient être réduites à des calculs de séquences d'éliminations de variables sur des combinaisons de fonctions locales utilisant des opérateurs variés, sous une forme intuitivement couverte par celle donnée à l'équation 2.10.

Les trois éléments fondamentaux (une structure algébrique, un réseau PFU, et une séquence d'éliminations de variables) nécessaires pour définir formellement cette équation et lui donner du sens sont introduits dans les chapitres 3, 4 et 5 respectivement.

Chapitre 3

Une structure algébrique générique pour la décision séquentielle dans l'incertain

Le premier élément du cadre PFU est une structure algébrique spécifiant comment les informations fournies par les plausibilités, les faisabilités et les utilités sont combinées et synthétisées. La structure algébrique utilisée s'appuie sur les travaux de Friedman, Halpern et Chu [41, 47, 19], qui proposent une généralisation axiomatique et algébrique des notions classiques de probabilité, d'utilité et d'utilité espérée, sous les dénominations de *plausibilité*, d'*utilité* et d'*utilité espérée généralisée* (des approches similaires sont développées dans [108, 22]). La structure proposée s'écarte cependant sur certains points de leurs propositions.

3.1 Quelques définitions algébriques

Définition 3.1. $(E, \circledast)$ *est un* monoïde commutatif *ssi* E *est un ensemble et* $\circledast$ *est un opérateur binaire* $(E \times E \to E)$ *qui est associatif* $(x \circledast (y \circledast z) = (x \circledast y) \circledast z)$*, commutatif* $(x \circledast y = y \circledast x)$*, et qui possède un élément neutre* $1_E \in E$ $(x \circledast 1_E = 1_E \circledast x = x)$*.*

Définition 3.2. $(E, \oplus, \otimes)$ *est un* semi-anneau commutatif *ssi*

- $(E, \oplus)$ *est un monoïde commutatif dont l'élément neutre est noté* 0_E *;*
- $(E, \otimes)$ *est un monoïde commutatif dont l'élément neutre est noté* 1_E*,*
- 0_E *est absorbant pour* $\otimes$ $(x \otimes 0_E = 0_E)$*,*
- $\otimes$ *est distributif par rapport à* $\oplus$ $(x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z))$*.*

Définition 3.3. *Soit* E *un ensemble partiellement ordonné par* $\preceq$*. Un opérateur* $\circledast$ *sur* E *est dit monotone ssi* $(x \preceq y) \to (x \circledast z \preceq y \circledast z)$ *pour tous* $x, y, z \in E$*.*

3.2 Structure de plausibilité

L'exemple de la chasse au trésor introduit au chapitre 1 utilise des *probabilités* pour modéliser l'incertitude. Sous hypothèse d'indépendance, les probabilités sont agrégées via un opérateur $\otimes_p = \times$ et synthétisées via un opérateur $\oplus_p = +$. Mais d'autres théories peuvent être utilisées pour modéliser l'incertitude, comme par exemple la théorie des *possibilités* [36] ou celle des *fonctions de Spohn* [102]. Avec la première, une option possible consiste à utiliser $\otimes_p = \min$ pour combiner les plausibilités et $\oplus_p = \max$ pour synthétiser des plausibilités "marginales", tandis qu'avec la seconde, $\otimes_p = +$ et $\oplus_p = \min$.

Afin de manipuler des formes générales de plausibilités, nous définissons la notion de *structure de plausibilité*. Une structure de plausibilité est définie comme un triplet $(E_p, \oplus_p, \otimes_p)$ où E_p est un ensemble de *degrés de plausibilité* équipé d'un *ordre partiel* $\preceq_p$, $\oplus_p$ est un *opérateur d'élimination* et $\otimes_p$ un *opérateur de combinaison* sur les plausibilités. Nous imposons à cette structure de satisfaire les axiomes de base sur les plausibilités proposés par Friedman et Halpern dans [41, 47]. Nous étendons cependant la structure qu'ils proposent pour que $\oplus_p$ et $\otimes_p$ soient clos sur E_p. Ceci nous conduit à la définition suivante :

Définition 3.4. *Une structure de plausibilité est un triplet* $(E_p, \oplus_p, \otimes_p)$ *tel que :*

- $(E_p, \oplus_p, \otimes_p)$ *est un semi-anneau commutatif ; l'élément neutre de* $\oplus_p$ *est noté* 0_p *et l'élément neutre de* $\otimes_p$ *est noté* 1_p *;*
- E_p *est équipé d'un ordre partiel* $\preceq_p$ *dont* 0_p *est l'élément minimum ;*
- $\oplus_p$ *et* $\otimes_p$ *sont monotones pour* $\preceq_p$*.*

0_p est associé aux événements impossibles et 1_p aux événements certains. A noter qu'alors que cette définition impose que 0_p soit l'élément minimum de E_p, elle n'impose pas que 1_p en soit l'élément maximum. La structure de plausibilité associée aux probabilités est par exemple $(\mathbb{R}^+, +, \times)$, avec $\preceq_p = \leq$, $0_p = 0$ et $1_p = 1$.

3.3 Structure de faisabilité

Du fait qu'une décision est soit faisable, soit infaisable, la *structure de faisabilité* que nous utilisons n'est pas paramétrable. C'est en fait un cas particulier de structure de plausibilité : $(\{t, f\}, \vee, \wedge)$ avec $f \prec_p t$, $0_p = f$ et $1_p = t$.

Ainsi, les fonctions locales exprimant des faisabilités sont combinées par un $\wedge$ logique car une décision est faisable si et seulement si toutes les fonctions de faisabilité la juge faisable. Etant donnée une fonction locale de faisabilité F_i, une affectation A est faisable d'après F_i si et seulement s'il existe une affectation A' des variables de la portée de F_i non affectées par A telle que $F_i(A.A') = t$. Ceci explique pourquoi les éliminations sur les faisabilités se font via un $\vee$ logique.

3.4 Structure d'utilité

L'exemple de la chasse au trésor utilise des *utilités additives* (des coûts et des gains) pour modéliser les préférences. Elles sont agrégées via un opérateur $\otimes_u = +$. Mais si les préférences sont

modélisées par des *priorités*, elles sont agrégées via $\otimes_u = \min$. D'autres opérateurs d'agrégation des utilités peuvent également être utilisés.

Une *structure d'utilité* est définie comme une paire $(E_u, \otimes_u)$ où E_u est un ensemble de *degrés d'utilité* équipé d'un *ordre partiel* $\preceq_u$ et $\otimes_u$ un *opérateur de combinaison* sur les utilités. Des axiomes classiques sur les utilités nous conduisent à la définition suivante :

Définition 3.5. *Une structure d'utilité est une paire* $(E_u, \otimes_u)$ *telle que :*

— $(E_u, \otimes_u)$ *est un monoïde commutatif dont l'élément neutre est noté* 1_u *;*

— E_u *est équipé d'un ordre partiel* $\preceq_u$ *et* $\otimes_u$ *est monotone.*

1_u est associé aux situations indifférentes du point de vue de l'agrégation des utilités. La structure d'utilité associée aux utilités additives classiques est par exemple $(\mathbb{R}, +)$, avec $\preceq_u = \leq$ et $1_u = 0$.

La distinction entre plausibilités, faisabilités et utilités est importante et peut être justifiée par des arguments purement algébriques. Etant donné que les opérateurs $\otimes_p$ et $\otimes_u$ peuvent être différents (par exemple $\otimes_p = \times$ et $\otimes_u = +$ dans la théorie de l'utilité espérée probabiliste avec utilités additives), il est tout d'abord nécessaire de distinguer plausibilités et utilités.

Il est aussi nécessaire de distinguer les faisabilités des utilités et des plausibilités. En effet, prenons l'exemple d'un jeu de cartes où chaque joueur doit jouer deux cartes prises parmi des valets, des dames et des rois. Jouer un valet (respectivement une dame, un roi) rapporte 5, 10 et 20 points respectivement. La décision la meilleure est alors évidemment de jouer deux fois un roi et la décision la pire consiste à jouer deux fois un valet. Mais supposons que jouer deux fois la même carte soit interdit.

Dans ce cas, si l'on veut calculer des décisions optimales, il est nécessaire de restreindre les opérations d'optimisation à la partie faisable du domaine des variables de décision. Comme aucun élément de l'ensemble des degrés d'utilité E_u ne peut être ignoré à la fois par l'opérateur min et par l'opérateur max, les faisabilités doivent être l'objet d'un traitement spécifique. Intuitivement, l'infaisabilité n'est pas une notion relative (alors que l'utilité l'est) : l'infaisabilité correspond à des règles communément admises par tous les agents décideurs et se situe ainsi hors de toute échelle d'utilité.

Afin d'ignorer les valeurs infaisables des variables de décision, nous utilisons l'opérateur de troncature $\star$ introduit à la définition 1.6 page 19. Afin d'éliminer une variable de décision d'une fonction locale φ tout en ignorant les décisions infaisables spécifiées par une fonction de faisabilité F_i, il suffit alors d'éliminer x sur $(F_i \star \varphi)$ au lieu de φ, de manière à associer la valeur infaisable $\Diamond$ aux décisions infaisables.

3.5 Structure d'utilité espérée

Dans la théorie de l'*utilité espérée probabiliste*, la formule classique $\sum_i p_i \times u_i$ traduit une agrégation des probabilités et des utilités via un opérateur $\otimes_{pu} = \times$ et une synthèse du résultat via un opérateur $\oplus_u = +$. Cette formule se généralise sous la forme $\bigoplus_{u,i} (p_i \otimes_{pu} u_i)$, avec par exemple $\otimes_{pu} = \min$ et $\oplus_u = \max$ dans la théorie de l'*utilité espérée possibiliste optimiste* [37] ou $\otimes_{pu} = +$ et $\oplus_u = \min$ dans la théorie de l'*utilité* espérée avec *fonctions de Spohn* [45] (avec des utilités uniquement positives).

Afin de généraliser ces structures existantes, nous définissons la notion de *structure d'utilité espérée*. Une structure d'utilité espérée est définie comme un quadruplet $(E_p, E_u, \oplus_u, \otimes_{pu})$ où $\oplus_u$ est un opérateur d'élimination sur les utilités et $\otimes_{pu}$ un opérateur de combinaison entre plausibilités et utilités. Nous imposons à cette structure de satisfaire les axiomes de base sur l'utilité espérée généralisée proposés dans [19]. Pour prendre en compte l'aspect éventuellement *séquentiel* de la décision, nous imposons cependant des axiomes supplémentaires justifiés par la théorie des loteries [106]. Ceci nous conduit à la définition suivante :

Définition 3.6. *Une structure d'utilité espérée est un quadruplet $(E_p, E_u, \oplus_u, \otimes_{pu})$ tel que :*

- *$(E_u, \oplus_u, \otimes_{pu})$ est un semi-module sur $(E_p, \oplus_p, \otimes_p)$, ce qui implique que $(E_p, \oplus_p, \otimes_p)$ soit un semi-anneau commutatif, que $(E_u, \oplus_u)$ soit un monoïde commutatif avec un élément neutre noté 0_u et que l'opérateur binaire $\otimes_{pu}$ ($E_p \times E_u \to E_u$) satisfasse les propriétés suivantes :*
  - *distributivité par rapport à $\oplus_p$ et $\oplus_u$;*
  - *$\forall p_1, p_2 \in E_p, \forall u \in E_u, p_1 \otimes_{pu} (p_2 \otimes_{pu} u) = (p_1 \otimes_p p_2) \otimes_{pu} u$;*
  - *$\forall u \in E_u, (0_p \otimes_{pu} u = 0_u) \wedge (1_p \otimes_{pu} u = u)$;*
- *$\oplus_u$ est monotone et $\otimes_{pu}$ monotone à droite pour $\preceq_u$.*

0_u est associé aux situations *indifférentes* du point de vue de l'élimination des utilités. La structure d'utilité espérée associée à l'utilité espérée probabiliste est par exemple $(\mathbb{R}^+, \mathbb{R}, +, \times)$, avec $0_u = 0$. Toutes les structures présentées dans le tableau 3.1 sont des structures d'utilité espérée.

	E_p	$\preceq_p$	$\oplus_p$	$\otimes_p$	$0_p, 1_p$	E_u	$\preceq_u$	$\otimes_u$	$\oplus_u$	$\otimes_{pu}$	$\bot_u, 0_u, 1_u$
1	$\mathbb{R}^+$	$\leq$	$+$	$\times$	$0, 1$	$\mathbb{R} \cup \{-\infty\}$	$\leq$	$+$	$+$	$\times$	$-\infty, 0, 0$
2	$\mathbb{R}^+$	$\leq$	$+$	$\times$	$0, 1$	$\mathbb{R}^+$	$\leq$	$\times$	$+$	$\times$	$0, 0, 1$
3	$[0, 1]$	$\leq$	max	min	$0, 1$	$[0, 1]$	$\leq$	min	max	min	$0, 0, 1$
4	$[0, 1]$	$\leq$	max	min	$0, 1$	$[0, 1]$	$\leq$	min	min	$\max(1-p, u)$	$0, 1, 1$
5	$\mathbb{N} \cup \{\infty\}$	$\geq$	min	$+$	$\infty, 0$	$\mathbb{N} \cup \{\infty\}$	$\geq$	$+$	min	$+$	$\infty, \infty, 0$
6	$\{t, f\}$	$\preceq_{bool}$	$\vee$	$\wedge$	f, t	$\{t, f\}$	$\preceq_{bool}$	$\wedge$	$\vee$	$\wedge$	f, f, t
7	$\{t, f\}$	$\preceq_{bool}$	$\vee$	$\wedge$	f, t	$\{t, f\}$	$\preceq_{bool}$	$\wedge$	$\wedge$	$\to$	f, t, t
8	$\{t, f\}$	$\preceq_{bool}$	$\vee$	$\wedge$	f, t	$\{t, f\}$	$\preceq_{bool}$	$\vee$	$\vee$	$\wedge$	f, f, f
9	$\{t, f\}$	$\preceq_{bool}$	$\vee$	$\wedge$	f, t	$\{t, f\}$	$\preceq_{bool}$	$\vee$	$\wedge$	$\to$	f, t, f

TABLE 3.1 – Ensembles et opérateurs utilisés dans plusieurs cadres classiques : (1) utilité espérée probabiliste avec utilités additives (permet de calculer l'espérance d'un gain ou d'un coût), (2) utilité espérée probabiliste avec utilités multiplicatives (permet de calculer la probabilité que des contraintes soient satisfaites), (3) utilité espérée possibiliste optimiste, (4) utilité espérée possibiliste pessimiste, (5) utilité qualitative avec kappa-rankings et utilités uniquement positives, (6) utilité espérée booléenne optimiste avec utilités conjonctives (permet de savoir s'il existe un monde possible dans lequel tous les buts d'un ensemble de buts B sont satisfaits), (7) utilité espérée booléenne pessimiste avec utilités conjonctives (permet de savoir si dans tous les mondes possibles tous les buts d'un ensemble de buts B sont satisfaits), (8) utilité espérée booléenne optimiste avec utilités disjonctives (permet de savoir s'il existe un monde possible dans lequel au moins un but d'un ensemble de buts B est satisfait), (9) utilité espérée booléenne pessimiste avec utilités disjonctives (permet de savoir si dans tous les mondes possibles au moins un but d'un ensemble de buts B est satisfait).

Le problème du repas d'affaire Pour illustrer les définitions précédentes et à venir, considérons l'exemple suivant. *Pierre invite Jean and Marie (un couple divorcé) à un repas d'affaire pour les convaincre d'investir dans son entreprise. Pierre sait que si Jean est présent à la fin du dîner, il*

investira $10K€$ *et que si Marie est présente à la fin du dîner, elle investira* $50K€$*. Pierre sait que Jean et Marie ne seront pas présents ensemble (car l'un deux doit garder leur fils), qu'au moins l'un d'entre eux viendra et que le cas "Jean vient et Marie ne vient pas" se produit avec une probabilité* 0.6*. Concernant le menu, Pierre peut commander du poisson ou de la viande pour le plat principal et du vin rouge ou du vin blanc pour le vin. Cependant, le restaurant refuse de servir du poisson avec du vin rouge. Jean n'aime pas le vin blanc et Marie n'aime pas la viande. Si le menu ne leur convient pas, alors ils quitteront la table. Si Jean vient, Pierre ne veut pas qu'il parte car il est son meilleur ami.*

Exemple 3.7. *Le problème du dîner utilise la structure d'utilité espérée associée à l'utilité espérée additive probabiliste (ligne 1) : la structure de plausibilité est* $(\mathbb{R}^+, +, \times)$*,* $\oplus_u = +$*,* $\otimes_{pu} = \times$*, et les utilités sont des gains additifs :* $(E_u, \otimes_u) = (\mathbb{R} \cup \{-\infty\}, +)$*, avec la convention que* $u + (-\infty) = -\infty$*.*

Hypothèses implicites et cadres non couverts Les hypothèses faites sur les structures de plausibilité, d'utilité et d'utilité espérée résultent de compromis toujours discutables entre des considérations sémantiques et algorithmiques : pouvoir englober le plus grand nombre possible de situations et d'approches, garder à l'esprit que les problèmes concrets doivent pouvoir être représentés de façon suffisamment compacte et résolus de façon raisonnablement efficace.

Par exemple, le fait de supposer l'existence d'ensembles E_p et E_u de degrés de plausibilité et d'utilité impose que plausibilités et utilités soient *cardinales*. Des approches purement ordinales comme par exemple les réseaux de préférences conditionnelles (*CP-nets* [14]) ne sont pas couvertes par le cadre proposé.

De la même façon, le fait que l'agrégation d'une plausibilité avec une utilité produise une utilité ($\otimes_{pu} : E_p \times E_u \to E_u$) repose sur une hypothèse implicite de *commensurabilité* entre plausibilités et utilités. Des travaux tels que [39], qui ne font pas cette hypothèse, ne sont pas couverts.

Enfin, les axiomes imposés impliquent que seules des plausibilités à caractère *distributionnel* sont couvertes : la plausibilité d'un ensemble E d'affectations des variables est supposée être complètement déterminée par la plausibilité de chacune des affectations complètes couvertes par E. De ce fait, des cadres non distributionnels comme les *fonctions de croyance* de Dempster-Shafer [96] ne sont pas couverts.

3.6 Résumé

Dans ce chapitre, nous avons introduit la notion de *structure d'utilité espérée*, qui constitue le premier élément clé du cadre PFU. Les structures algébriques définies spécifient comment les plausibilités sont combinées et synthétisées (via $\otimes_p$ et $\oplus_p$), comment les utilités sont combinées (via $\otimes_u$) et comment plausibilités et utilités sont prises en compte simultanément (via $\oplus_u$ et $\otimes_{pu}$). Plus précisément, les structures algébriques de base utilisées sont :

— un semi-anneau commutatif $(E_p, \oplus_p, \otimes_p)$ pour manipuler des plausibilités,
— un monoïde commutatif $(E_u, \otimes_u)$ pour manipuler des utilités,
— un semimodule $(E_p, E_u, \oplus_p, \otimes_{pu})$ pour calculer des utilités espérées.

L'ajout d'axiomes de monotonie à ces structures algébriques classiques permet d'obtenir ce que nous appelons structure de plausibilité, structure d'utilité, et structure d'utilité espérée. Ces structures

couvrent des modèles variés de modélisation des plausibilités et des utilités. Elles sont inspirées de structures existantes définies par Friedman, Chu et Halpern [41, 47, 19]. Les différences principales tiennent à l'ajout d'axiomes permettant de traiter des problèmes de décision séquentielle (comportant plusieurs étapes de décision) et à l'ajout d'axiomes pour des raisons d'efficacité des algorithmes futurs.

Chapitre 4

Réseaux de Plausibilité-Faisabilité-Utilité

Le second élément du cadre PFU est un réseau de fonctions locales P_i, F_i et U_i (cf. équation 2.10 page 30) portant sur des variables d'un ensemble V. Ce réseau définit une représentation compacte et structurée des quantités globales que sont les plausibilités sur l'état de l'environnement, les faisabilités sur les décisions et les utilités sur l'environnement et les décisions. Ce chapitre définit de tels réseaux PFU et analyse les relations entre d'un côté l'expression de quantités globales par une combinaison de quantités locales et de l'autre la notion d'indépendance conditionnelle.

Nous appellons dorénavant *fonction de plausibilité* une fonction locale à valeurs dans E_p (l'ensemble des degrés de plausibilité), *fonction de faisabilité* une fonction locale à valeurs dans $\{t, f\}$ (l'ensemble des degrés de faisabilité) et *fonction d'utilité* une fonction locale à valeurs dans E_u (l'ensemble des degrés d'utilité).

4.1 Variables de décision et variables d'environnement

Les exemples introduits jusqu'alors mettent en évidence une distinction importante entre variables de décision, contrôlées par un agent décideur, et variables d'environnement, non contrôlées par un agent décideur.

En conséquence, le premier élément de la structure graphique proposée est un ensemble V de *variables à domaines finis*, partitionné en deux sous-ensembles : un ensemble V_D de *variables de décision* et un ensemble V_E de *variables d'environnement.*

Exemple 4.1. *Pour modéliser le problème du repas d'affaire, nous introduisons six variables : bp_J et bp_M (valeur t ou f), qui représentent respectivement les présences de Jean et Marie au début du repas, ep_J et ep_M (valeur t ou f), qui représentent leur présence en fin de repas, mc (valeur fish ou meat), qui représente le choix du plat principal, et w (valeur white ou red), qui représente le choix du vin. Ainsi, nous avons $V_D = \{mc, w\}$ et $V_E = \{bp_J, bp_M, ep_J, ep_M\}$.*

4.2 Vers des fonctions locales de faisabilité et de plausibilité

L'utilisation de fonctions locales pour représenter une quantité globale soulève certaines questions : comment et pourquoi des fonctions locales peuvent-elles être utilisées pour représenter des quantités globales et inversement, lorsque des fonctions locales sont exprimées, quel sens donner à leur agrégation. Nous montrons que ces questions sont fortement liées à une sorte d'équivalence entre factorisation et indépendance conditionnelle.

4.2.1 Une première étape de factorisation utilisant des indépendances conditionnelles

Il est tout d'abord possible d'étendre un résultat central sur les réseaux bayésiens [73] qui fait le lien entre le DAG associé à un réseau bayésien et la factorisation de la distribution de probabilité jointe exprimée par ce réseau. Ce résultat utilise la notion de *compatibilité* entre un DAG et une distribution de probabilité jointe : un DAG G sur un ensemble de variables V est dit compatible avec une distribution de probabilité jointe sur V si et seulement si chaque variable $x \in V$ est conditionnellement indépendante de ses non-descendants dans G, étant donnés ses parents dans G. Ceci conduit au théorème suivant : si G est un DAG sur un ensemble de variables V qui est compatible avec une distribution de probabilité jointe $\mathcal{P}_V$ sur V, alors $\mathcal{P}_V = \prod_{x \in V} \mathcal{P}_{x|pa_G(x)}$.

L'extension de ce résultat probabiliste au cadre plus général des plausibilités nécessite d'étendre la notion d'*indépendance conditionnelle*. Encore une fois, nous nous appuyons sur les travaux de Friedman et Halpern [41, 47], tout en nous en écartant sur certains aspects. Si S est un ensemble de variables, nous définissons une *distribution de plausibilité* sur S comme une fonction $\mathcal{P}_S$ de $dom(S)$ dans E_p, telle que $\bigoplus_{p\ A \in dom(S)} \mathcal{P}_S(A) = 1_p$. Une distribution de plausibilité sur S induit une distribution de plausibilité sur tout $S' \subseteq S$: $\mathcal{P}_{S'} = \bigoplus_{p\ dom(S-S')} \mathcal{P}_S$. A partir de là, pour des structures de plausibilités dites *conditionnables*, il est possible d'introduire la notion de *distribution de plausibilité conditionnelle* sur S_1 sachant S_2 (S_1 et S_2 disjoints), qui vérifie certaines propriétés telles que $\mathcal{P}_{S_1,S_2} = \mathcal{P}_{S_1 \mid S_2} \otimes_p \mathcal{P}_{S_2}$. Sur cette base, nous disons que S_1 est *conditionnellement indépendant* de S_2 sachant S_3 (S_1, S_2 et S_3 disjoints) si et seulement si $\mathcal{P}_{S_1,S_2|S_3} = \mathcal{P}_{S_1|S_3} \otimes_p \mathcal{P}_{S_2|S_3}$. Dans le cas des réseaux bayésiens, le DAG défini est un DAG dont les sommets représentent des variables. Il est parfois plus naturel d'utiliser un DAG dont les sommets correspondent à des ensembles de variables, comme ce qui est fait dans les graphes chaînés. Nous disons aussi qu'un DAG G sur un ensemble de composantes C (une composante étant un ensemble de variables) est compatible avec une distribution de plausibilité jointe sur $S = \cup_{c \in C} c$ si et seulement si chaque composante $c \in G$ est conditionnellement indépendante de ses non-descendantes dans G, étant données ses parentes dans G. Il est alors possible d'établir le théorème suivant :

Théorème 4.2. *Si un DAG G sur un ensemble de composantes C est compatible avec une distribution de plausibilité $\mathcal{P}_S$ jointe sur $S = \bigcup_{c \in C} c$, alors $\mathcal{P}_S = \bigotimes_{p\ c \in C} \mathcal{P}_{c|pa_G(c)}$.*

Le théorème 4.2 nous fournit un premier niveau de factorisation composante par composante de la distribution de plausibilité jointe, via quelques étapes techniques permettant de pallier la présence de variables de différentes natures (variables de décision et variables d'environnement).

4.2.2 Etapes de factorisations supplémentaires

Mais il est possible d'aller plus loin puisque, pour chaque composante $c \in C$, la distribution de plausibilité conditionnelle $\mathcal{P}_{c|pa_G(c)}$ peut être elle-même factorisée en fonctions locales de plausibilité P_i associées à c. Si $Fact(c)$ représente l'ensemble des facteurs permettant d'exprimer $\mathcal{P}_{c|pa_G(c)}$, nous pouvons écrire $\mathcal{P}_{c|pa_G(c)} = \underset{P_i \in Fact(c)}{\otimes_p} P_i$. Cette seconde factorisation implique notamment que

$$\underset{c}{\oplus_p} (\underset{P_i \in Fact(c)}{\otimes_p} P_i) = 1_p$$

Ainsi, le résultat de l'agrégation des fonctions locales associées à une composante c est bien une distribution de plausibilité conditionnelle.

Par exemple, dans le cas d'un réseau bayésien, rien ne s'oppose à ce que les probabilités conditionnelles $\mathcal{P}_{x|pa_G(x)}$ s'expriment comme un produit de fonctions plus locales, éventuellement sous forme de contraintes en cas d'information binaire (0 ou 1 ; voir par exemple [29]). Dans certains cas, exprimer une décomposition de $\mathcal{P}_{c\,|\,pa_G(c)}$ est assez naturel. Dans d'autres cas, comme pour les champs de Markov [18], de telles factorisations supplémentaires peuvent être obtenues en utilisant des techniques systématiques. Enfin, on peut envisager d'utiliser d'autres définitions d'indépendance conditionnelle, comme par exemple celle utilisée pour les réseaux de valuation [99]. Ces étapes de factorisation supplémentaires sont intéressantes car réduire la taille des portées des fonctions locales utilisées peut être très avantageux d'un point de vue algorithmique.

Comme la structure de faisabilité est un cas particulier de structure de plausibilité, le même type de résultat peut être établi concernant les faisabilités, c'est-à-dire qu'une distribution de faisabilité globale peut s'exprimer comme une combinaison (plus exactement une conjonction) de faisabilités conditionnelles $\mathcal{F}_{c\,|\,pa_G(c)}$, chaque facteur $\mathcal{F}_{c\,|\,pa_G(c)}$ pouvant lui-même être exprimé comme une combinaison de fonctions locales.

Ceci nous montre finalement une façon naturelle permettant obtenir des fonctions locales de plausibilité et de faisabilité et un DAG de composantes représentant certaines normalisations.

Exemple 4.3. *Considérons le problème du repas d'affaire pour illustrer les deux étapes de factorisation. Afin d'obtenir le DAG, nous pouvons identifier des indépendances conditionnelles telles que "la présence de Jean en fin de repas est conditionnellement indépendante de la présence de Marie en fin de repas sachant la présence de Jean au début et le menu choisi". Les indépendances conditionnelles peuvent nous mener à définir le DAG de composantes donné à la figure 4.1(a). Nous avons par exemple une composante* $\{bp_J, bp_M\}$ *qui contient deux variables qui sont corrélées (on ne peut pas dire si* bp_J *a une influence causale sur* bp_M *ou si* bp_M *a une influence causale sur* bp_J*, c'est pourquoi il peut être naturel de mettre ces deux variables dans une même composante).*

Le théorème 4.2, qui définit en quelque sorte la première étape de factorisation, nous assure que les fonctions globales de plausibilité et de faisabilité se factorisent de la manière suivante : $\mathcal{P}_{V_E\,|\,V_D} = \mathcal{P}_{bp_J,bp_M} \times \mathcal{P}_{ep_J\,|\,bp_J,bp_M,mc,w} \times \mathcal{P}_{ep_M\,|\,bp_J,bp_M,mc,w}$ *et* $\mathcal{F}_{V_D\,|\,V_E} = \mathcal{F}_{mc,w}$.

La seconde étape de factorisation revient à exprimer chacun des facteurs ci-dessus en utilisant une combinaison de fonctions locales. Le facteur $\mathcal{P}_{bp_J,bp_M}$ *peut être exprimé comme une combinaison de fonctions locales de plausibilités. La première,* P_1*, spécifie les probabilités de présences de Jean et Marie en début de repas :* P_1 *est défini par* $P_1((bp_J,t).(bp_M,f)) = 0.6$, $P_1((bp_J,f).(bp_M,t)) = 0.4$, *et* $P_1((bp_J,t).(bp_M,t)) = P_1((bp_J,f).(bp_M,f)) = 0$. *Il est possible d'ajouter une information*

déterministe redondante via une seconde fonction de plausibilité P_2, définie comme la contrainte $bp_J \neq bp_M$ ($P_2(A) = 1$ si la contrainte est satisfaite, 0 sinon). Nous obtenons alors $\mathcal{P}_{bp_J,bp_M} = P_1 \otimes_p P_2$ et $Fact(\{bp_J, bp_M\}) = \{P_1, P_2\}$.

$\mathcal{P}_{ep_J \mid bp_J,bp_M,mc,w}$ peut également être spécifié comme une combinaison de deux fonctions de plausibilité P_3 et P_4. P_3 exprime que Jean n'est pas présent à la fin du repas s'il ne l'était pas au début : P_3 correspond à la contrainte dure $(bp_J = f) \rightarrow (ep_J = f)$. La fonction locale $P_4 : (bp_J = t) \rightarrow ((ep_J = t) \leftrightarrow (w \neq white))$ est une contrainte dure qui indique que Jean quitte le dîner si et seulement si du vin blanc est choisi. Ainsi, nous obtenons $\mathcal{P}_{ep_J \mid bp_J,bp_M,mc,w} = P_3 \otimes_p P_4$ et $Fact(\{ep_J\}) = \{P_3, P_4\}$. De manière analogue, $\mathcal{P}_{ep_M \mid bp_J,bp_M,mc,w} = P_5 \otimes_p P_6$, avec P_5, P_6 définies comme des contraintes et $Fact(\{ep_M\}) = \{P_5, P_6\}$.

Concernant les faisabilités, $\mathcal{F}_{mc,w}$ peut être exprimé par une contrainte dure indiquant que commander du vin rouge avec du poisson n'est pas permis : $F_1 : \neg((mc = fish) \wedge (w = red))$ et $Fact(\{mc, w\}) = \{F_1\}$. L'association entre fonctions locales et composantes est représentée à la figure 4.1(a).

4.3 Fonctions locales d'utilité

Nous introduisons enfin des fonctions locales d'utilité portant indifféremment sur les variables de décision ou d'environnement. Soit ces fonctions locales sont directement définies, comme ce qui est fait dans les CSP ou les diagrammes d'influence, soit elles peuvent être obtenues à partir de travaux tels que [3], qui font le lien entre factorisation et indépendance conditionnelle pour les utilités lorsque $\otimes_u = +$.

Exemple 4.4. *Dans l'exemple du repas d'affaire, nous pouvons définir trois fonctions locales d'utilité. Une première fonction d'utilité binaire U_1 exprime que Pierre ne veut pas que Jean quitte le dîner : U_1 correspond à la contrainte dure $(bp_J = t) \rightarrow (ep_J = t)$ ($U_1(A) = 0$ si la contrainte est satisfaite, $-\infty$ sinon). Deux fonctions d'utilité unaires U_2 et U_3, définies par $U_2((ep_J, t)) = 10$ et $U_2((ep_J, f)) = 0$ et $U_3((ep_M, t)) = 50$ et $U_3((ep_M, f)) = 0$ respectivement, indiquent les gains obtenus en fonction des personnes présentes en fin de repas. Toutes les fonctions locales introduites jusqu'alors sont représentées à la figure 4.1(b).*

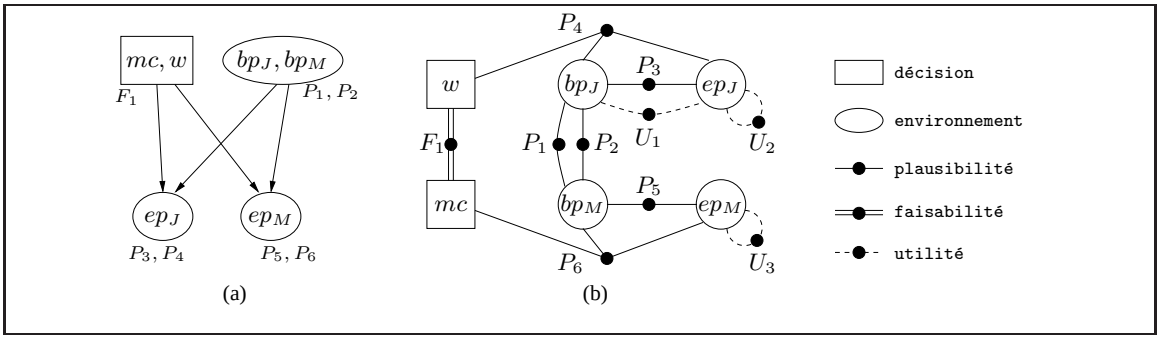

Figure 4.1: (a) DAG de composantes (b) Réseau de fonctions locales.

4.4 Définition formelle d'un réseau PFU

Ceci nous conduit à la définition suivante d'un réseau PFU :

Définition 4.5. *Un* réseau PFU *sur une structure d'utilité espérée est un quintuplet* (V, G, P, F, U) *tel que*

- $V = \{x_1, x_2, \ldots\}$ *est un ensemble fini de variables à domaines finis, partitionné en un ensemble* V_D *de variables de décision et un ensemble* V_E *de variables d'environnement*
- G *est un DAG dont les sommets, appelés composantes, sont une partition de* V*, plus précisément l'union d'une partition* $\mathcal{C}_D$ *de* V_D *et d'une partition* $\mathcal{C}_E$ *de* V_E*.*
- $P = \{P_1, P_2, \ldots\}$ *est un ensemble fini de fonctions de plausibilité. Chaque* $P_i \in P$ *est associée à une unique composante* $c \in \mathcal{C}_E$ *qui satisfait* $sc(P_i) \subset c \cup pa_G(c)$*. L'ensemble des fonctions de plausibilité* $P_i \in P$ *associées à une composante* $c \in \mathcal{C}_E$ *est noté* $Fact(c)$ *et doit satisfaire* $\underset{c}{\oplus_p} \left(\otimes_{p\, P_i \in Fact(c)} P_i \right) = 1_p$*.*
- $F = \{F_1, F_2, \ldots\}$ *est un ensemble fini de fonctions de faisabilité. Chaque* $F_i \in F$ *est associée à une unique composante* $c \in \mathcal{C}_D$ *qui satisfait* $sc(F_i) \subset c \cup pa_G(c)$*. L'ensemble des fonctions de faisabilité* $F_i \in F$ *associées à une composante* $c \in \mathcal{C}_D$ *est noté* $Fact(c)$ *et doit satisfaire* $\underset{c}{\vee} \left(\wedge_{F_i \in Fact(c)} F_i \right) = t$*.*
- $U = \{U_1, U_2, \ldots\}$ *est un ensemble fini de fonctions d'utilité.*

4.5 Liens avec des cadres existants

Le cadre des réseaux PFU couvre de nombreux cadres existant en Intelligence Artificielle.

Par exemple, un *problème de satisfaction de contraintes* (CSP) Pb peut être modélisé comme un PFU $(V, G, \emptyset, \emptyset, U)$ où V contient toutes les variables de Pb, considérées comme des variables de décision, où G contient une seule composante englobant toutes les variables et où U contient toutes les contraintes de Pb, considérées comme des fonctions locales d'utilité (si elles étaient considérées comme des fonctions locales de faisabilité, les conditions de normalisation interdiraient de considérer des CSP incohérents). La démarche est identique pour un *problème de satisfiabilité d'une formule booléenne* (SAT), à la seule différence que les variables sont booléennes et les contraintes des clauses. Il en est de même pour les CSP *valués* et, malgré les apparences, pour les *formules booléennes quantifiées* (QBF) et les CSP *quantifiés* (QCSP) ; les différences apparaîtront au niveau des requêtes. Avec les CSP *mixtes*, la distinction est faite entre variables de décision et variables d'environnement. Avec les *problèmes de satisfiabilité stochastiques* et les CSP *stochastiques*, des composantes apparaissent et des fonctions locales de plausibilité s'ajoutent aux fonctions locales d'utilité.

Un *réseau bayésien* Pb peut être modélisé comme un réseau PFU $(V, G, P, \emptyset, \emptyset)$ où V contient toutes les variables de Pb, considérées comme des variables d'environnement, où une composante est associée à chaque variable, où le DAG G est celui de Pb et où P contient les distributions de probabilité conditionnelles de Pb (une par variable), considérées comme des fonctions locales de plausibilité.

Un *diagramme d'influence* Pb peut aussi être modélisé comme un PFU $(V, G, P, \emptyset, U)$: V contient les variables d'environnement et les variables de décision de Pb ; une composante est associée à chaque variable ; G est le DAG associé à Pb sans les variables d'utilité et sans les arcs

vers les variables de décision et d'utilité ; P contient des fonctions locales de plausibilité correspondant aux distributions de probabilité conditionnelles sur les variables d'environnement et U contient des fonctions locales d'utilité (une associée à chaque variable d'utilité u de Pb, dont la portée est l'ensemble des parents de u dans le DAG associé à Pb). Avec les *réseaux de valuation*, apparaissent en plus des fonctions locales de faisabilité.

Un *processus décisionnel markovien* (MDP) à horizon fini Pb peut aussi être modélisé comme un PFU $(V, G, P, \emptyset, U)$: pour chaque instant i, une variable d'environnement s_i est associée à l'état à l'instant i et une variable de décision d_i à la décision à l'instant i ; une composante est associée à chaque variable ; G est le DAG où, pour chaque instant i, s_{i+1} a pour parent s_i et d_i ; P contient des fonctions locales de plausibilité correspondant aux probabilités de transition $P_{s_{i+1}|s_i,d_i}$ et U contient des fonctions locales d'utilité correspondant aux gains locaux U_{s_i,d_i}. Pour les *processus partiellement observables* (POMDP), il est nécessaire d'ajouter, pour chaque instant i, une variable d'environnement o_i associée à l'observation à l'instant i, d'indiquer dans le DAG que pour chaque instant i, o_i a pour parent s_i et d'ajouter les probabilités d'observation $P_{o_i|s_i}$ aux fonctions locales de plausibilité. Avec les MDP *factorisés*, les variables s_i sont décomposées en autant de variables que de facteurs. Avec les MDP *possibilistes* [91], rien ne change, hormis les opérateurs d'agrégation et d'élimination utilisés.

Il en est de même pour un *problème de planification* à horizon fixé h dans le cadre STRIPS. Il peut être modélisé comme un PFU (V, G, P, F, U) : une variable d'environnement s_i est associée à chaque instant i et à chaque composante booléenne de l'état ; une variable de décision d_i est associée à chaque instant ; G est le DAG où, pour chaque instant i, d_i a *a priori* pour parents tous les s_i, et s_{i+1} a pour parents d_i et tous les s_i ; P contient des fonctions locales de plausibilité correspondant aux effets des actions, F contient des fonctions locales de faisabilité correspondant aux pré-conditions des actions et U contient des fonctions d'utilité booléenne décrivant les différents buts et portant sur les variables s_h.

4.6 Résumé

Dans ce chapitre, nous avons introduit le second élément du cadre PFU : un réseau de fonctions locales de plausibilité, de faisabilité et d'utilité, avec un DAG capturant des conditions de normalisation. La factorisation de plausibilités, faisabilités et utilités globales en fonctions locales a été reliée à la notion d'indépendance conditionnelle. Ceci nous permet d'obtenir une méthode constructive pour spécifier des fonctions locales représentant une fonction globale donnée. D'un point de vue purement technique, la définition des réseaux PFU (définition 4.5) est relativement succinte.

Chapitre 5

Requêtes sur un réseau PFU

Une requête correspond à une tâche de raisonnement concernant l'information exprimée par un réseau PFU. Des exemples de requêtes informelles concernant le problème du repas d'affaire sont les suivantes :

1. "Quel est le meilleur choix de menu si Pierre ne sait pas qui est présent au début ?"
2. "Quel est le meilleur choix de menu si Pierre sait exactement qui vient ?"
3. "Comment maximiser l'investissement espéré si le restaurant choisit le plat principal en premier et si Pierre est pessimiste concernant ce choix, si ensuite la présence des convives est observée, et si enfin Pierre choisit le vin ?"

La distinction entre réseaux PFU et requêtes sur un réseau PFU est notamment cohérente avec la démarche actuelle utilisée dans le cadre des diagrammes d'influence qui consiste à relaxer les liens appelés *liens d'information* (*Unconstrained Influence Diagrams* [52], *Limited Memory Influence Diagrams* [61]). Elle rend explicite le fait que les requêtes ne changent pas les relations locales qui existent entre les variables.

Dans ce chapitre, nous définissons une classe de requêtes sur des réseaux PFU. Nous supposons qu'une *séquence de décisions* doit être effectuée et que l'ordre dans lequel les décisions sont prises et les observations sont réalisées est connu. Nous faisons aussi une hypothèse de *no-forgetting*, c'est-à-dire que lorsqu'une décision est prise par un agent, cet agent se rappele de toutes les décisions et de toutes les observations préalablement réalisées. A partir de maintenant, nous supposons également que l'ensemble des degrés d'utilité est *totalement* ordonné. Dans le contexte d'un calcul et d'une exécution automatique de décisions, cette hypothèse d'ordre total, qui est valide dans des formalismes variés, permet de toujours pouvoir identifier des règles de décision optimales. Voir la section 5.6 pour une discussion sur l'extension de ce travail au cas d'un ordre partiel sur les utilités.

Deux définitions de la réponse à une requête sont fournies, la première formalisant un processus de raisonnement fondé sur des arbres de décision et la seconde étant plus opérationnelle. Il est montré que les deux définitions introduites sont équivalentes.

5.1 Modélisation d'une requête

Les exemples de requêtes introduits au chapitre 1 montrent des séquences d'éliminations de variables qui sont à chaque fois cohérentes avec l'ordre des décisions et des observations et qui utilisent des opérateurs d'élimination fonctions de la nature des variables (décision ou environnement) et, pour les variables de décision, fonctions du caractère coopératif ou non des agents. Ceci nous conduit à la définition suivante d'une *requête* sur un réseau PFU :

Définition 5.1. *Une requête sur un réseau PFU* (V, G, P, F, U) *est une séquence Sov de paires (opérateur d'élimination-ensemble de variables)* (op, S) *telle que*

1. *tous les* S *sont disjoints ;*
2. *soit* $S \subseteq V_D$ *(variables de décision) et* $op = \min$ *ou* $\max$*, soit* $S \subseteq V_E$ *(variables d'environnement) et* $op = \oplus_u$ *;*
3. *les variables qui n'apparaissent pas dans Sov sont des variables de décision ;*
4. *pour toute paire* (x, y) *de variables de* V *de natures différentes (l'une est une variable de décision, l'autre est une variable d'environnement) telle que la composante contenant* x *est ascendante de la composante contenant* y *dans* G*,* x *n'apparaît pas à droite de* y *dans Sov (ce qui implique qu'elle apparaisse à gauche ou qu'elle soit libre).*

La dernière condition est une condition relative à la causalité qui permet d'exclure des requêtes du type $\oplus_{ubp_J, bp_M, ep_J, ep_M} \max_{mc,w}$, dans lesquelles il est implicitement supposé que les présences à la fin sont connues avant le choix du menu, ce qui n'est pas causalement cohérent.

La définition d'une requête n'impose pas que toutes les variables de V apparaissent dans Sov. Les variables qui n'y apparaissent pas sont dites *libres*, les autres *liées*. Notons que rien n'interdit que dans certains cadres $\oplus_u = \min$ ou $\oplus_u = \max$, ce qui veut dire que min ou max peuvent parfois être utilisés pour éliminer des variables de décision et des variables d'environnement.

5.2 Réponse à une requête : définition sémantique utilisant les arbres de décision

Il est possible de définir la réponse à une requête en utilisant les concepts d'*arbre de décision* et de *loterie* présentés dans l'exemple du chapitre 1. Le résultat est le suivant sous l'hypothèse que la structure de plausibilité soit conditionnable et que toutes les variables soient liées (si toutes les variables ne sont pas liées, la réponse à une requête est simplement une fonction de l'affectation des variables libres) :

Définition 5.2. *La réponse sémantique Sem-Ans*(Q) *à une requête* $Q = (Sov, (V, G, P, F, U))$ *est définie récursivement de la façon suivante, si* $\mathcal{P}_V$*,* $\mathcal{F}_V$ *et* $\mathcal{U}_V$ *sont respectivement les fonctions globales de plausibilité, de faisabilité et d'utilité associées au réseau PFU :*

$$
\begin{aligned}
&(1)\quad \textit{Sem-Ans}(Q) = \mathcal{V}_{Sr}(Sov, Pb, \emptyset)\\
&(2)\quad \mathcal{V}_{Sr}(\emptyset, Pb, A) = \mathcal{U}_V(A)\\
&(3)\quad (x \in V_D) \wedge (op = \min) \rightarrow\\
&\qquad \mathcal{V}_{Sr}((op, x).Sov, Pb, A) = \min_{a \in dom(x)} \mathcal{F}_V((x, a)|A) \star \mathcal{V}_{Sr}(Sov, Pb, A.(x, a))\\
&(4)\quad (x \in V_D) \wedge (op = \max) \rightarrow\\
&\qquad \mathcal{V}_{Sr}((op, x).Sov, Pb, A) = \max_{a \in dom(x)} \mathcal{F}_V((x, a)|A) \star \mathcal{V}_{Sr}(Sov, Pb, A.(x, a))\\
&(5)\quad (x \in V_E) \wedge (op = \oplus_u) \rightarrow\\
&\qquad \mathcal{V}_{Sr}((op, x).Sov, Pb, A) = \underset{a \in dom(x)}{\oplus_u} \mathcal{P}_V((x, a)|A) \otimes_{pu} \mathcal{V}_{Sr}(Sov, Pb, A.(x, a))
\end{aligned}
$$

Cette définition, sémantiquement justifiée par les concepts d'arbre de décision et de loterie, pose des problèmes au niveau algorithmique. En effet, le calcul de $\textit{Sem-Ans}(Q)$ nécessite l'évaluation de quantités telles que $\mathcal{F}_V((x, a)|A)$ et $\mathcal{P}_V((x, a)|A)$ qui ne sont pas présentes telles quelles dans la définition d'un réseau PFU et dont le calcul (de type calcul d'une probabilité marginale) peut être de complexité exponentielle en fonction du nombre de variables non affectées par A.

5.3 Réponse à une requête : définition opérationnelle

Il est cependant possible de considérer une autre définition de la réponse à une requête qu'on peut qualifier d'*opérationnelle* dans la mesure où elle ne présente pas les mêmes difficultés algorithmiques que la précédente. En supposant que toutes les variables sont liées (l'hypothèse d'une structure de plausibilité conditionnable n'est pas ici nécessaire) :

Définition 5.3. *La réponse opérationnelle $\textit{Op-Ans}(Q)$ à une requête Q est définie récursivement de la façon suivante :*

$$
\begin{aligned}
&(1)\quad \textit{Op-Ans}(Q) = \mathcal{V}_{Or}(Sov, Pb, \emptyset)\\
&(2)\quad \mathcal{V}_{Or}(\emptyset, Pb, A) = ((\underset{F_i \in F}{\wedge} F_i) \star ((\underset{P_i \in P}{\otimes_p} P_i) \otimes_{pu} (\underset{U_i \in U}{\otimes_u} U_i)))(A)\\
&(3)\quad \mathcal{V}_{Or}((op, x).Sov, Pb, A) = op_{a \in dom(x)} \mathcal{V}_{Or}(Sov, Pb, A.(x, a))
\end{aligned}
$$

Elle peut aussi s'écrire de la manière suivante :

$$
\textit{Op-Ans}(Q) = Sov((\underset{F_i \in F}{\wedge} F_i) \star (\underset{P_i \in P}{\otimes_p} P_i) \otimes_{pu} (\underset{U_i \in U}{\otimes_u} U_i)) \tag{5.1}
$$

c'est-à-dire qu'elle correspond exactement à l'équation 2.10.

Dans la mesure où elle fait appel uniquement à des quantités présentes dans la définition d'un réseau PFU, cette définition est algorithmiquement plus attrayante. Sa justification sémantique est cependant moins claire.

5.4 Théorème d'équivalence

Il est heureusement possible de lever ces difficultés en établissant le théorème suivant d'équivalence entre les deux définitions proposées :

Théorème 5.4. *Si la structure de plausibilité est conditionnable, les réponses sémantique et opérationnelle à une requête correcte sont identiques : Sem-Ans*(Q) = *Op-Ans*(Q)*. De plus, les règles de décision qui peuvent être obtenues dans les deux cas en mémorisant pour chaque décision la(les) valeur(s) optimisante(s) (l'*argmax *ou l'*argmin*) sont identiques.*

Ce théorème donne une justification sémantique à la définition opérationnelle algorithmiquement intéressante, ce qui ne veut pas dire qu'avec cette définition les choses seront forcément algorithmiquement faciles. Du fait de ce théorème d'équivalence, nous notons la réponse à une requête simplement par $Ans(Q)$.

Si on considère par exemple, pour le problème du trésor, la requête ayant pour séquence d'élimination $\min_{li_P} \max_{li_J} \sum_{he_J} \max_{do} \sum_{he_P} \sum_{ga,tr}$, la réponse est : 2900€ (espérance de gain de Jean). La politique associée de Pierre (faux ami de Jean) est d'écouter indifféremment à la porte A ou B. Si Pierre écoute à la porte A (respectivement B), la politique optimale de Jean est d'écouter à la porte B (respectivement A). En ce qui concerne la porte à ouvrir, Jean n'a le choix qu'entre A et B. S'il n'entend pas le gangster, sa politique optimale est d'ouvrir la porte à laquelle il a écouté. Sinon, sa politique optimale est d'ouvrir l'autre porte.

Il est également possible de définir la notion de *requête bornée*, qui correspond à un problème de décision consistant à déterminer si la réponse à une requête a une valeur supérieure à un certain seuil (et à trouver des règles de décision associées).

5.5 Requêtes classiques dans les cadres couverts

Avec un *problème de satisfaction de contraintes* (CSP) $Pb = (V, C)$, la requête classique utilise $Sov = \max_V$ (avec max = $\exists$). Il en est de même pour un problème SAT. Pour un CSP *valué*, elle utilise aussi $Sov = \min_V$, avec un min qui dépend de la structure de valuation choisie.

Avec une *formule booléenne quantifiée* (QBF) ou un CSP *quantifié* (QCSP) apparaît une libre alternance entre min = $\forall$ et max = $\exists$. Avec un *problème de satisfiabilité stochastique* (SSAT) ou un CSP *stochastique* (SCSP), on trouve aussi une libre alternance entre max pour les variables de décision et + pour les variables d'environnement.

Avec un *réseau bayésien*, le calcul d'une distribution de probabilité sur une variable y se ramène à des requêtes utilisant $Sov = \sum_{V-\{y\}}$. La recherche de l'hypothèse la plus vraisemblable sur la valeur d'une variable y amène à considérer cette variable comme une variable de décision et à considérer une requête utilisant $Sov = \max_y \cdot \sum_{V-\{y\}}$.

Avec un *diagramme d'influence* ou un *réseau de valuation* Pb, on a une alternance entre max pour les variables de décision et + pour les variables d'environnement, sachant que l'ordre d'élimination doit être cohérent avec le DAG associé à Pb.

Avec un *processus décisionnel markovien* (MDP) à horizon fini, on a une alternance, à chaque instant i, entre max pour d_i et + pour s_{i+1}. Avec un *processus partiellement observable* (POMDP), on a la même alternance entre max pour d_i et + pour o_{i+1}, mais tous les s_i (qui sont non directement

observables) sont repoussés à la fin de la séquence et éliminés avec $+$. Il est intéressant de remarquer qu'avec un MDP possibiliste pessimiste, on a une alternance entre max pour d_i et min pour s_{i+1}, alors qu'avec un MDP possibiliste optimiste, max est le seul opérateur d'élimination (aussi bien pour d_i que pour s_{i+1}), ce qui donne une structure algébrique identique à celle des VCSP possibilistes.

Avec un *problème de planification* à horizon fixé dans le cadre STRIPS, on se retrouve dans la même situation que dans le cadre CSP, du fait que $\max = \oplus_u = \vee$.

Plus formellement, il est possible d'établir le théorème suivant :

Théorème 5.5. *Les requêtes et les requêtes bornées permettent d'exprimer et de résoudre la liste suivante de problèmes :*

1. *dans le cadre des problèmes de satisfiabilité : SAT, QBF, stochastic SAT (SSAT) et extended-SSAT [62].*
2. *cadre des CSP :*
   - *tester la cohérence d'un CSP [63] ; trouver une solution à un CSP, compter le nombre de solutions d'un CSP.*
   - *Chercher une solution pour un CSP valué [94].*
   - *Résoudre un QCSP [13].*
   - *Trouver des règles de décision conditionnelles ou inconditionnelles pour un CSP mixte ou un CSP mixte probabiliste [38].*
   - *Trouver une politique optimale pour un CSP stochastique ou trouver une politique ayant une valeur supérieure à un certain seuil ; résoudre un problème d'optimisation stochastique [107].*
3. *Programmation linéaire en nombres entiers [95] avec des variables à domaines finis.*
4. *Recherche d'un plan solution de longueur $\leq k$ pour un problème de planification classique (planification à la STRIPS [40, 44]).*
5. *Résoudre des requêtes classiques sur un réseau bayésien [73], sur un champ de Markov [18] ou sur des graphes chaînés [42], avec des plausibiltés exprimées comme des probabilités, des possibilités ou des κ-rankings :*
   - *Calculer des distributions de probabilité.*
   - *Résoudre un problème MAP (Maximum A Posteriori hypothesis).*
   - *Résoudre un problème MPE (Most Probable Explanation).*
   - *Calculer la plausibilité d'une observation ou la plausibilité d'une clause logique dans un réseau hybride [29].*
6. *Résoudre un diagramme d'influence [48].*
7. *Avec un horizon fini, résoudre un MDP probabiliste, un MDP possibiliste, un MDP basé sur les κ-rankings, complètement ou partiellement observable (POMDP), factorisé ou non [86, 68, 91, 16, 15].*

5.6 Extensions à d'autres classes de requêtes

Les requêtes pourraient être rendues plus complexes en relâchant certaines hypothèses :

— Telle quelle, la définition des requêtes s'applique si l'ensemble des degrés d'utilité E_u est totalement ordonné par $\preceq_u$. Etendre les résultats à des ordres *partiels* est quasi-immédiat lorsque $(E_u, \preceq_u)$ est un *treillis*, ce qui permet au cadre PFU de subsumer les *semiring-based CSP* [9, 10]. D'autres extensions à des ordres partiels devraient pouvoir permettre d'englober les MDP algébriques [74].

— L'hypothèse de *no-forgetting* pourrait également être relâchée, dans l'esprit des diagrammes d'influence à mémoire limitée (*limited memory influence diagrams*, LIMIDs [61]), dans lesquels on cherche des règles de décision fonctions uniquement de certaines variables, et non de toutes les variables présentes dans l'historique des observations.

— L'ordre dans lequel les décisions sont prises et les observations sont réalisées a été supposé connu. Dans certaines situations, il peut être utile non seulement de calculer des règles de décisions optimales, mais aussi de déterminer un ordre optimal dans lequel prendre des décisions et faire des observations. Dans cette voie, les travaux sur les diagrammes d'influence avec décisions non totalement ordonnées [52] pourraient être considérés.

— Enfin, relâcher l'hypothèse de variables à domaines finis n'est pas direct car transformer un + en intégrale ou faire des opérations d'optimisation sur des domaines continus requiert quelques propriétés supplémentaires. Dans cette direction, tous les travaux concernant les réseaux temporels simples (*Simple Temporal Problems*, STPs [32]) and leurs extensions [53, 103, 105, 90] peuvent être considérés. Dans ces problèmes, les variables sont des points dans le temps à valeurs dans des intervalles continus et des contraintes de durée entre certains de ces points sont définies. Cependant, dans ces formalismes, la gestion de l'incertain s'applique uniquement à des non déterminismes booléens, qui spécifient quelles sont les évolutions possibles et impossibles de l'environnement.

5.7 Résumé

Ce chapitre a introduit le dernier élément clé du cadre PFU : une classe de requêtes sur des réseaux PFU. La réponse à une requête a été définie dans un premier temps à partir d'arbres de décision. Le premier résultat important de ce chapitre est le théorème 5.4, qui donne un fondement théorique à une seconde définition de la réponse à une requête, équivalente à la première. Cette seconde définition, qui identifie la réponse à une requête à une séquence d'éliminations de variables sur une combinaison de fonctions locales, est mieux adaptée aux besoins des algorithmes futurs étant donnée qu'elle utilise uniquement les fonctions locales définies par un réseau PFU. Le second résultat important est le théorème 5.5 qui montre que nombre de requêtes usuelles peuvent être exprimées comme des requêtes PFU.

Au final, le cadre PFU est entièrement spécifié par les définitions 3.4, 3.5, 3.6 pour la structure algébrique, par la définition 4.5 pour les réseaux PFU et par la définition 5.1 et l'équation 5.1 pour les requêtes.

5.8 Conclusion de la partie I : gains et coûts du cadre PFU

Une meilleure compréhension Le théorème 5.5 montre que de nombreux cadres existants sont des instances du cadre PFU. A travers cette unification, les similitudes et les différences entre

les formalismes existants peuvent apparaître plus clairement. Par exemple, une comparaison des VCSP et de la version optimiste des MDP possibilistes, à travers la définition opérationnelle de la réponse à une requête révèle que d'un point de vue purement algébrique, un MDP possibiliste optimiste (complètement ou partiellement observable) n'est qu'une forme spécifique de CSP flou. Il s'ensuit que les librairies disponibles pour résoudre des VCSP peuvent être directement utilisées pour résoudre de tels MDP.

Du point de vue théorique, une étude de la complexité temporelle et spatiale du calcul de l'équation 2.10 (page 30) peut permettre d'obtenir des bornes de complexité à moindre coût sur plusieurs formalismes simultanément. On peut également chercher à caractériser des propriétés induisant une complexité théorique donnée.

Expressivité accrue Le pouvoir d'expression accru des réseaux PFU est le résultat d'une flexibilité présente à plusieurs niveaux : (1) flexibilité du modèle de plausibilité/utilité ; (2) flexibilité en termes de réseaux possibles ; (3) flexibilité des requêtes considérées en termes de scénarios modélisables. Cette expressivité permet aux réseaux PFU de couvrir des formes génériques de problèmes de décision séquentielle avec plausibilités, faisabilités et utilités, avec des agents coopératifs ou antagonistes, des observabilités partielles et des paramètres dans le processus décisionnel via la présence de variables libres.

Aucun des cadres indiqués dans le théorème 5.5 (le théorème de subsumption) ne présente une telle flexibilité. Plus spécifiquement, comparé aux diagrammes d'influence [48, 52, 101, 71, 51] ou aux réseaux de valuations (VN [98, 100, 34]), le cadre PFU peut manipuler autre chose que de l'utilité espérée additive probabiliste, il permet de faire des éliminations avec l'opérateur min pour modéliser la présence d'agents antagonistes. Ainsi, les formules booléennes quantifiées ne peuvent pas être représentées par l'intermédiaire d'un diagramme d'influence ou d'un réseau de valuation, mais elles peuvent l'être par une requête PFU. En outre, les réseaux PFU font intervenir un DAG capturant certaines conditions de normalisation concernant les plausibilités et les faisabilités, alors que les réseaux de valuation n'encapsulent pas cette information. Comparé aux diagrammes d'influence ou aux réseaux de valuation dits séquentiels (*sequential influence diagrams* [51], *sequential valuation networks* [34]), le cadre PFU est aussi capable d'exprimer des problèmes de décision dits *assymétriques*, dans lesquels une variable de décision peut ne pas avoir à être considérée dans le processus décisionnel, via l'ajout de valeurs supplémentaires dans le domaine de certaines variables.

En fait, certains problèmes simples qui peuvent être exprimés comme des requêtes PFU ne pouvent pas être définis directement dans d'autres formalismes. Parmi ces problèmes, on citer des problèmes faisant intervenir "faisabilités *avec conditions de normalisation* + exigences dures" (l'utilisation d'un CSP pour modéliser un tel problème fait perdre l'information fournie par les conditions de normalisation, à moins d'utiliser des variables supplémentaires permettant définir des contraintes sur les contraintes). De même pour les problèmes de "décision séquentielle du type diagramme d'influence, raisonnant à partir d'utilités espérées possibilistes", qui pourraient être baptisés diagrammes d'influence possibilistes. [1] De même pour l'instance "CSPs stochastiques sans hypothèse de contingence", pour l'instance "max-QBF" (analogue au problème max-SAT),

1. Les diagrammes d'influence possibilistes ont été proposés très récemment dans un travail effectué parallèlement à cette thèse [43]. Ce formalisme est une instanciation simple du cadre PFU.

ou pour l'instance "VCSP quantifiés", qui pourrait correspondre à des VCSP faisant intervenir une alternance de minimisations et de maximisations modélisant la présence de décideurs antagonistes. Ainsi, le cadre PFU couvre de nouveaux formalismes.

Le coût de la flexibilité et de l'expressivité accrue est que le cadre PFU ne peut être décrit aussi succintement que par exemple le cadre des CSP.

Algorithmes génériques La partie II va montrer qu'il est possible de définir des algorithmes génériques pour répondre à une requête sur un réseau PFU. Comme indiqué précédemment, construire des algorithmes génériques peut générer des fertilisations croisées, dans le sens où cela peut permettre à chacun des formalismes couverts de bénéficier de techniques développées dans un autre formalisme couvert. Cette démarche est en adéquation avec les efforts réalisés au cours de ces dernières années pour généraliser des méthodes de résolution utilisées parallèlement pour résoudre des problèmes différents. Par exemple, la propagation de contraintes souples est un outil qui améliore grandement la résolution des VCSP ; intégrer un tel outil dans un solver générique défini dans le cadre PFU permettrait de l'utiliser directement pour résoudre des diagrammes d'influence. L'utilisation d'opérateurs algébriques abstraits peut également permettre d'identifier des propriétés algorithmiquement intéressantes, ou d'induire des conditions nécessaires ou suffisantes pour qu'un algorithme particulier soit utilisable.

Cependant, certaines techniques peuvent être fortement liées à un formalisme spécifique ou à un type de problème, et dans ce cas un algorithme dédié peut être notoirement plus efficace qu'un algorithme générique. Une solution minimisant la portée de cette objection est simplement de dire que le cadre PFU est une opportunité pour généraliser de tels algorithmes dédiés, en caractérisant clairement les propriétés algébriques qui le rendent si efficace. De plus, même si des algorithmes spécialisés sont généralement meilleurs que des algorithmes génériques, il existe des cas dans lesquels des outils génériques s'avèrent plus performants que des algorithmes spécialisés. Voir par exemple [93] ou l'utilisation de solvers SAT pour résoudre des problèmes de planification à la STRIPS.

Deuxième partie

Algorithmes génériques pour répondre à des requêtes sur des réseaux PFU

Chapitre 6

Premiers algorithmes génériques

Le cadre PFU est un cadre flexible qui unifie plusieurs formalismes existants. On pourrait penser qu'un écueil inhérent à cette généricité est que répondre à une requête sur un réseau PFU est hors de portée, du moins si on souhaite le faire de manière efficace. Un des objectifs des chapitres à venir est de montrer qu'il n'en est rien et que la possibilité des répondre efficacement ou non à une requête est une conséquence de la requête considérée elle-même, et non de la généricité.

Initialement, le cadre PFU a été construit non seulement pour ses aptitudes en termes de représentation de la connaissance, mais aussi pour permettre de définir des algorithmes génériques capables de répondre à des requêtes. Certains choix ont même été justifiés par des considérations algorithmiques. En d'autres termes, l'objectif est de pouvoir répondre à des requêtes aussi efficacement que possible, et non uniquement de les exprimer. Dans ce qui suit, nous introduisons des schémas de résolution génériques qui sont soit des généralisations d'algorithmes déjà existants, soit des techniques nouvelles applicables directement à tous les formalismes couverts par le cadre PFU. Ce chapitre présente deux premières approches permettant de répondre à des requêtes PFU dans le cas général, c'est-à-dire sans hypothèses algébriques supplémentaires. Ces approches sont toutes deux developpées à partir de la définition opérationnelle de la réponse à une requête, définie par $Ans(Q) = Sov((\wedge_{F_i \in F} F_i) \star (\otimes_{p P_i \in P} P_i) \otimes_{pu} (\otimes_{u U_i \in U} U_i))$. Plus précisément, nous introduisons :

- un algorithme générique de recherche arborescente très basique ;
- un algorithme générique utilisant des mécanismes d'élimination de variables [6], afin d'exploiter au mieux la factorisation en fonction locales.

Des résultats de complexité théorique fonctions d'un paramètre appelé largeur induite contrainte sont également fournis.

6.1 Un algorithme naïf de recherche arborescente

Sur la base du cadre proposé et de la définition opérationnelle de la réponse à une requête, il est possible de définir des algorithmes génériques capables de répondre à n'importe quelle requête sur n'importe quel réseau PFU.

Un premier algorithme possible est un algorithme de type *recherche arborescente* qui utilise

comme ordre d'affectation des variables un ordre compatible avec la séquence d'élimination Sov utilisée dans la requête : si deux variables x et y appartiennent à deux paires (opérateur d'élimination-ensemble de variables) différentes pov_x et pov_y et si pov_x est située avant pov_y (à gauche) dans Sov, x est affectée avant y ; si elles appartiennent à une même paire, leur ordre d'affectation est indifférent (du point de vue du résultat, pas forcément du point de vue de l'efficacité algorithmique).

Cet algorithme collecte au niveau de chaque feuille f la combinaison des plausibilités, des faisabilités et des utilités pour l'affectation complète associée à f. Il synthétise au niveau de chaque nœud non feuille n les valeurs fournies par les nœuds fils de n en utilisant l'opérateur d'élimination associé à n. La figure 6.1 montre le pseudo-code d'un tel algorithme (avec toujours l'hypothèse que toutes les variables sont liées). La réponse à la requête Sov sur un PFU Pb est fournie par l'appel **RechArbPFU**$(Sov, Pb, \emptyset)$.

```
RechArbPFU(Sov, (V, G, P, F, U), A)
début
  si Sov = ∅ alors
    retourner ((∧_{F_i∈F} F_i) ⋆ (⊗_{p P_i∈P} P_i) ⊗_{pu} (⊗_{u U_i∈U} U_i))(A)
  sinon
    (op, S).Sov' ← Sov
    choisir x ∈ S
    si S = {x} alors Sov ← Sov' sinon Sov ← (op, S − {x}).Sov'
    d ← dom(x)
    res ← ◇
    tant que d ≠ ∅ faire
      choisir a ∈ d
      d ← d − {a}
      res ← op(res, RechArbPFU(Sov, Pb, A.(x, a)))
    retourner res
fin
```

Figure 6.1: Un algorithme générique de type recherche arborescente.

Cet algorithme naïf a une complexité *temporelle exponentielle* en fonction du nombre de variables, mais une complexité *spatiale polynomiale*.

Ceci nous fournit au moins une information intéressante sur la *classe de complexité* du problème de réponse à une requête PFU, problème noté AnswerPFU : sachant que le problème QBF (satisfiabilité d'une formule booléenne quantifiée) est *PSPACE-complet* et est un sous-problème du problème AnswerPFU, ce dernier est au moins *PSPACE-dur* ; mais sachant qu'il est *PSPACE* (existence d'un algorithme de complexité spatiale polynomiale), il est forcément *PSPACE-complet* [1].

6.2 Elimination de variables : une première version naïve

Un autre algorithme possible est un algorithme de type *élimination de variables* [6, 97, 25, 56] utilisant comme ordre d'élimination un ordre inverse de celui utilisé par la recherche arborescente : si deux variables x et y appartiennent à deux paires (opérateur d'élimination-ensemble de variables)

1. En toute rigueur, le problème AnswerPFU n'est pas un problème de décision et il faudrait parler du problème de décision associé, par exemple, le problème consistant à déterminer si la réponse à une requête est supérieure ou égale à une valeur fixée.

différentes pov_x et pov_y et si pov_x est située avant pov_y (à gauche) dans Sov, y est éliminée avant x ; si elles appartiennent à une même paire, leur ordre d'élimination est indifférent.

Cet algorithme commence par combiner toutes les fonctions locales de plausibilité, de faisabilité et d'utilité en une seule fonction globale d'utilité. Il en élimine ensuite les variables les unes après les autres en utilisant à chaque fois l'opérateur d'élimination associé. La figure 6.2 montre le pseudo-code d'un tel algorithme.

ElimVarPFU(Sov, Pb)
début
  $L_0 \leftarrow ((\wedge_{F_i \in F} F_i) \star (\otimes_{p P_i \in P} P_i) \otimes_{pu} (\otimes_{u U_i \in U} U_i))$
  tant que $Sov \neq \emptyset$ **faire**
    $Sov'.(op, S) \leftarrow Sov$
    choisir $x \in S$
    si $S = \{x\}$ **alors** $Sov \leftarrow Sov'$ **sinon** $Sov \leftarrow Sov'.(op, S - \{x\})$
    $L_0 \leftarrow op_x\, L_0$
  retourner L_0
fin

Figure 6.2: Un algorithme générique naïf de type élimination de variables.

Cet algorithme naïf est pire que le précédent puisqu'il a une complexité *temporelle* et *spatiale exponentielle*. De la même façon que le premier algorithme de recherche arborescente, cet algorithme n'exploite pas la *factorisation* en fonctions locales de plausibilité, de faisabilité et d'utilité. Sans propriétés algébriques supplémentaires, il est en fait impossible d'utiliser des algorithmes d'élimination classiques, comme par exemple *bucket elimination* [25], qui exploitent la structure de l'hypergraphe des fonctions locales.

La raison principale est qu'aucun axiome n'est imposé concernant les relations entre l'opérateur $\otimes_u$ et les autres opérateurs. Plus precisément, il est par exemple impossible dans le cas général de calculer la quantité $\underset{x}{\oplus_u} P_x \otimes_{pu} (U_x \otimes_u U_y)$ en ne considérant que les fonctions locales qui dépendent de x.

Nous avons retenu deux ensembles de propriétés supplémentaires, parce qu'ils sont l'un ou l'autre satisfaits par chacun des cadres classiques et parce qu'ils permettent l'un et l'autre l'utilisation de ces algorithmes d'élimination classiques. Dans ce qui suit, les opérateurs de combinaison $\otimes$ sur E sont étendus à $E \cup \{\Diamond\}$ par $e \otimes \Diamond = \Diamond \otimes e = \Diamond$.

6.3 Deux conditions suffisantes de décomposabilité

Les deux axiomes retenues permettant de n'avoir à considérer que les fonctions locales ayant x dans leur portée lorsqu'une variable x est éliminée sont notés Ax^{SG} et Ax^{SA} et s'écrivent de la manière suivante :

$$Ax^{SA} : \begin{cases} \otimes_u \text{ distributif par rapport à } \oplus_u \\ \text{et } p \otimes_{pu} (u_1 \otimes_u u_2) = (p \otimes_{pu} u_1) \otimes_u u_2 \text{ pour tout } (p, u_1, u_2) \in E_p \times E_u \times E_u \end{cases}$$

Ax^{SG} : "$\otimes_u = \oplus_u$ sur E_u" (mais pas sur $E_u \cup \{\Diamond\}$)

La première condition suffisante de décomposabilité est notée Ax^{SA} comme "axiome du cas

semi-anneau", car il confère à $(E_u, \oplus_u, \otimes_u)$ une structure de semi-anneau. La seconde condition suffisante de décomposabilité est notée Ax^{SG} comme "axiome pour le cas semi-groupe', car il rend la structure $(E_u, \oplus_u, \otimes_u)$ en quelque sorte équivalente à la structure $(E_u, \oplus_u)$, qui est un semi-groupe. Le tableau 6.1 montre que ces deux axiomes disjoints couvrent des structures classiques d'utilité espérée.

	E_p	E_u	$\otimes_u$	$\oplus_u$	$\otimes_{pu}$	Axiome satisfait
1	$\mathbb{R}^+$	$\mathbb{R} \cup \{-\infty\}$	$+$	$+$	$\times$	SG
2	$\mathbb{R}^+$	$\mathbb{R}^+$	$\times$	$+$	$\times$	SA
3	$[0,1]$	$[0,1]$	min	max	min	SA
4	$[0,1]$	$[0,1]$	min	min	$\max(1-p, u)$	SG
5	$\mathbb{N} \cup \{\infty\}$	$\mathbb{N} \cup \{\infty\}$	$+$	min	$+$	SA
6	$\{t, f\}$	$\{t, f\}$	$\wedge$	$\vee$	$\wedge$	SA
7	$\{t, f\}$	$\{t, f\}$	$\wedge$	$\wedge$	$\rightarrow$	SG
8	$\{t, f\}$	$\{t, f\}$	$\vee$	$\vee$	$\wedge$	SG
9	$\{t, f\}$	$\{t, f\}$	$\vee$	$\wedge$	$\rightarrow$	SA

TABLE 6.1 – Structures d'utilité espérées satisfaisant Ax^{SA} ou Ax^{SG} : (1) utilité espérée probabiliste avec utilités additives (permet de calculer l'espérance d'un gain ou d'un coût), (2) utilité espérée probabiliste avec utilités multiplicatives (permet de calculer la probabilité que des contraintes soient satisfaites), (3) utilité espérée possibiliste optimiste, (4) utilité espérée possibiliste pessimiste, (5) utilité qualitative avec kappa-rankings et utilités uniquement positives, (6) utilité espérée booléenne optimiste avec utilités conjonctives, (7) utilité espérée booléenne pessimiste avec utilités conjonctives, (8) utilité espérée booléenne optimiste avec utilités disjonctives, (9) utilité espérée booléenne pessimiste avec utilités disjonctives.

La proposition 6.1 montre que dès que l'un des deux axiomes est satisfait, l'élimination d'une variable peut se faire en ne considérant que les fonctions locales qui dépendent de cette variable. Etant donné un ensemble de fonctions locales Φ et une variable x, nous notons Φ^{+x} l'ensemble des fonctions ayant x dans leur portée ($\Phi^{+x} = \{\varphi \in \Phi \,|\, x \in (\varphi)\}$) et $\Phi^{-x} = \Phi - \Phi^{+x}$. De plus, étant donnés des ensembles P, F et U de fonctions locales de plausibilité, de faisabilité et d'utilité respectivement, nous notons $(\wedge_{F_i \in F} F_i) \star (\otimes_{p\, P_i \in P} P_i) \otimes_{pu} (\otimes_{u U_i \in U} U_i)$ sous la forme $F \star P \otimes_{pu} U$, c'est-à-dire que nous considérons la combinaison de fonctions du même type comme implicite.

Proposition 6.1. *Soit $(E_p, E_u, \oplus_u, \otimes_{pu})$ une structure d'utilité espérée. Soit P et U des ensembles de fonctions locales de plausibilité et d'utilité respectivement.*

Si l'axiome Ax^{SA} est verifié, alors

$$\underset{x}{\oplus_u} (P^{+x} \otimes_{pu} U) \quad = \quad U^{-x} \otimes_u (\underset{x}{\oplus_u} (P^{+x} \otimes_{pu} U^{+x})) \tag{6.1}$$

Si l'axiome Ax^{SG} est vérifié, alors

$$\underset{x}{\oplus_u} (P^{+x} \otimes_{pu} U) \quad = \quad ((\underset{x}{\oplus_p} P^{+x}) \otimes_{pu} U^{-x}) \otimes_u (\underset{x}{\oplus_u} (P^{+x} \otimes_{pu} U^{+x})) \tag{6.2}$$

Cette proposition peut être illustrée par les structures utilisées pour la satisfaction espérée probabiliste et pour l'utilité additive espérée probabiliste. Dans le premier cas, qui satisfait Ax^{SA}, il est possible d'écrire $\sum_x (P^{+x} \times U) = U^{-x} \times (\sum_x (P^{+x} \times U^{+x}))$, alors que dans le second cas, nous pouvons écrire $\sum_x (P^{+x} \times (U^{-x} + U^{+x})) = ((\sum_x P^{+x}) \times U^{-x}) + (\sum_x (P^{+x} \times U^{+x}))$.

6.4 Algorithme d'élimination de variables amélioré

Comme nous allons le voir, les axiomes Ax^{SA} et Ax^{SG} permettent de calculer la réponse à une requête en utilisant un véritable algorithme d'élimination de variables, c'est-à-dire un algorithme permettant d'éliminer une variable x en ne considérant que les fonctions locales dépendantes de x.

6.4.1 Algorithme amélioré dans le cas semi-anneau

Lorsque l'axiome Ax^{SA} est satisfait, il est possible de se ramener à une structure algébrique beaucoup plus simple : il est possible de montrer que l'on peut transformer l'espace des degrés de plausibilité E_p en l'espace des degrés d'utilité E_u via un morphisme $\phi : p \to p \otimes_{pu} 1_u$. Grâce à ce morphisme, il est possible de montrer qu'il suffit de travailler sur un seul espace E égal à E_u faisant intervenir un seul opérateur de combinaison $\otimes$ égal à $\otimes_u$ et un seul opérateur d'élimination $\oplus$ égal à $\oplus_u$. En d'autres termes, l'axiome Ax^{SA} est équivalent à l'axiome suivant :

$$Ax^{SA'} : \begin{cases} (E_p, \preceq_p) = (E_u, \preceq_u) = (E, \preceq) \\ \otimes_p = \otimes_{pu} = \otimes_u = \otimes \\ \oplus_p = \oplus_u = \oplus \end{cases}$$

La structure algébrique du cadre PFU devient alors beaucoup plus simple.

Définition 6.2. *$(E, \oplus, \otimes)$ is un semi-anneau commutatif monotone totalement ordonné (*totally ordered Monotonic Commutative Semiring (MCS)*) si et seulement si $(E, \oplus, \otimes)$ est un semi-anneau commutatif muni d'un ordre total tel que $\oplus$ et $\otimes$ sont monotones pour cet ordre total.*

Proposition 6.3. *$(E_p, E_u, \oplus_u, \otimes_{pu})$ est une structure d'utilité espérée totalement ordonnée qui satisfait $Ax^{SA'}$ (les structures de plausibilité et d'utilité espérée sous-jacente étant $(E_p, \oplus_p, \otimes_p)$ et $(E_u, \otimes_u)$ respectivement) si et seulement si $(E_u, \oplus_u, \otimes_u)$ est un MCS totalement ordonné.*

Ainsi, lorsque $Ax^{SA'}$ est satisfait, la structure algébrique du cadre PFU est tout simplement un MCS totalement ordonné $(E, \oplus, \otimes) = (E_u, \oplus_u, \otimes_u)$. La condition de normalisation sur les composantes d'environnement devient

$$\bigoplus_c \Big(\bigotimes_{P_i \in Fact(c)} P_i \Big) = 1_E$$

et la réponse opérationnelle à une requête prend la forme

$$Ans(Q) = Sov\Big(\Big(\bigwedge_{F_i \in F} F_i \Big) \star \Big(\bigotimes_{\varphi \in P \cup U} \varphi \Big) \Big) \tag{6.3}$$

En outre, au lieu d'exprimer les faisabilités sur $\{t, f\}$, nous pouvons les exprimer sur $\{1_E, \lozenge\}$ en associant la valeur 1_E à t et la valeur $\lozenge$ à f. Cet artifice technique préserve la valeur d'une requête car $t \star u = 1_E \otimes u$ et $f \star u = \lozenge \otimes u$. La réponse à une requête Q devient $Ans(Q) = Sov(\otimes_{\varphi \in P \cup F \cup U} \varphi)$. Ainsi, le cas semi-anneau peut nécessiter plusieurs opérateurs d'élimination (min, max et $\oplus$) mais ne requiert en réalité qu'un seul opérateur de combinaison ($\otimes$).

Proposition 6.4. *Soit $(E, \oplus, \otimes)$ un MCS totalement ordonné. Nous étendons $\oplus$ et $\otimes$ sur $E \cup \{\lozenge\}$ par $u \oplus \lozenge = \lozenge \oplus u = u$ et $u \otimes \lozenge = \lozenge \otimes u = \lozenge$. Alors, pour tout $op \in \{\min, \max, \oplus\}$, $(E \cup \{\lozenge\}, op, \otimes)$ est un semi-anneau commutatif.*

La propriété précédente assure notamment une distributivité de l'opérateur de combinaison $\otimes$ sur tous les opérateurs d'élimination utilisés. Ce résultat est essentiel pour pouvoir utiliser l'algorithme d'élimination de variables donné à la figure 6.3, qui exploite la factorisation en fonctions locales. Le premier appel est **VE-answerQ**$(Sov, \otimes, P \cup F \cup U)$.

VE-answerQ$(Sov, \circledast, \Phi)$
début
  si $Sov = \emptyset$ **alors retourner** Φ
  sinon
    $Sov'.(op, S) \leftarrow Sov$
    choisir $x \in S$
    si $S = \{x\}$ **alors** $Sov \leftarrow Sov'$ **sinon** $Sov \leftarrow Sov'.(op, S - \{x\})$
    $\varphi_0 \leftarrow op_x \left(\circledast_{\varphi \in \Phi^{+x}} \varphi\right)$
    $\Phi \leftarrow (\Phi - \Phi^{+x}) \cup \{\varphi_0\}$
    retourner *VE-answerQ*$(Sov, \circledast, \Phi)$
fin

Figure 6.3: Un algorithme d'élimination de variables générique utilisant les factorisations disponibles (Sov : séquence d'éliminations, $\circledast$: opérateur de combinaison, Φ : ensemble de fonctions locales).

6.4.2 Algorithme amélioré dans le cas semi-groupe

Le cas semi-groupe nécessite un peu plus de travail que le cas semi-anneau. La raison est que l'équation 6.2 ne crée pas uniquement une nouvelle fonction d'utilité résultant de l'élimination de x : elle crée d'une part une nouvelle fonction de plausibilité $\underset{x}{\oplus_p} P^{+x}$ qui doit être combinée avec toutes les fonctions d'utilité de U^{-x}, et d'autre part une nouvelle fonction d'utilité $\underset{x}{\oplus_u} (P^{+x} \otimes_{pu} U^{+x})$. Autrement dit, la quantité obtenue après élimination de x n'est pas directement de la forme $Sov'(F' \star P' \otimes_{pu} U')$, avec Sov' la séquence d'élimination restant à traiter et F', P' et U' des nouveaux ensembles de fonctions locales.

Afin d'exploiter une forme générale qui ne varie pas durant les différentes étapes d'éliminations, une solution consiste à travailler sur des couples (fonction de plausibilité,fonction d'utilité) appelés des *potentiels* [70] (ces potentiels n'ont rien à voir avec les potentiels utilisés dans un champ de Markov).

Définition 6.5. *Un* potentiel *est une paire (P_0, U_0) composée d'une fonction de plausibilité P_0 et d'une fonction d'utilité U_0. Deux opérateurs sont définis sur les paires plausibilité-utilité :*

- *un opérateur de combinaison $\boxtimes$ défini par $(p_1, u_1) \boxtimes (p_2, u_2) = (p_1 \otimes_p p_2, (p_1 \otimes_{pu} u_2) \otimes_u (p_2 \otimes_{pu} u_1))$,*
- *un opérateur d'élimination $\boxplus$ défini par $(p_1, u_1) \boxplus (p_2, u_2) = (p_1 \oplus_p p_2, u_1 \oplus_u u_2)$.*

Enfin, un ordre partiel est défini sur les paires plausibilité-utilité par $(p, u_1) \preceq (p, u_2)$ ssi $u_1 \preceq u_2$.

Dans ce qui suit, nous considérons également chaque fonction de faisabilité comme un potentiel. Comme seul un ordre partiel est défini sur les potentiels, certaines étapes techniques sont requises pour assurer que des opérations de minimisation ou de maximisation soient bien définies. Ces étapes techniques impliquent de travailler sur des réseaux PFU dits raffinés et nécessitent une légère modification de la définition de Φ^{+x} pour un ensemble de fonctions Φ. Ces étapes permettent progressivement d'arriver au résultat suivant :

Proposition 6.6. *Soit $Q = (Sov, \mathcal{N})$ une requête. Soit $T(Sov)$ la séquence de paires opérateur-variables obtenus à partir de Sov en remplaçant les $\oplus_u$ par des $\boxplus$.*

VE-answerQ$(T(Sov), \boxtimes, \{(P_i, 1_u), P_i \in P\} \cup F \cup \{(1_p, U_i), U_i \in U\})$ *renvoie un ensemble de potentiels* Π *tel que* $\boxtimes_{\varphi\in\Pi}\varphi(A) = \begin{cases} (1_p, Ans(Q)(A)) \text{ si } Ans(Q)(A) \neq \Diamond \\ \Diamond \text{ sinon} \end{cases}$

Notons que comme dans le cas semi-anneau, avec le passage de Ax^{SA} à $Ax^{SA'}$, il est possible de transformer l'axiome Ax^{SG} en un axiome $Ax^{SG'}$ plus simple permettant d'utiliser seulement deux opérateurs paramétrables ($\oplus$ et $\otimes$) pour définir la réponse à une requête. Mais cette fois, la transformation n'est pas gratuite, c'est-à-dire qu'elle nécessite certaines conditions telles que la possibilité de translater l'échelle des utilités pour obtenir une structure non bipolaire, c'est-à-dire avec des degrés d'utilité soit tous inférieurs à 0_u soit tous supérieurs à 0_u. Avec l'axiome $Ax^{SG'}$, la réponse à une requête s'écrit :

$$Ans(Q) = Sov((\underset{F_i\in F}{\wedge} F_i) \star (\underset{P_i\in P}{\otimes} P_i) \otimes (\underset{U_i\in U}{\oplus} U_i)) \tag{6.4}$$

avec $(E, \oplus, \otimes)$ un MCS totalement ordonné.

6.4.3 Cas général

Les cas semi-anneau et semi-groupe définissent deux conditions *suffisantes* permettant d'utiliser la factorisation en fonctions locales. Montrer à quel point ces conditions sont nécesssaires est un problème ouvert. Lorsque ni Ax^{SA} ni Ax^{SG} n'est satisfait, on peut se ramener à un calcul du type :

$$Ans(Q) = Sov((\underset{F_i\in F}{\wedge} F_i) \star (\underset{P_i\in P}{\otimes} P_i) \otimes U_0) = Sov(\underset{\varphi\in P\cup F\cup\{U_0\}}{\otimes} \varphi) \tag{6.5}$$

Ainsi, le cas général est un cas particulier du cas semi-anneau à condition d'agrèger toutes les fonctions locales d'utilité pour obtenir une unique fonction globale d'utilité. L'algorithme **VE-answerQ** peut à nouveau être utilisé, avec **VE-answerQ**$(Sov, \otimes, P \cup F \cup \{U_0\})$ comme premier appel.

Le tableau 6.2 résume l'utilisation de l'algorithme **VE-answerQ** pour répondre à une requête PFU. Notons que pour chaque cas, *aucune hypothèse algébrique supplémentaire n'est nécessaire.*

CAS	PREMIER APPEL
semi-anneau (Ax^{SA})	**VE-answerQ**$(Sov, \otimes, P \cup F \cup U)$
semi-groupe (Ax^{SG})	**VE-answerQ**$(T(Sov), \boxtimes, \{(P_i, 1_u), P_i \in P\} \cup F \cup \{(1_p, U_i), U_i \in U\})$
cas général	**VE-answerQ**$(Sov, \otimes, P \cup F \cup \{U_0\})$, with $U_0 = \otimes_{uU_i\in U} U_i$

TABLE 6.2 – Utilisation de l'algorithme **VE-answerQ**.

6.5 Evaluation de la complexité théorique

Cette section fournit des bornes supérieures sur la complexité temporelle et spatiale de l'algorithme **VE-answerQ**. Les résultats de complexité sont exprimés en fonction d'un paramètre connu sous le nom de largeur induite contrainte [50, 72]. Les bornes données sont valables pour tous les formalismes couverts par le cadre PFU.

6.5.1 Largeur induite

La largeur induite [28, 27] est un paramètre définissant une borne supérieure sur la complexité théorique des algorithmes classiques d'élimination de variables. Elle est aussi connue sous le nom de largeur d'arbre [88] ou de *k-tree number* [2]. Etant donnée une requête mono-opérateur sur un modèle graphique (V, Φ), la largeur induite est définie à partir de l'hypergraphe $\mathcal{G} = (V, \{sc(\varphi) \mid \varphi \in \Phi\})$ associé à ce modèle graphique.

Définition 6.7. *Un ordre d'élimination o sur un ensemble de variables $V = \{x_1, \ldots, x_n\}$ est une bijection de $\{1, \ldots, n\}$ dans V. Pour tout $k \in \{1, \ldots, n\}$, $o(k)$ est appelé la k-ème variable éliminée dans o.*

Un ordre d'élimination o induit un ordre total $\preceq$ sur V, défini par $o(n) \prec \ldots \prec o(2) \prec o(1)$, $x \prec y$ signifiant que y doit être éliminée avant x. Ceci nous permet de considérer de manière abusive que o est un ordre total sur V.

Définition 6.8. *(Largeur induite d'un ordre d'élimination) Soit $\mathcal{G} = (V_\mathcal{G}, H_\mathcal{G})$ un hypergraphe. Soit o un ordre d'élimination sur $V_\mathcal{G}$. o peut être utilisé pour générer une séquence d'hypergraphes $\mathcal{G}_1, \ldots, \mathcal{G}_{n+1}$ (avec $n = |V_\mathcal{G}|$), définie par*

- *$\mathcal{G}_1 = \mathcal{G}$*
- *si $\mathcal{G}_k = (V_k, H_k)$ et si x est la k-ème variable éliminée dans o, alors $\mathcal{G}_{k+1} = (V_k - \{x\}, (H_k - H_k^{+x}) \cup \{h_{k+1}\})$, avec H_k^{+x} l'ensemble des hyper-arêtes de H_k impliquant la variable x et $h_{k+1} = (\cup_{h \in H_k^{+x}} h) - \{x\}$ l'hyper-arête créée de l'étape k à l'étape $k+1$ (étape d'élimination de variable).*

La largeur induite de $\mathcal{G}$ pour l'ordre d'élimination o, notée $w_\mathcal{G}(o)$ est égale à la taille maximale des hyper-arêtes créées, c'est-à-dire $w_\mathcal{G}(o) = \max_{k \in \{1,\ldots,n\}} |h_{k+1}|$.

Informellement, l'hyper-arête h_{k+1} créée de l'étape k à l'étape $k+1$ est obtenue en considérant l'ensemble H_k^{+x} des hyper-arêtes de $\mathcal{G}_k$ qui "dépendent" de x et en "reliant" toutes les variables impliquées dans H_k^{+x} (sauf x). Ceci signifie que l'élimination de x crée une nouvelle fonction locale de portée h_{k+1}. Le nombre $1 + w_\mathcal{G}(o)$ correspond au nombre maximal de variables à considérer simultanément pendant les étapes d'élimination de variables.

Un résultat connu est que les complexités temporelles et spatiales d'un algorithme classique d'élimination de variables sont exponentielles en la largeur induite de l'ordre d'élimination utilisé.

Définition 6.9. *(Largeur d'un hypergraphe $\mathcal{G}$) Soit $\mathcal{G} = (V_\mathcal{G}, H_\mathcal{G})$ un hypergraphe. La largeur de $\mathcal{G}$, notée $w_\mathcal{G}$, est égale à la largeur minimale induite par un ordre d'élimination sur $V_\mathcal{G}$: si $\mathcal{O}$ représente l'ensemble de tous les ordres d'élimination possibles sur $V_\mathcal{G}$, alors $w_\mathcal{G} = \min_{o \in \mathcal{O}} w_\mathcal{G}(o)$.*

La largeur induite d'un hypergraphe est égale au nombre minimal de variables à considérer simultanément par un algorithme d'élimination de variables lorsqu'un ordre d'élimination optimal

est utilisé. Le problème de décision associé à la recherche d'un ordre d'élimination optimal est cependant un problème NP-complet [2]. Si seulement un sous-ensemble des variables doit être éliminé, alors la largeur induite est définie de manière similaire. Dans ce qui suit, nous considérons que l'ensemble des variables à éliminer est implicite.

6.5.2 Largeur induite contrainte

Les requêtes multi-opérateurs imposent certaines contraintes sur l'ordre d'élimination des variables. La complexité est alors donnée par la *largeur induite contrainte* [50, 72].

Définition 6.10. *Soit $\preceq$ un ordre partiel sur V. L'ensemble des* linéarisations *de $\preceq$, noté $lin(\preceq)$, est l'ensemble des ordres totaux $\preceq'$ sur V qui satisfont $(x \preceq y) \rightarrow (x \preceq' y)$.*

Définition 6.11. *Soit $\mathcal{G} = (V_\mathcal{G}, H_\mathcal{G})$ un hypergraphe et soit $\preceq$ un ordre partiel sur $V_\mathcal{G}$. La largeur induite contrainte $w_\mathcal{G}(\preceq)$ de $\mathcal{G}$ avec contraintes sur l'ordre d'élimination données par $\preceq$ ("$x \prec y$" signifie "y doit être éliminée avant x") est définie par $w_\mathcal{G}(\preceq) = \min_{o \in lin(\preceq)} w_\mathcal{G}(o)$.*

Les contraintes sur l'ordre d'élimination induites par une séquence d'éliminations de variables Sov sont formellement définies de la manière suivante :

Définition 6.12. *Soit $Q = (Sov, \mathcal{N})$ une requête sur un réseau PFU tel que $Sov = (op_1, S_1) \cdot (op_2, S_2) \cdots (op_q, S_q)$. L'*ordre partiel *$\preceq_{Sov}$ induit par Sov est donné par $S_1 \prec_{Sov} S_2 \prec_{Sov} \ldots \prec_{Sov} S_q$. Cet ordre partiel force les variables de S_j à être éliminées avant les variables de S_i dès que $i < j$.*

Par exemple, l'ordre partiel induit par la séquence d'élimination $Sov = \min_{x_1,x_2} \sum_{x_3,x_4} \max_{x_5}$ est défini par $\{x_1, x_2\} \prec_{Sov} \{x_3, x_4\} \prec_{Sov} x_5$.

Proposition 6.13. *Soit $\mathcal{G} = (V, \Phi)$ un modèle graphique. Soit Sov une séquence de paires opérateurs-variables sur V. Avec un ordre d'élimination optimal, l'algorithme **VE-answerQ**$(Sov, \circledast, \Phi)$ a une complexité spatiale et temporelle $O(|\Phi| \cdot d^{1+w_\mathcal{G}(\preceq_{Sov})})$, avec d la taille maximale du domaine des variables de V.*

Ainsi, étant donné une requête $Q = (Sov, \mathcal{N})$ sur un réseau PFU $\mathcal{N} = (V, G, P, F, U)$,

— répondre à une requête dans le cas semi-anneau ou semi-groupe est $O(|P \cup F \cup U| \cdot d^{1+w_\mathcal{G}(\preceq_{Sov})})$ en temps et en espace, avec $\mathcal{G} = (V, \{sc(\varphi) \mid \varphi \in P \cup F \cup U\})$ l'hypergraphe associé au réseau PFU ;

— répondre à une requête dans le cas général est $O((|P| + |F| + 1) \cdot d^{1+w_\mathcal{G}(\preceq_{Sov})})$, avec $\mathcal{G} = (V, \{sc(\varphi) \mid \varphi \in P \cup F \cup \{U_0\}\})$ l'hypergraphe associé au réseau PFU dans lequel toutes les fonctions locales d'utilité sont fusionnées pour donner une fonction d'utilité unique U_0.

6.6 Vers une diminution de la largeur induite contrainte

Puisqu'une variation linéaire de la largeur induite engendre une variation exponentielle des complexités temporelle et spatiale, il peut être utile de diminuer la largeur induite.

6.6.1 Relâchement de contraintes sur l'ordre d'élimination

Les contraintes sur l'ordre d'élimination sont définies par l'ordre partiel $\preceq_{Sov}$ induit par la séquence d'élimination Sov. Ce choix peut être inutilement contraignant car certaines éliminations avec des opérateurs distincts peuvent parfois être interverties si l'on tient compte des portées des fonctions locales en jeu.

En effet, si l'on considère par exemple un calcul du type

$$\max_{x_1,\ldots,x_q} \sum_y \max_{x_{q+1}} \left(P_y \times c_{y,x_1} \times \prod_{i\in\{1,\ldots,q\}} c_{x_i,x_{q+1}} \right).$$

alors la largeur induite contrainte vaut q car dans ce cas, $\preceq_{Sov}$ force la variable x_{q+1}, qui est liée avec q autres variables, à être éliminée en premier. Cependant, les portées des fonctions locales garantissent que le calcul suivant donne le même résultat :

$$\max_{x_1} \left(\left(\sum_y P_y \times c_{y,x_1} \right) \times \left(\max_{x_2,\ldots,x_{q+1}} \left(\prod_{i\in\{1,\ldots,q\}} c_{x_i,x_{q+1}} \right) \right) \right).$$

Dans ce cas, la largeur induite devient égale à 1, par exemple en utilisant l'ordre d'élimination $x_{q+1} \prec x_q \prec \ldots \prec x_2 \prec x_1 \prec y$. Ainsi, la complexité théorique passe de $O(d^{q+1})$ à $O(d^2)$.

Par conséquent, définir les contraintes sur l'ordre d'élimination uniquement à partir de Sov est trop contraignant et potentiellement exponentiellement sous-optimal. Ainsi, il est possible de révéler certaines libertés dans l'ordre d'élimination en tenant compte des portées des fonctions locales.

6.6.2 Travail sur l'hypergraphe des fonctions locales

Tout d'abord, les conditions de normalisation peuvent être utilisées pour ne pas effectuer de calculs inutiles du type $\sum_x P_{x\,|\,pa(x)}$ lorsque $P_{x\,|\,pa(x)}$ est une distribution de probabilité conditionnelle sur x sachant $pa_G(x)$.

Ensuite, il peut exister des décompositions qui utilisent plus que la distributivité des opérateurs d'élimination sur les opérateurs de combinaison. Par exemple, si l'on considère un diagramme d'influence associé au calcul

$$\max_{x_1,\ldots,x_q} \sum_y P_y \cdot \left(U_{y,x_1} + \cdots + U_{y,x_q} \right)$$

alors la largeur induite contrainte vaut q. Cependant, étant donné que $\sum_S (U_1 + U_2) = (\sum_S U_1) + (\sum_S U_2)$, on peut écrire

$$\max_{x_1,\ldots,x_q} \sum_y P_y \cdot \left(U_{y,x_1} + \cdots + U_{y,x_q} \right) = (\max_{x_1} \sum_y P_y \cdot U_{y,x_1}) + \cdots + (\max_{x_q} \sum_y P_y \cdot U_{y,x_q})$$

Cette *duplication* de la variable y permet d'obtenir un calcul de largeur 1. Le processus de duplication est utilisable dès que l'opérateur d'élimination est égal à l'opérateur de combinaison. Cette technique peut s'appliquer aux éliminations avec $\forall$ pour les QBF et les QCSP, aux éliminations avec min pour les MDP possibilistes ou avec + sur les diagrammes d'influence.

6.7 Résumé

Dans ce chapitre, nous avons introduit un algorithme d'élimination de variable nommé **VE-answerQ**. Cet algorithme peut répondre à une requête tout en utilisant les factorisations de quantités globales en fonctions locales, pourvu qu'une des deux conditions de décomposabilité (axiomes Ax^{SA} et Ax^{SG}) soit satisfaite. L'utilisation de cet algorithme est résumée par le tableau 6.2, qui montre que dans le cas semi-anneau, son utilisation est plutôt naturelle, que dans le

cas semi-groupe, elle requiert l'utilisation d'éléments appelés potentiels, et que dans le cas général, elle nécessite de combiner toutes les fonctions d'utilité locales en une fonction d'utilité globale.

Le principe de cet algorithme est d'éliminer les variables dans un ordre compatible avec la séquence d'éliminations multi-opérateurs Sov. Ses complexités temporelle et spatiale sont exponentielles en la largeur induite contrainte. Comme indiqué dans la dernière partie de ce chapitre, cet algorithme ne profite cependant pas toujours de la *structure réelle* d'une requête multi-opérateur, et ce pour plusieurs raisons :

1. Premièrement, il est restrictif de définir des contraintes sur l'ordre d'élimination uniquement à partir de la séquence d'éliminations de variables : certaines libertés dans l'ordre des éliminations peuvent apparaître si la portée des fonctions locales est prise en compte.
2. Ensuite, l'algorithme **VE-answerQ** utilise essentiellement la distributivité d'un opérateur de combinaison par rapport à des opérateurs d'élimination, alors qu'il peut exister des décompositions supplémentaires utilisant le mécanisme de duplication précédemment introduit.
3. Enfin, les réseaux PFU contiennent certaines conditions de normalisation qui n'ont jusqu'alors pas été utilisées. C'est un tort, car les conditions de normalisation peuvent complètement masquer la complexité réelle d'un problème.

Utiliser ces trois mécanismes peut diminuer la largeur induite contrainte et conduire à des gains exponentiels en termes de complexité théorique. Ce constat nous amène à définir des techniques plus sophistiquées permettant de faire apparaître la structure réelle des requêtes multi-opérateurs de manière automatique.

Chapitre 7

Structuration des requêtes multi-opérateurs

La largeur induite contrainte peut être améliorée en analysant la structure des requêtes multi-opérateurs, afin de révéler les contraintes réelles sur l'ordre d'élimination, de faire apparaître les décompositions possibles et de supprimer des calculs inutiles. Ce faisant, des gains exponentiels en termes de complexité théorique peuvent être obtenus.

Les techniques introduites dans ce chapitre visent à automatiser le processus de structuration d'une requête. Elles ne sont pas juste des généralisations de techniques déjà existantes définies dans des formalismes couverts par le cadre PFU et peuvent par conséquent contribuer à la résolution des formules booléennes quantifiées, des problèmes de satisfiabilité stochastique, des CSP quantifiés ou stochastiques, des diagrammes d'influence probabilistes ou possibilistes, ou encore des MDP factorisés.

Les étapes de structuration d'une requête vont nous mener progessivement à une nouvelle architecture de calcul appelée l'architecture des *DAGs de clusters multi-opérateurs*. Cette dernière permet de répondre à une requête plus efficacement (en termes de largeur induite) qu'avec l'algorithme **VE-answerQ** introduit au chapitre précédent.

7.1 Retour sur les requêtes multi-opérateurs considérées

Par la suite, nous considérons qu'un des deux axiomes $Ax^{SR'}$ ou $Ax^{SG'}$ est satisfait (notons à nouveau que le cas général est un sous-cas du cas semiring dès lors que l'on agrège toutes les fonctions d'utilité). Ainsi, nous supposons que :

- Au lieu d'avoir une structure de plausibilité, une structure d'utilité et une structure d'utilité espérée, nous considérons plus simplement un MCS $(E, \oplus, \otimes)$ totalement ordonné.
- Les conditions de normalisation sur les composantes d'environnement c d'un réseau PFU (V, G, P, F, U) deviennent $\oplus_c(\otimes_{P_i \in Fact(c)} P_i) = 1_E$.
- La définition opérationnelle de la valeur d'une requête $Q = (Sov, (V, G, P, F, U))$ devient :
  - $Ans(Q) = Sov((\wedge_{F_i \in F} F_i) \star (\otimes_{P_i \in P} P_i) \otimes (\otimes_{U_i \in U} U_i))$ dans le cas semi-anneau $(Ax^{SR'})$,
  - $Ans(Q) = Sov((\wedge_{F_i \in F} F_i) \star (\otimes_{P_i \in P} P_i) \otimes (\oplus_{U_i \in U} U_i))$ dans le cas semi-groupe $(Ax^{SG'})$.

7.2 D'une requête à des nœuds de calcul

Le processus de structuration raisonne à partir d'éléments appelés des *nœuds de calcul*. Ces nouveaux éléments de représentation sont introduits car les éléments considérés jusqu'alors nous empêche d'utiliser certains mécanismes : par exemple, l'utilisation des potentiels dans le cas semi-groupe nous empêche d'utiliser le mécanisme de duplication.

Définition 7.1. *Un* nœud de calcul *sur un ensemble E est :*

- *soit une fonction locale φ à valeurs dans E (nœud de calcul* atomique*) ;*
- *soit un triplet $(sov, \circledast, N)$ tel que $(E, \circledast)$ est un monoïde commutatif, N est un ensemble de nœuds de calcul et sov est une séquence de couples opérateur-variable(s) utilisant des opérateurs op tels que (E, op) soit un monoïde commutatif sur E.*

Par exemple, si P_1, P_2 sont deux fonctions locales de plausibilités et si U_1, U_2 sont deux fonctions locales d'utilité, alors P_1, P_2, U_1, U_2 sont des nœuds de calcul atomiques. Les triplets $n_1 = (\sum_x, \times, \{P_1\})$ et $n_2 = (\sum_{y,z,t}, \times, \{P_2, U_2\})$ sont aussi des nœuds de calcul, tout comme $n_3 = (\min_q \max_r, +, \{n_1, n_2, U_1\})$. Implicitement, un nœud de calcul représente un calcul à réaliser. Ceci est rendu explicite par l'intermédiaire de la définition de la valeur d'un nœud de calcul.

Définition 7.2. *Soit n un nœud de calcul. La* valeur de n*, notée $val(n)$, est définie par*

$$val(n) = \begin{cases} n \text{ si } n \text{ est atomique} \\ sov(\circledast_{n' \in N}\, val(n')) \text{ si } n = (sov, \circledast, N) \end{cases}$$

L'ensemble des variables éliminées par n, noté $V_e(n)$, est vide si n est atomique et est égal à l'ensemble des variables apparaissant dans sov si $n = (sov, \circledast, N)$.

La portée de n*, notée $sc(n)$, est définie par* $sc(n) = \begin{cases} sc(\varphi) \text{ si } n = \varphi \text{ est atomique} \\ (\cup_{n' \in N}\, sc(n')) - V_e(n) \text{ si } n = (sov, \circledast, N) \end{cases}$

L'ensemble des fils de n*, noté $Sons(n)$, est un ensemble de nœuds de calcul qui est vide si n est atomique et qui vaut N si $n = (sov, \circledast, N)$.*

Par exemple, la valeur de n_1 est $val(n_1) = \sum_x P_1$, la valeur de n_2 est $val(n_2) = \sum_{y,z,t}(P_2 \times U_2)$ et la valeur de n_3 est $val(n_3) = \min_q \max_r(val(n_1) + val(n_2) + U_1)$. Ainsi, un nœud $(sov, \circledast, N)$ définit une séquence d'éliminations sov sur une combinaison de nœuds de calcul via l'opérateur $\circledast$. Un nœud de calcul peut être représenté comme la racine d'un arbre de nœuds de calcul (cf. figure 7.1).

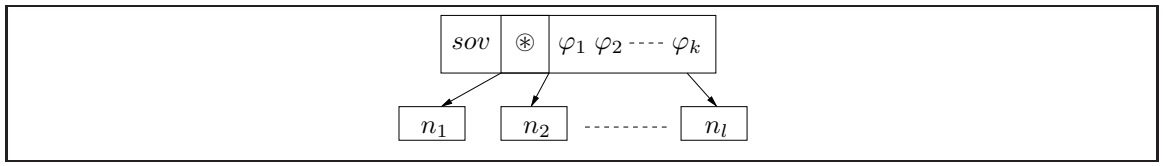

Figure 7.1: Un nœud de calcul $(sov, \circledast, N)$, avec $\{\varphi_1, \ldots, \varphi_k\}$ (respectivement $\{n_1, \ldots, n_l\}$) l'ensemble de nœuds de calcul atomiques (respectivement non atomiques) de N.

Les définitions précédentes sont étendues à des ensembles de nœuds de calcul N par $sc(N) = \cup_{n' \in N} sc(n')$, $V_e(N) = \cup_{n' \in N} V_e(n')$ et $Sons(N) = \cup_{n' \in N} Sons(n')$.

De plus, pour tout $op \in \{\min, \max, \oplus\}$, nous définissons l'ensemble de nœuds de N réalisant des éliminations uniquement avec un opérateur op par $N[op] = \{n \in N \mid n = (op_S, \circledast, N')\}$. L'ensemble $N - N[op]$ est quant à lui noté $N[\neg op]$. Par exemple, pour $N = \{n_1, n_2, n_3\}$ avec

$n_1 = (\sum_x, \times, \{P_1\})$, $n_2 = (\sum_{y,z,t}, \times, \{P_2, U_2\})$ et $n_3 = (\min_q \max_r, +, \{n_1, n_2, U_1\})$, nous avons $N[+] = \{n_1, n_2\}$ et $N[\neg+] = \{n_3\}$.

Enfin, étant donné un ensemble de nœuds de calcul N, nous définissons N^{+x} (respectivement N^{-x}) comme l'ensemble des nœuds de N dont la portée contient (resp. ne contient pas) la variable x : $N^{+x} = \{n \in N \mid x \in sc(n)\}$ (resp. $N^{-x} = \{n \in N \mid x \notin sc(n)\}$).

La valeur d'un nœud de calcul peut facilement être reliée à la réponse à une requête $Q = (Sov, (V, G, P, F, U))$:

— Dans le cas semi-anneau, $Ans(Q) = val(n_0)$ avec $n_0 = (Sov, \otimes, P \cup F \cup U)$.

— Dans le cas semi-groupe, $Ans(Q) = val(n_0)$ avec $n_0 = (Sov, \oplus, \{(\emptyset, \otimes, P \cup F \cup \{U_i\}), U_i \in U\})$. En effet, $val(n_0) = Sov(\oplus_{U_i \in U}(\otimes_{\varphi \in P \cup F \cup \{U_i\}} \varphi)) = Sov((\wedge_{F_i \in F} F_i) \star (\otimes_{P_i \in P} P_i) \otimes (\oplus_{U_i \in U} U_i))$.

Nous définissons également de manière explicite la notion d'ordre d'élimination compatible avec une séquence d'éliminations.

Définition 7.3. *Un ordre d'élimination o sur V est* compatible avec *une séquence d'élimination Sov portant sur V ssi $o \in lin(\preceq_{Sov})$. Si $op(x)$ correspond à l'opérateur d'élimination de x dans Sov, alors $Sov(o)$ représente la séquence d'éliminations suivante ($o(k)$ est la k-ème variable eliminée dans o) :*

$$Sov(o) = op(o(n))_{o(n)} \cdots op(o(2))_{o(2)} \cdot op(o(1))_{o(1)}$$

Exemple 7.4. *Soit $Sov = \min_{x_1,x_2} \sum_{x_3,x_4} \max_{x_5}$. L'ordre d'élimination $o : x_1 \prec x_2 \prec x_4 \prec x_3 \prec x_5$ est compatible avec Sov et $Sov(o) = \min_{x_1} \min_{x_2} \sum_{x_4} \sum_{x_3} \min_{x_5}$. L'ordre d'élimination $o' : x_4 \prec x_2 \prec x_1 \prec x_3 \prec x_5$ n'est pas compatible avec Sov car $x_4 \prec x_2$ alors que $x_2 \prec_{Sov} x_4$.*

Une structuration en deux temps

Structurer une requête est synonyme de réécrire le nœud de calcul initial n_0. Pour ce faire, nous utilisons un mécanisme de structuration en deux temps :

1. Nous cherchons d'abord ce que nous appelons la *macrostructure* d'une requête multi-opérateur. Cette première étape revient à révéler les libertés disponibles dans l'ordre d'élimination des variables et à déterminer les décompositions possibles (elle ne correspond pas à déterminer un ordre d'élimination optimal).

   Etant donné un ordre d'élimination o compatible avec la séquence d'élimination Sov d'une requête, cette macrostructure est obtenue par l'intermédiaire de règles de réécriture qui permettent de *simuler* les décompositions induites par l'élimination des variables de droite à gauche de $Sov(o)$. Une règle de réécriture $R : n_1 \rightsquigarrow n_2$ permet de transformer un nœud de calcul n_1 en un nœud de calcul $n_2 = R(n_1)$. Plus précisément, trois types de règles sont introduites :

   — des règles de *décomposition*, qui décomposent la structure en utilisant notamment le mécanisme de duplication ;

   — des règles de *recomposition*, qui permettent de réveler des libertés dans l'ordre d'élimination des variables ;

   — des règles de *simplification*, qui permettent de supprimer des calculs inutiles du fait des normalisations.

2. Une fois la macrostructure obtenue, la seconde étape de structuration, plus fine que la précédente, consiste à exploiter au mieux les libertés dans l'ordre d'élimination qui ont été révélées par la première étape. Cette seconde phase est réalisée en utilisant des techniques classiques de décomposition arborescente.

Nous supposons ici qu'il n'y a pas de faisabilité. Si des faisabilités interviennent, alors des mécanismes de structuration peuvent également être définis (voir la version anglaise de la thèse). Le processus de structuration diffère suivant si le cas considéré est le cas semi-groupe ou le cas semi-anneau.

7.3 Structuration des requêtes multi-opérateurs : le cas semi-anneau

Pour rendre les règles de réécriture plus lisibles, nous notons les nœuds de calcul $(sov, \otimes, N)$ simplement par (sov, N) car dans le cas semi-anneau, l'opérateur de combinaison utilisé est à chaque fois $\otimes$.

7.3.1 Macrostructuration d'une requête par règles de réécriture

Soit o un ordre d'élimination compatible avec la séquence d'élimination Sov d'une requête. Notre point de départ est le nœud de calcul non structuré obtenu directement à partir de la requête. Ce nœud s'écrit $n_0 = (Sov(o), \otimes, P \cup U)$, ou autrement dit $(Sov(o), P \cup U)$ dans le cas semi-anneau. Il peut être vu comme un arbre de nœuds de calcul (*Tree of Computation Nodes*, CNT) et est de ce fait noté $CNT_0(Q, o)$. Par exemple, à la figure 7.2, $CNT_0(Q, o)$ correspond au premier nœud. L'application de règles de réécriture va générer une séquence d'arbres de nœuds de calcul : pour tout $k \in \{0, \ldots, |Sov| - 1\}$, la macrostructure à l'étape $k+1$, notée $CNT_{k+1}(Q, o)$, est obtenue à partir de la structure $CNT_k(Q, o)$ à l'étape k en considérant l'élimination la plus à droite encore non traitée dans $Sov(o)$ et en appliquant trois règles de réécriture différentes :

1. Une première règle dite de décomposition, notée DR (*Decomposition Rule*), utilise d'une part la distributivité de $\otimes$ par rapport aux opérateurs d'élimination (pour faire en sorte que l'élimination d'une variable x fasse intervenir seulement les fonctions locales ayant x dans leur portée) et d'autre part le procédé de duplication. La règle de réécriture DR met en œuvre ces deux types de décompositions. Elle s'écrit de la manière suivante :

$$\boxed{DR} \qquad \left(\underset{x}{sov.\, op}, N\right) \rightsquigarrow \begin{cases} (sov, N^{-x} \cup \{(op_x, \{n\}) \mid n \in N^{+x}\}) \text{ si } op = \otimes \\ (sov, N^{-x} \cup \{(op_x, N^{+x})\}) \text{ sinon} \end{cases}$$

   Dans la figure 7.2, DR transforme la structure initiale $CNT_0(Q, o) = (\min_{x_1} \max_{x_2} \max_{x_3} \min_{x_4} \max_{x_5}, \{\varphi_{x_3,x_4}, \varphi_{x_1,x_4}, \varphi_{x_1,x_5}, \varphi_{x_2,x_5}, \varphi_{x_3,x_5}\})$ en la structure
   $CNT_1(Q, o) = (\min_{x_1} \max_{x_2} \max_{x_3} \min_{x_4}, \{\varphi_{x_3,x_4}, \varphi_{x_1,x_4}, (\max_{x_5}, \{\varphi_{x_1,x_5}, \varphi_{x_2,x_5}, \varphi_{x_3,x_5}\})\})$
   (cas $op \neq \otimes$ utilisant juste la distributivité de $\wedge$ par rapport à max).
   L'élimination de x_4 avec l'opérateur min transforme ensuite $CNT_1(Q, o)$ en $CNT_2(Q, o) = (\min_{x_1} \max_{x_2} \max_{x_3}, \{(\min_{x_4}, \{\varphi_{x_3,x_4}\}), (\min_{x_4}, \{\varphi_{x_1,x_4}\}), (\max_{x_5}, \{\varphi_{x_1,x_5}, \varphi_{x_2,x_5}, \varphi_{x_3,x_5}\})\})$
   (cas $op = \otimes = \min$, utilisant une duplication de x_4).

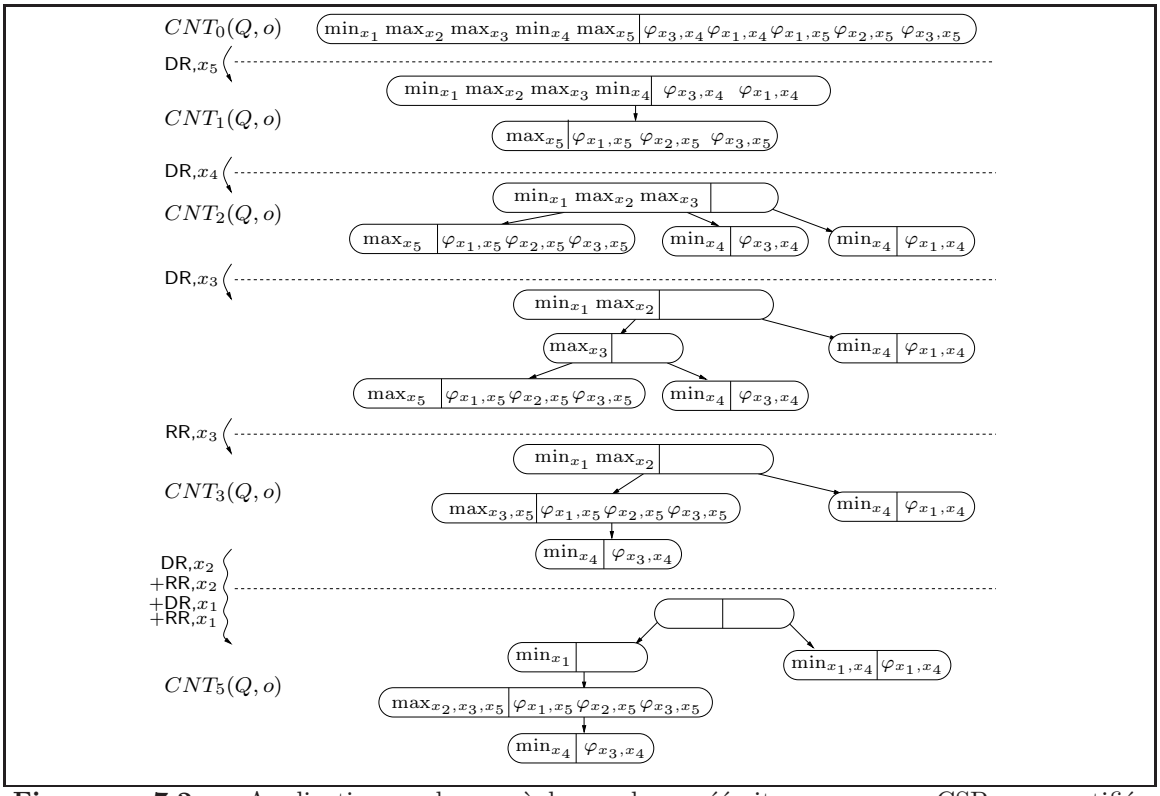

Figure 7.2: Application des règles de réécriture sur CSP quantifié: $\min_{x_1} \max_{x_2,x_3} \min_{x_4} \max_{x_5} (\varphi_{x_3,x_4} \wedge \varphi_{x_1,x_4} \wedge \varphi_{x_1,x_5} \wedge \varphi_{x_2,x_5} \wedge \varphi_{x_3,x_5})$, avec l'ordre d'élimination $o : x_1 \prec x_2 \prec x_3 \prec x_4 \prec x_5$.

2. Une seconde règle dite de recomposition, notée RR (*Recomposition Rule*), a pour but de révéler des libertés dans l'ordre d'élimination pour les nœuds créés par la règle DR.

$$\boxed{RR} \qquad \left(\underset{x}{op}, N\right) \rightsquigarrow \left(\underset{\{x\} \cup V_e(N[op])}{op}, N[\neg op] \cup Sons(N[op])\right)$$

La formulation de RR signifie que si un nœud de calcul est chargé de faire une élimination op_x et a des fils qui doivent faire des éliminations du type op_S en utilisant le même opérateur op, alors il n'y a aucune raison d'imposer aux variables de S d'être éliminées avant x. La règle RR rend cela explicite en fusionnant les nœuds de calcul correspondant. Sur l'exemple de la figure 7.2, RR transforme le nœud $(\max_{x_3}, \{(\min_{x_4}, \{\varphi_{x_3,x_4}\}), (\max_{x_5}, \{\varphi_{x_1,x_5}, \varphi_{x_2,x_5}, \varphi_{x_3,x_5}\})\})$ créé par $DR(CNT_2(Q,o))$, en le nœud "recomposé"

$$(\max_{x_3,x_5}, \{(\min_{x_4}, \{\varphi_{x_3,x_4}\}), \varphi_{x_1,x_5}, \varphi_{x_2,x_5}, \varphi_{x_3,x_5}\})$$

qui apparaît au sein de la structure $CNT_3(Q,o)$. Ainsi, la règle RR révèle que même si $x_3 \prec_{Sov} x_5$, rien n'empêche d'éliminer x_3 avant x_5.

3. Une troisième règle dite de simplification, notée SR (*Simplification Rule*), permet d'exploiter les conditions de normalisation $\oplus_c(\otimes_{P_i \in Fact(c)} P_i) = 1_E$:

$$\boxed{SR} \qquad [\text{Préconditions} : (c \in \mathcal{C}_E(G)) \wedge (c \cap (S \cup sc(N)) = \emptyset)]$$
$$(\underset{S \cup c}{\oplus}, N \cup Fact(c)) \rightsquigarrow (\underset{S}{\oplus}, N)$$

Par exemple, SR transforme un nœud $n = (\sum_{x,y,z}, \{P_{x\,|\,y,z}, P_y, P_z, c_y\})$ en le nœud simplifié $n' = (\sum_{y,z}, \{P_y, P_z, c_y\})$ grâce à la condition de normalisation $\sum_x P_{x\,|\,y,z} = 1$. Une autre application de SR génère un nœud de calcul encore plus simple, $n'' = (\sum_{y,}, \{P_y, c_y\})$. SR ne peut alors plus être appliquée sur n''. Intrinsèquement, la raison pour laquelle des simplifications peuvent être disponibles est qu'il peut être ardu pour la personne spécifiant une requête d'identifier toutes les indépendances conditionnelles disponibles.

Il est important de noter que la règle SR peut elle-même révéler de nouvelles décompositions et de nouvelles libertés dans l'ordre d'élimination. C'est pourquoi les techniques utilisées peuvent restructurer un nœud qui a été simplifié (voir la version anglaise pour les détails techniques de la chose).

Partant d'une requête $Q = (Sov, \mathcal{N})$ et d'un ordre d'élimination o compatible avec Sov, l'arbre de nœuds de calcul obtenu après utilisation des règles de réécriture DR, RR et DR en traitant les éliminations de droite à gauche de $Sov(o)$ est noté $CNT(Q, o)$.

Quelques propriétés satisfaites par la macrostructure obtenue

Le théorème 7.5 montre que l'arbre de nœuds de calcul $CNT(Q, o)$ obtenu à partir d'une requête $Q = (Sov, \mathcal{N})$ et d'un ordre d'élimination o est en réalité indépendant de l'ordre d'élimination $o \in lin(\preceq_{Sov})$ choisi au départ.

Théorème 7.5. *Soit $Q = (Sov, \mathcal{N})$ une requête. Alors, pour tous $o, o' \in lin(\preceq_{Sov})$, $CNT(Q, o) = CNT(Q, o')$.*

Ce résultat nous permet de noter $CNT(Q, o)$ simplement sous la forme $CNT(Q)$.

Comme indiqué par le théorème 7.6, la structure finale obtenue est correcte, c'est-à-dire que sa valeur est bien égale à la réponse $Ans(Q)$ à la requête Q considérée.

Théorème 7.6. *Soit $Q = (Sov, \mathcal{N})$ une requête. Alors, $val(CNT(Q)) = Ans(Q)$.*

Enfin, il est enfin possible de montrer que le processus de structuration a une complexité polynomiale en temps et en espace.

7.3.2 Une seconde étape de structuration utilisant des décompositions arborescentes

La macrostructure obtenue est un arbre de nœuds de calcul mono-opérateurs de la forme $(\min_S, \otimes, N)$, $(\max_S, \otimes, N)$ ou $(\oplus_S, \otimes, N)$. Nous pouvons maintenant rechercher des structures plus fines en utilisant un processus de structuration interne pour chaque nœud de calcul obtenu. L'objectif est ici d'utiliser au mieux les libertés dans l'ordre d'élimination révélées par le processus de macrostructuration.

Pour ce faire, des techniques de décomposition arborescente sont utilisées (voir la version anglaise pour une présentation complète de la notion de décomposition arborescente). Ces techniques sont des outils génériques utilisés pour traiter notamment des CSPs ou des réseaux bayésiens. Elles exploitent les propriétés topologiques des modèles graphiques considérés de manière à structurer un problème donné en un arbre de problèmes élémentaires à résoudre [89, 2, 88, 54, 12, 57]. La complexité du problème global est alors fonction de la complexité du problème élémentaire le plus

dur à résoudre, d'où l'intérêt de chercher de bonnes décompositions du problème initial. Les techniques de décomposition arborescente sont basiquement utilisées pour des problèmes ne faisant intervenir qu'un seul opérateur d'élimination et qu'un seul opérateur de combinaison, ce qui est le cas de tous les nœuds de calcul mono-opérateurs obtenus après la phase de macrostructuration.

Une fois les techniques de décomposition arborescente utilisées, la structure obtenue contient d'une part une macrostructure donnée par les nœuds de calcul et d'autre part une structure plus fine à l'intérieur de chaque nœud. Nous obtenons ainsi une architecture de calcul générale appelée *arbre de clusters multi-opérateur*.

Définition 7.7. *Un arbre de clusters multi-opérateur (*Multi-operator Cluster Tree*, MCTree) est un arbre enraciné* (C, E) [1] *dont chaque sommet* $c \in C$*, appelé un cluster, est étiqueté par trois éléments :*

- *un ensemble de variables* $V(c)$*,*
- *un ensemble de fonctions locales* $\Phi(c)$ *à valeurs dans un ensemble* E*,*
- *et un couple* $(\oplus^c, \otimes^c)$ *d'opérateurs sur* E *tels que* $(E, \oplus^c, \otimes^c)$ *est un semi-anneau commutatif.*

La largeur d'un MCTree est définie par $w = \max_{c \in C} |V(c)| - 1$.

Nous spécifions explicitement un opérateur d'élimination et un opérateur de combinaison à utiliser dans chaque cluster de manière à gérer proprement la nature multi-opérateur des requêtes considérées. La figure 7.3 montre un exemple de MCTree qui peut être obtenu à partir d'un problème de satisfiabilité stochastique étendue [62].

Définition 7.8. *La valeur d'un cluster* c *d'un MCTree est définie par*

$$val(c) = \bigoplus_{V(c)-V(pa(c))}^{c} \left(\left(\bigotimes_{\varphi \in \Phi(c)}^{c} \varphi \right) \otimes^c \left(\bigotimes_{s \in Sons(c)}^{c} val(s) \right) \right)$$

La valeur $val(M)$ *d'un MCTree* M *est égale à la valeur de son cluster racine.*

Théorème 7.9. *Soit* Q *une requête. Soit* M *un MCTree que l'on peut obtenir à partir de* $CNT(Q)$*. Alors,* $val(M) = Ans(Q)$*. De plus, chaque règle de décision optimale dans* $val(M)$ *pour une variable non dupliquée est aussi une règle de décision optimale dans* $Ans(Q)$ *et pour toute variable de décision dupliquée, il existe au moins une règle de décision optimale dans* $val(M)$ *qui est aussi optimale dans* $Ans(Q)$*.*

En fait, les règles de décision optimales peuvent être mémorisées sur les séparateurs du MCTree (le séparateur entre deux clusters c et $s \in Sons(c)$ est l'ensemble de variables $V(c) \cap V(s)$).

7.3.3 Comparaison avec une approche non structurée

Une analyse fine de la structure d'une rêquete peut engendrer des gains exponentiels, comme l'ont montré les exemples introduits à la section 6.6. Un résultat plus fort peut être établi. Ce résultat est qu'en termes de largeur induite (ou largeur d'arbre), l'approche structurée est *toujours au moins aussi bonne* qu'une approche non structurée du type de celle utilisée dans l'algorithme **VE-answerQ**.

1. C est l'ensemble des sommets de l'arbre, appelés des clusters, et E est l'ensemble des arêtes de l'arbre.

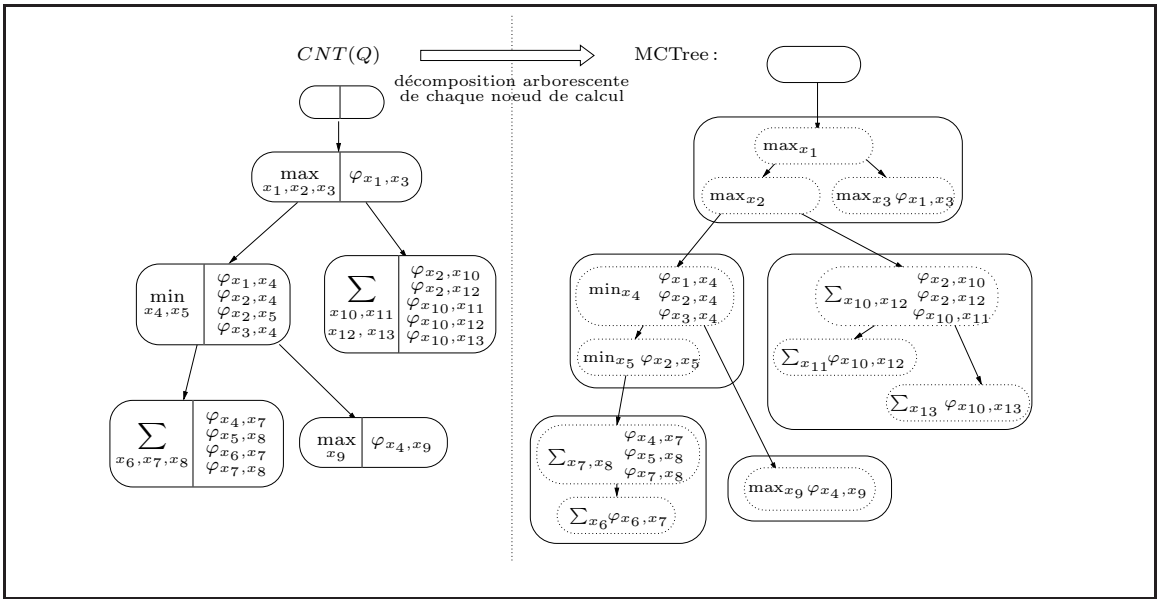

Figure 7.3: Exemple d'arbre de clusters multi-opérateur obtenue à partir de $CNT(Q)$. Notons qu'un cluster c est représenté par (1) l'ensemble de variables $V(c) - V(pa(c))$ qu'il élimine, son opérateur d'élimination $\oplus^c$ et l'ensemble $\Phi(c)$ des fonctions qui lui sont associées ; dans le cas semi-anneau, nous avons toujours $\otimes^c = \otimes$; (2) l'ensemble de ses fils.

Définition 7.10. *La largeur d'un arbre de nœuds de calcul CNT, notée w_{CNT}, est égale à la largeur minimale d'un MCTree qui peut être obtenu à partir d'une décomposition arborescente de CNT.*

Théorème 7.11. *Soit $Q = (Sov, \mathcal{N})$ une requête sur un réseau PFU $\mathcal{N} = (V, G, P, \emptyset, U)$. Soit $\mathcal{G} = (V, \{sc(\varphi), \varphi \in P \cup U\})$ l'hypergraphe associé à $\mathcal{N}$. Alors, $w_{CNT(Q)} \leq w_{\mathcal{G}}(\preceq_{Sov})$.*

Pour le QCSP pris en exemple à la figure 7.2, $w_{CNT(Q)} = 1$ alors que la largeur induite contrainte vaut $w_{\mathcal{G}}(\preceq_{Sov}) = 3$: ainsi, la complexité théorique passe de $O(|\Phi| \cdot d^4)$ à $O(|\Phi| \cdot d^2)$.

Des différences plus importantes peuvent être observées entre la largeur avant structuration ($w_{\mathcal{G}}(\preceq_{Sov})$) et la largeur après structuration ($w_{CNT(Q)}$) sur des problèmes de plus grande taille. Des expérimentations ont par exemple été réalisées sur des instances de la librairie QBF, pour lesquels des gains d'un facteur pouvant aller jusqu'à 10 *en termes de largeur* ont pu être observés (rappelons que la complexité théorique est exponentielle en la largeur induite).

Comparaison avec les approches existantes Les règles de réécriture que nous avons définies peuvent être reliées avec l'approche des arbres de quantificateurs (*quantifier tree* [5]) récemment introduite pour résoudre des formules booléennes quantifiées. Le principe de cette approche est d'analyser les structures cachées de QBF exprimées en forme prénexe normale conjonctive par l'intermédiaire de mécanismes de structuration. Cette analyse génère des gains importants en termes de temps de résolution d'une QBF. Les techniques de structuration utilisées pour construire les arbres de quantificateurs correspondent exactement à l'application de la règle de décomposition DR instanciée pour la structure algébrique utilisée par les QBF, i.e. pour $\oplus = \vee$ et $\otimes = \wedge$.

Les MCTrees fournissent une explication théorique aux résultats expérimentaux constatés avec les arbres de quantificateur, via la largeur d'arbre. De plus, étant définie dans un cadre algébrique général, notre approche étend et généralise entièrement l'approche des arbres de quantificateurs.

Elle est en effet applicable à d'autres formalismes que les QBF, incluant les QCSP, SSAT ou les CSP stochastiques. En outre, les arbres de quantificateurs n'utilisent ni de règle de recomposition RR, ni de décomposition en arbre de clusters minimisant la largeur d'arbre, ni de règle de simplification (ce dernier point étant relativement normal puisqu'une formule booléenne quantifiée ne fait pas intervenir de normalisations).

7.4 Structuration des requêtes multi-operateurs : le cas semi-groupe

7.4.1 Processus de structuration

La structuration des requêtes dans le cas semi-anneau nous conduit à une architecture de calcul générale, appelée l'architecture MCTree, qui fait intervenir plusieurs opérateurs d'élimination et un seul opérateur de combinaison. Les choses sont techniquement plus complexes dans le cas semi-groupe, qui fait intervenir plusieurs opérateurs de combinaison ($\otimes$ et $\oplus$). A nouveau, nous utilisons un mécanisme de structuration en deux temps avec une phase de macrostructuration et une phase de décomposition en arbres de clusters.

Cette fois, la phase de macrostructuration crée une séquence de DAGs de nœuds de calcul au lieu d'une séquence d'arbres de nœud de calcul. La raison intuitive est que dans le cas semi-groupe, les mêmes plausibilités s'appliquent sur toutes les fonctions locales d'utilité, puisque par exemple on peut écrire $\sum_S(P \times (U_1 + U_2 + U_3)) = (\sum_S(P \times U_1)) + (\sum_S(P \times U_2)) + (\sum_S(P \times U_3))$. Ceci explique que certains calculs fait sur les plausibilités soient partagés par plusieurs nœuds de calcul, d'où la structure de DAG.

Une fois toutes les éliminations considérées, nous obtenons un DAG de nœuds de calcul noté $CNDAG(Q)$, sur lequel nous pouvons alors utiliser des techniques de décomposition arborescente. Ceci nous mène finalement à une architecture de calcul général appelée un DAG de clusters multi-opérateur (*Multi-operator Cluster DAG*, MCDAG).

Définition 7.12. *Un* DAG de clusters multi-opérateur *est un DAG dans lequel chaque sommet c, appelé un cluster, est étiqueté par trois éléments :*

- *un ensemble de variables $V(c)$,*
- *un ensemble $\Phi(c)$ de fonctions locales à valeurs dans E,*
- *et un couple $(\oplus^c, \otimes^c)$ d'opérateurs sur E tels que $(E, \oplus^c, \otimes^c)$ est un semi-anneau commutatif.*

La largeur d'un MCDAG est définie par $w = \max_{c \in C} |V(c)| - 1$. La hauteur d'un MCDAG est égale au nombre maximal de variables qui apparaissent sur un chemin allant de la racine vers une feuille du MCDAG.

Définition 7.13. *La valeur d'un cluster c d'un MCDAG est définie par*

$$val(c) = \bigoplus_{V(c)-V(pa(c))}^{c} \left(\left(\bigotimes_{\varphi \in \Phi(c)}^{c} \varphi \right) \otimes^c \left(\bigotimes_{s \in Sons(c)}^{c} val(s) \right) \right)$$

La valeur d'un MCDAG est la valeur de son nœud racine.

La figure 7.4 donne un exemple de MCDAG qui peut être obtenu à partir d'un DAG de nœuds de calcul $CNDAG(Q)$.

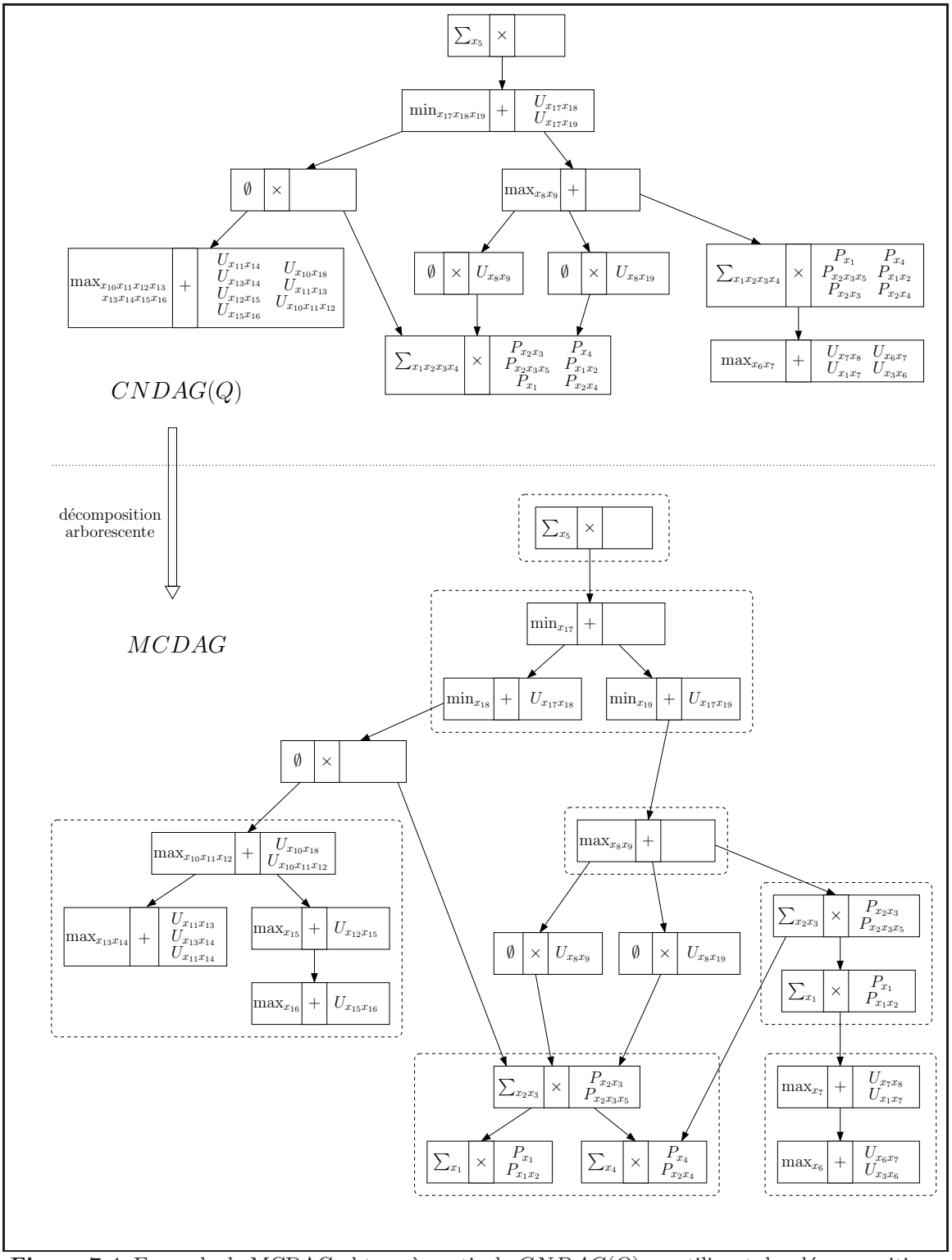

Figure 7.4: Exemple de MCDAG obtenu à partir de $CNDAG(Q)$ en utilisant des décompositions arborescentes et en fusionnant des clusters faisant exactement le même calcul.

7.4.2 Comparaison avec une approche non structurée

Définition 7.14. *Soit Q une requête. La largeur de $CNDAG(Q)$ est la largeur minimale d'un MCDAG qui peut être obtenu à partir de $CNDAG(Q)$ en utilisant des décompositions en arbre de clusters.*

Le théorème 7.15 ci-dessous montre que structurer des requêtes multi-opérateurs ne peut être que bénéfique en termes de largeur, et ainsi la largeur d'arbre "subie" par un algorithme d'élimination de variables sur un MCDAG n'est jamais plus grande que la largeur d'arbre subie par l'algorithme **VE-answerQ** qui utilise une approche non structurée.

Théorème 7.15. *Soit $Q = (Sov, \mathcal{N})$ une requête sur un réseau PFU $\mathcal{N} = (V, G, P, \emptyset, U)$. Soit $\mathcal{G} = (V, \{sc(\varphi), \varphi \in P \cup U\})$ l'hypergraphe associé à $\mathcal{N}$. Alors, $w_{CNDAG(Q)} \leq w_{\mathcal{G}}(\preceq_{Sov})$.*

Comparaison avec des approches existantes Comparés aux architectures de calcul existantes pour résoudre des diagrammes d'influence, les MCDAGs peuvent donner des résultats potentiellement exponentiellement meilleurs étant donné qu'ils peuvent faire décroître linéairement la largeur d'arbre en utilisant :

- le mécanisme de duplication, qui a en fait déjà été utilisé dans la littérature, où il était appliqué à la volée durant le processus d'élimination [26] ; dans notre cas, la duplication est utilisée en preprocessing et en synergie avec d'autres règles de réécriture, ce qui peut accroître son impact. L'idée principale derrière la duplication est qu'un diagramme d'influence exprime deux types d'indépendances : l'une pour la distribution globale de probabilité qu'il définit, et l'autre pour la fonction globale d'utilité. Dans l'architecture MCDAG, ces deux aspects sont clairement utilisés alors que toutes les architectures à base de potentiels utilisent une forme plus faible d'indépendance, qui mixe les deux formes énoncées ci-dessus ;
- le relâchement des contraintes sur l'ordre d'élimination ; ce mécanisme peut être relié avec la notion d'information pertinente (*relevant information*) utilisée classiquement pour réduire la complexité spatiale de stockage des règles de décision. Dans notre cas, l'affaiblissement des contraintes sur l'ordre d'élimination a aussi un intérêt en termes de complexité temporelle ;
- l'utilisation des conditions de normalisation : classiquement, ces conditions sont utilisées à la volée par un mécanisme connu sous le nom de Lazy Propagation [64]. Les conditions de normalisation sont plus exploitées dans notre cas car elles peuvent modifier l'architecture de calcul elle-même en révélant des libertés cachées dans l'ordre d'élimination.

Enfin, l'architecture MCDAG n'utilise aucune opération de division.

Les MCDAGs peuvent également être utilisés pour des diagrammes d'influence possibilistes pessimistes ou des problèmes de planification classique dans lesquels il s'agit de trouver un plan permettant d'arriver à au moins un but parmi un ensemble de buts.

7.5 Conclusion : l'architecture MCDAG, une architecture de calcul générique

Ce chapitre a présenté comment structurer des requêtes multi-opérateurs de manière systématique. Le processus de structuration met en jeu deux étapes majeures :

— Une étape de macrostructuration, à base de règles de réécriture.
— Une étape de décomposition en arbres de clusters, exploitant les libertés révélées par la première étape.

Ceci nous conduit à l'architecture MCTree dans le cas semi-anneau (cf. définition 7.7 page 73), et à l'architecture MCDAG dans le cas semi-groupe (cf. définition 7.12 page 75). Ces deux architectures, qui satisfont des propriétés de correction et d'unicité, permettent d'obtenir une meilleure largeur induite (ou largeur d'arbre). Comparée à d'autres architectures de calculs faisant de l'élimination de variables, l'architecture MCDAG est la seule à utiliser à la fois plusieurs opérateurs d'élimination et plusieurs opérateurs de combinaison.

L'architecture MCTree étant un cas particulier de l'architecture MCDAG, nous pouvons en fait raisonner uniquement sur l'architecture MCDAG et ne plus dissocier les cas semi-anneau et semi-groupe par la suite, comme illustré à la figure 7.5.

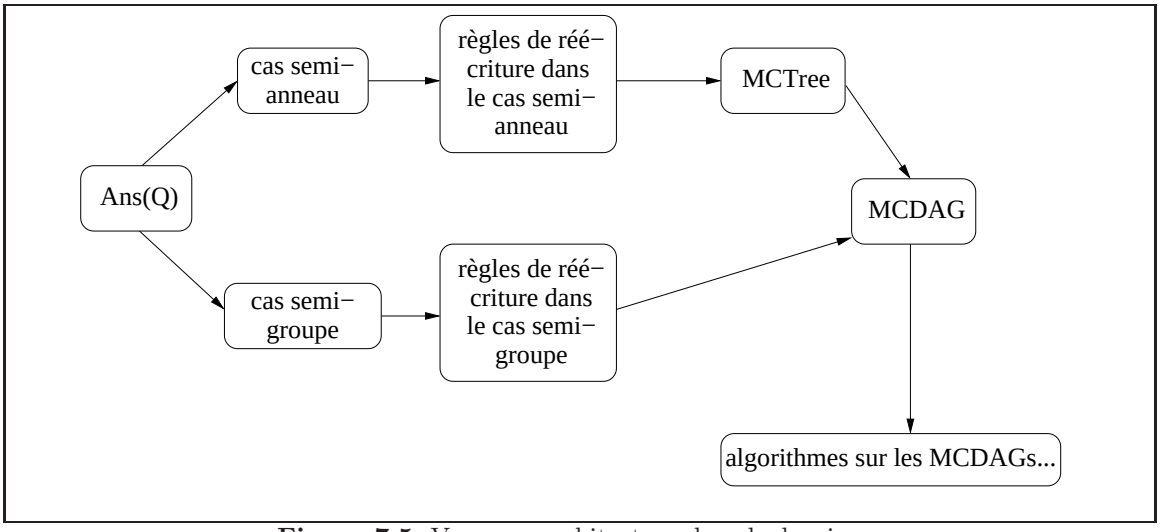

Figure 7.5: Vers une architecture de calcul unique.

Une autre manière de formuler cette conclusion consiste à dire que calculer la réponse $Ans(Q)$ à une requête Q ainsi que des règles de décision optimales correspond à résoudre le problème suivant :

Soit $(E, \oplus, \otimes)$ un MCS totalement ordonné.
Soit m un MCDAG mettant en jeu des fonctions locales à valeurs dans E et des clusters utilisant $(\oplus^c, \otimes^c) \in \{(\min, \oplus), (\max, \oplus), (\min, \otimes), (\max, \otimes), (\oplus, \otimes)\}$
Calculer la valeur de m et des règles de décision optimales.

Une fois l'architecture MCDAG obtenue, un nouvel algorithme générique d'élimination de variables consiste tout simplement à dire que dès qu'un cluster c a reçu la valeur $val(s)$ de chacun de ses fils $s \in Sons(c)$, il peut calculer sa propre valeur

$$val(c) = \oplus^c{}_{V(pa(c))-V(c)} \left(\left(\otimes^c{}_{\varphi \in \Phi(c)} \varphi \right) \otimes^c \left(\otimes^c{}_{s \in Sons(c)} val(s) \right) \right)$$

puis la transmettre à chacun de ses parents. La valeur du cluster racine est alors égale à la réponse à la requête.

Pour chaque cluster c, $val(c)$ peut être calculé en éliminant une à une les variables de $V(pa(c)) - V(c)$, comme fait précédemment, ou en considérant toutes les variables de $V(pa(c)) - V(c)$ simultanément. La seconde approche, connue sous le nom d'élimination de clusters (*Cluster-Tree Elimination*, CTE [6]), généralise l'algorithme d'élimination de variables et fournit d'une part la même complexité théorique et d'autre part une meilleure complexité spatiale, exponentielle en la taille du plus grand séparateur entre deux clusters du MCDAG. De telles méthodes sont également connues dans la littérature sous le nom de programmation dynamique (non sérielle), algorithme d'arbre de jonction ou de relaxation parfaite [69].

Même si tous ces algorithmes peuvent répondre à des requêtes, ils n'utilisent ni des mécanismes de retour arrière (*backtrack*), ni des mécanismes d'élagage de l'espace de recherche. Partant de ce constat, la prochaine étape consiste à utiliser de tels outils au sein de l'architecture MCDAG.

Chapitre 8

Algorithme de recherche arborescente structurée sur l'architecture MCDAG

Répondre à une requête PFU est équivalent à calculer la valeur d'un MCDAG. Cette seconde tâche peut être réalisée de manière relativement naturelle en utilisant un algorithme d'élimination de variables (VE) ou d'élimination de clusters (CTE). Ces algorithmes calculent la valeur d'un MCDAG en propageant les informations des feuilles vers la racine. L'algorithme VE fournit une complexité temporelle exponentielle en la largeur du MCDAG, au prix d'une complexité spatiale aussi exponentielle en la largeur du MCDAG. L'algorithme CTE donne la même complexité temporelle, tout en assurant une meilleure complexité spatiale, exponentielle seulement en la taille maximale d'un séparateur entre deux clusters du MCDAG.

Parallèlement, une méthode de recherche telle que la recherche arborescente en profondeur d'abord offre une complexité spatiale linéaire. De plus, malgré une complexité spatiale théorique plus élevée, une recherche arborescente est souvent meilleure qu'un algorithme d'élimination de variables en pratique, notamment lorsque des techniques d'élagage de l'espace de recherche par des bornes sont utilisées.

Afin de bénéficier à la fois de l'efficacité pratique de la recherche arborescente et de la bonne complexité temporelle théorique des algorithmes d'élimination de variables, nous introduisons un algorithme générique de recherche arborescente structurée qui tire partie des décompositions structurelles exprimées par l'architecture MCDAG. Dans son principe, une telle idée n'est pas nouvelle : en particulier, plusieurs méthodes de recherche arborescente structurée ont été récémment définies [21, 49, 31].

Cependant, ces méthodes existantes permettent basiquement de calculer une séquence mono-opérateur d'éliminations de variables sur une combinaison mono-opérateur de fonctions locales. Cette nature mono-opérateur facilite grandement l'utilisation de bornes. De plus, les schémas existants fonctionnent soit avec des opérateurs bien spécifiques, soit avec une structure algébrique faisant des hypothèses plus fortes que celles faites en considérant un MCS (*Monotonic Commutative Semiring*) totalement ordonné.

Ainsi, nous avons besoin de définir des algorithmes de recherche arborescente structurée capable de gérer la nature multi-opérateur des requêtes considérées (ou, de manière équivalente, des MC-DAGs considérés). Comme précédemment indiqué cela soulève de nouvelles questions concernant l'utilisation de bornes dans le contexte d'une alternance d'éliminations utilisant les opérateurs min, max et $\oplus$.

8.1 Algorithmes de recherche arborescente structurée existants

Différents algorithmes de recherche arborescente structurée ont déjà été définis dans la littérature. Parmi eux, on peut entre autres trouver les algorithmes de recherche AND/OR [31], la méthode *recursive conditioning* [21] et la méthode BTD (Backtrack bounded by Tree Decomposition [49]). La méthode de recherche que nous proposons s'inspire de l'algorithme BTD. Voir la version anlgaise de la thèse pour une présentation de ces schémas existants.

Recherche arborescente structurée sur un MCDAG Nous présentons de manière incrémentale un algorithme qui utilise la structure des MCDAGs et qui généralise l'algorithme BTD utilisé sur les CSP et les CSP valués. Nous partons d'une version de base qui correspond à une recherche arborescente structurée sans mémorisation et sans utilisation de bornes, pour arriver à une recherche arborescente structurée utilisant à la fois des bornes pour élaguer l'espace de recherche et des techniques de mémorisation permettant d'éviter des calculs redondants.

Dans ce qui suit, nous supposons qu'il n'y a ni variables libres dans la requête, ni faisabilités. Il est possible d'étendre les mécanismes proposés au cas avec variables libres et avec faisabilités.

8.2 Un premier algorithme de recherche arborescente structurée

Le premier algorithme que nous définissons correspond à un parcours du MCDAG de la racine vers les feuilles (alors qu'un algorithme d'élimination de variables ou de clusters propage de l'information des feuilles du MCDAG vers la racine). Nous introduisons tout d'abord une définition essentielle pour la compréhension du reste de ce chapitre.

Définition 8.1. *Soit c un cluster du MCDAG. Soit $V \subset V(c) - V(pa(c))$ un sous-ensemble des variables à éliminer dans c. Soit $\Phi \subset \Phi(c)$ un sous-ensemble des fonctions locales associées à c. Soit A une affectation des variables de $V(c) - V$ et des variables impliquées dans les ascendants de c dans le MCDAG. Nous définissons $val(c, A, V, \Phi)$ par*

$$val(c, A, V, \Phi) = \underset{V}{\oplus^c} \left(\left(\underset{\varphi \in \Phi}{\otimes^c} \varphi(A) \right) \otimes^c \left(\underset{s \in Sons(c)}{\otimes^c} val(s)(A) \right) \right)$$

avec $val(s)(A)$ la valeur donnée par la définition 7.8 page 73.

En d'autres termes, $val(c, A, V, \Phi)$ correspond à l'élimination des variables de V sur la combinaison des fonctions locales dans Φ et des valeurs des clusters fils de c. Les valeurs sont prises pour

une affectation A et les opérateurs d'élimination et de combinaison utilisés sont les opérateurs $\oplus^c$ et $\otimes^c$ du cluster c.

Proposition 8.2. *Soit M un MCDAG associé à une requête Q. Soit c un cluster de M.*

(a) Soit r le cluster racine de M. Alors, $Ans(Q) = val(r, \emptyset, V(r), \Phi(r))$.

(b) $\forall x \in V$, $val(c, A, V, \Phi) = \oplus^c{}_{a \in dom(x)} ((\otimes^c{}_{\varphi \in \Phi_0} \varphi(A.(x, a))) \otimes^c val(c, A.(x, a), V - \{x\}, \Phi - \Phi_0))$, avec $\Phi_0 = \{\varphi \in \Phi \mid sc(\varphi) \cap (V - \{x\}) = \emptyset\}$

(c) $val(c, A, \emptyset, \Phi) = (\otimes^c{}_{\varphi \in \Phi} \varphi(A)) \otimes^c \left(\otimes^c{}_{s \in Sons(c)} val(s, A, V(s) - V(c), \Phi(s))\right)$

La proposition 8.2 permet de définir directement un algorithme de recherche arborescente structurée. Plus précisément, la proposition 8.2(a) montre que pour calculer la réponse $Ans(Q)$ à une requête Q associée à un MCDAG ayant pour racine r, il suffit de calculer $val(r, \emptyset, V(r), \Phi(r))$. Une utilisation récursive de la proposition 8.2(b) donne alors une méthode pour calculer cette quantité $val(r, \emptyset, V(r), \Phi(r))$, en affectant récursivement les variables de $V(r)$. Dès que toutes les variables de $V(r)$ sont affectées, des quantités telles que $val(r, A, \emptyset, \Phi)$ doivent être calculées. La proposition 8.2(c), nous permet alors d'écrire

$$val(r, A, \emptyset, \Phi) = (\otimes^c{}_{\varphi \in \Phi} \varphi(A)) \otimes^c \left(\otimes^c{}_{s \in Sons(r)} val(s, A, V(s) - V(r), \Phi(s))\right).$$

Chaque $val(s, A, V(s) - V(r), \Phi(s))$ peut ensuite être calculé en utilisant à nouveau la proposition 8.2(b). Ainsi, une application récursive et alternée des propositions 8.2(b) et 8.2(c) permet de calculer $Ans(Q)$.

L'algorithme associé au mécanisme précédemment décrit, nommé **TS-mcdag** (comme "Tree Search on MCDAG"), est donné à la figure 8.1. Comme il utilise la structure du problème, cet algorithme sera probablement plus performant que l'algorithme de recherche arborescente non structurée donné à la section 6.1.

```
TS-mcdag(c, A, V, Φ)
début
    si (V = ∅) alors
        S ← Sons(c)
        val ← ⊗^c_{φ∈Φ} φ(A)
        tant que S ≠ ∅ faire
            choisir s ∈ S
            S ← S − {s}
            val ← val ⊗^c TS-mcdag(s, A, V(s) − V(c), Φ(s))
        retourner (val)
    sinon
        choisir x ∈ V
        d ← dom(x)
        Φ0 ← {φ ∈ Φ(c) , sc(φ) ∩ (V − {x}) = ∅}
        val ← ◊
        tant que d ≠ ∅ faire
            choisir a ∈ d
            d ← d − {a}
            val ← val ⊕^c(⊗^c_{φ∈Φ0} φ(A.(x, a))) ⊗^c TS-mcdag(c, A.(x, a), V − {x}, Φ − Φ0)
        retourner (val)
fin
```

Figure 8.1: Un algorithme de recherche arborescente structurée sur un MCDAG.

La premier appel est **TS-mcdag**$(r, \emptyset, V(r), \Phi(r))$. Il est en fait possible de montrer que **TS-mcdag**(c, A, V, Φ) calcule la quantité $val(c, A, V, \Phi)$.

Proposition 8.3. *L'algorithme **TS-mcdag** est correct et complet, c'est-à-dire qu'il renvoie $Ans(Q)$.*

Proposition 8.4. *Soit M un MCDAG associé à une requête $Q = (Sov, (V, G, P, \emptyset, U))$. La complexité spatiale de l'algorithme **TS-mcdag** est $O(h \cdot (d + m))$ et sa complexité spatiale est*

$$O(m \cdot \mu \cdot d^h),$$

avec d la taille maximale des domaines des variables, h la hauteur du MCDAG, μ le nombre maximal de parents pour un cluster du MCDAG ($\mu = 1$ si le MCDAG est un arbre) et $m = |P \cup U|$ dans le cas semi-anneau et $m = (1 + |P|)(1 + |U|)$ dans le cas semi-groupe.

La proposition 8.4 montre que la largeur induite n'est pas le seul critère permettant de juger la qualité d'un MCDAG : la hauteur peut aussi être un critère de choix de MCDAG.

8.3 Ajout de techniques de mémorisation

L'algorithme **TS-mcdag** peut être amené à faire de nombreuses fois le même calcul. Mémoriser le résultat de certaines opérations permet d'éviter de telles redondances dans les calculs, en troquant de la complexité spatiale pour un gain de complexité temporelle. Le point clé est qu'étant donnés un cluster c et un fils s de c, l'algorithme précédent calcule la valeur $val(s)(A)$ du fils s pour de nombreuses affectations A, alors que certaines affectations mènent nécessairement au même résultat du fait de la structure du problème. La raison est que $val(s)(A) = val(s)(A')$ dès que A et A' coïncident sur le séparateur entre c et s, c'est-à-dire dès que $A^{\downarrow c \cap s} = A'^{\downarrow c \cap s}$. Afin d'éviter ces calculs redondants, nous introduisons une structure de mémorisation : la valeur mémorisée pour $val(s)(A^{\downarrow c \cap s})$ est notée $rec(s, A^{\downarrow c \cap s})$ et vaut **nil** si aucune valeur n'est mémorisée pour $val(s)(A^{\downarrow c \cap s})$. L'algorithme actualisé, appelé **RecTS-mcdag** comme *TS-mcdag with Recording*, est donné à la figure 8.2. Par rapport à l'algorithme précédent, seule une ligne est ajoutée.

Proposition 8.5. *L'algorithme **RecTS-mcdag** est correct et complet, c'est-à-dire qu'il renvoie $Ans(Q)$.*

Proposition 8.6. *Soit M un MCDAG associé à une requête $Q = (Sov, (V, G, P, F, U))$. Soit w la largeur du MCDAG. Calculer $Ans(Q)$ avec l'algorithme **RecTS-mcdag** sur le MCDAG M est $O(m \cdot d^{w+1})$ en temps, avec $m = |P \cup U|$ dans le cas semi-anneau et $m = (1 + |P|) \cdot (1 + |U|)$ dans le cas semi-groupe. La complexité spatiale est $O(N \cdot s \cdot d^s)$ avec N le nombre de clusters dans le MCDAG et s la taille du plus grand séparateur.*

Notons également que l'algorithme **RecTS-mcdag** peut être facilement transformé en une version *any-space*, dans laquelle certaines mémorisations peuvent être oubliées lorsque la mémoire disponible devient trop faible.

8.4 Utilisation de bornes

Un des intérêts de la recherche arborescente est de pouvoir élaguer certaines branches de l'espace de recherche. Cet élagage induit généralement un gain important en termes de complexités

```
RecTS-mcdag(c, A, V, Φ)
début
  si (V = ∅) alors
    S ← Sons(c)
    val ← ⊗^c_{φ∈Φ} φ(A)
    tant que S ≠ ∅ faire
      choisir s ∈ S
      S ← S − {s}
      si (rec(s, A^{↓c∩s}) = nil) alors rec(s, A^{↓c∩s}) ← RecTS-mcdag(s, A, V(s) − V(c), Φ(s))
      val ← val ⊗^c rec(s, A^{↓c∩s})
    retourner (val)
  sinon
    choisir x ∈ V
    d ← dom(x)
    Φ_0 ← {φ ∈ Φ , sc(φ) ∩ (V − {x}) = ∅}
    val ← ◊
    tant que d ≠ ∅ faire
      choisir a ∈ d
      d ← d − {a}
      val ← val ⊕^c (⊗^c_{φ∈Φ_0} φ(A.(x, a))) ⊗^c RecTS-mcdag(c, A.(x, a), V − {x}, Φ − Φ_0)
    retourner (val)
fin
```

Figure 8.2: Algorithme de recherche arborescente structurée utilisant de la mémorisation.

temporelle et spatiale en pratique car il permet d'éviter d'avoir à considérer des parties entières de l'espace de recherche.

Nous supposons dans ce qui suit que le MCS totalement ordonné $(E, \oplus, \otimes)$ admet un élément minimum $\bot$ égal à 0_E et un élément maximum $\top$. Cette hypothèse supplémentaire est gratuite dès que le MCS considéré satisfait l'axiome $(x \otimes y = 0_E) \rightarrow ((x = 0_E) \vee (y = 0_E))$, qui est satisfait par toutes les structures classiques d'utilité espérée.

8.4.1 Utilisation de bornes en présence de plusieurs opérateurs d'élimination

Une première difficulté dans l'adaptation des techniques de branch-and-bound sur les MCDAGs réside dans la présence de plusieurs opérateurs d'élimination. Pour remédier à cette difficulté, nous adaptons des idées venant de l'algorithme alpha-beta [55] utilisé en théorie des jeux (cet algorithme a d'ailleurs été étendu au cas des jeux stochastiques [4] où les opérateurs d'élimination min, max et + peuvent alterner).

L'adaptation des techniques alpha-beta nous conduit à utiliser une borne inférieur LB et une borne supérieure UB qui spécifient que la résultat d'un calcul local doit être compris strictement entre LB et UB pour être intéressant du point de vue du calcul global. Informellement, les clusters utilisant max comme opérateur d'élimination renforceront l'exigence donnée par LB (de manière à toujours chercher un résultat meilleur que le meilleur résultat connu) et les clusters utilisant min comme opérateur d'élimination renforceront l'exigence donnée par UB (de manière à toujours chercher un résultat pire que le pire résultat connu). Nous utilisons également deux valeurs spéciales $\bot^-$ et $\top^+$, avec $\bot^-$ plus petit que tout élément de E et $\top^+$ plus grand que tout élément de $E \cup \{\bot^-\}$. Imposer les exigences $LB = \bot^-$ et $UB = \top^+$ signifie qu'aucune exigence n'est imposée

sur le résultat cherché.

8.4.2 Utilisation de bornes sans inverse pour les opérateurs de combinaison

Une seconde difficulté réside dans le fait que la structure algébrique de MCS totalement ordonné ne permet pas de supposer l'existence d'un opérateur de différence $\oslash$ inverse de $\otimes$ ou l'existence d'un opérateur de différence $\ominus$ inverse de $\oplus$.

Du fait de la factorisation en fonctions locales, il se peut que l'on souhaite imposer des exigences telles que $e_\otimes \otimes val \prec UB$ sur une valeur val à calculer, avec $e_\otimes$ un facteur qui doit être combiné avec val en utilisant $\otimes$. Comme nous ne supposons pas l'existence d'un opérateur de division $\oslash$, nous ne pouvons pas directement imposer une exigence du type $val \prec UB \oslash e_\otimes$ et prendre $UB' = UB \oslash e_\otimes$ comme nouvelle borne supérieure imposée sur val.

La même remarque s'applique pour des exigences du type $val \oplus e_\oplus \prec UB$ avec $e_\oplus$ un facteur à combiner avec val en utilisant $\oplus$, car nous ne supposons pas l'existence d'un opérateur de différence $\ominus$ inverse de $\oplus$.

C'est pourquoi nous allons avoir besoin d'imposer des exigences complexes de la forme $e_\otimes \otimes val \oplus e_\oplus \prec UB$ ou $LB \prec e_\otimes \otimes val \oplus e_\oplus$. De plus, les facteurs $e_\otimes$ et $e_\oplus$ peuvent eux-mêmes ne pas être connus précisément : il se peut que seulement une borne inférieure $lb_\otimes$ et une borne supérieure $ub_\otimes$ sur $e_\otimes$ soient disponibles, et que seulement une borne inférieure $lb_\oplus$ et une borne supérieure $ub_\oplus$ sur $e_\oplus$ soient disponibles. Afin de manipuler uniquement des exigences à une inconnue, nous aurons besoin d'imposer des exigences de la forme :

$$(LB \prec ub_\otimes \otimes val \oplus ub_\oplus) \wedge (lb_\otimes \otimes val \oplus lb_\oplus \prec UB) \tag{8.1}$$

Ceci nous conduit à définir la notion de *borne complexe*.

Définition 8.7. *Une borne complexe est un sextuplet* $(LB, UB, lb_\otimes, ub_\otimes, lb_\oplus, ub_\oplus)$ *tel que* $LB \prec UB$, $lb_\otimes \preceq ub_\otimes$ *et* $lb_\oplus \preceq ub_\oplus$.

Informellement, imposer une borne complexe $(LB, UB, lb_\otimes, ub_\otimes, lb_\oplus, ub_\oplus)$ sur une quantité val à calculer signifie imposer l'équation 8.1. Grâce aux bornes complexes, certaines branches de l'espace de recherche vont pouvoir être coupées. Ainsi, si une branche de l'espace de recherche est chargée de calculer la valeur $val(c, A, V, \Phi)$ tout en satisfaisant une exigence complexe $\mathcal{B}$, alors la valeur exacte de $val(c, A, V, \Phi)$ n'est pas requise s'il est prouvé que l'exigence complexe $\mathcal{B}$ est nécessairement violée. Afin de représenter cela, nous introduisons la notion d'*évaluation bornée*.

Définition 8.8. *Soit* $\mathcal{B} = (LB, UB, lb_\otimes, ub_\otimes, lb_\oplus, ub_\oplus)$ *une borne complexe. Une évalution de* $val(c, A, V, \Phi)$ *bornée par* $\mathcal{B}$ *est un couple* $(lb, ub) \in E^2$ *tel que* $lb \preceq val(c, A, V, \Phi) \preceq ub$ *et tel que* $(lb = ub) \vee (lb_\otimes \otimes lb \oplus lb_\oplus = ub_\otimes \otimes ub \oplus ub_\oplus) \vee (LB \succeq ub_\otimes \otimes ub \oplus ub_\oplus) \vee (UB \preceq lb_\otimes \otimes lb \oplus lb_\oplus)$.

En d'autres termes, une évaluation de $val(c, A, V, \Phi)$ bornée par $\mathcal{B}$ doit fournir des bornes inférieures et supérieures lb et ub sur $val(c, A, V, \Phi)$ telles que l'une des conditions suivantes est satisfaite

1. $lb = ub$, i.e. la valeur exacte de $val(c, A, V, \Phi)$ est connue ;

2. $lb_\otimes \otimes lb \oplus lb_\oplus = ub_\otimes \otimes ub \oplus ub_\oplus$: dans ce cas, quelle que soit la valeur exacte de $val(c, A, V, \Phi)$, nous aurons $e_\otimes \otimes val(c, A, V, \Phi) \oplus e_\oplus = lb_\otimes \otimes lb \oplus lb_\oplus = ub_\otimes \otimes ub \oplus ub_\oplus$. Cela signifie que quelle que soit la valeur exacte de $val(c, A, V, \Phi)$, lb et ub garantissent qu'un unique degré global sera obtenu après combinaison avec le reste du problème ;
3. $LB \succeq ub_\otimes \otimes ub \oplus ub_\oplus$, i.e. la borne supérieure ub prouve que $val(c, A, V, \Phi)$ ne satisfait pas les exigences imposées par $\mathcal{B}$;
4. $UB \preceq lb_\otimes \otimes lb \oplus lb_\oplus$, i.e. la borne inférieure lb prouve que $val(c, A, V, \Phi)$ ne satisfait pas les exigences imposées par $\mathcal{B}$.

8.4.3 Définition de l'algorithme

Partant de là, il est possible d'ajouter l'utilisation de bornes dans un algorithme de recherche arborescente structurée. Cet algorithme utilise plusieurs fonctions :

— Une fonction $bound(c, A, V, \Phi)$, qui renvoie une borne inférieure lb et une borne supérieure ub sur $val(c, A, V, \Phi)$.

— Trois fonctions nommées $evalClusterMin(c, A, V, \Phi, \mathcal{B})$, $evalClusterMax(c, A, V, \Phi, \mathcal{B})$ et $evalClusterPlus(c, A, V, \Phi, \mathcal{B})$ qui calculent, pour un cluster éliminant les variables en utilisant min, max et $\oplus$ respectivement, une évaluation de $val(c, A, V, \Phi)$ bornée par la borne complexe $\mathcal{B}$. Lorsque l'ensemble V des variables à éliminer est vide, cette fonction fait appel à la fonction d'évaluation des fils de c (cf. ci-dessous).

— Une fonction $evalSons(c, A, \Phi, \mathcal{B})$, qui calcule une évaluation de $val(c, A, \emptyset, \Phi)$ bornée par $\mathcal{B}$. Cette fonction calcule une combinaison de la valeur des fils s de c. La valeur de chacun des fils (ou une évaluation de cette valeur) est obtenue en utilisant une des trois fonctions ci-dessus, en fonction de l'opérateur d'élimination utilisé par s.

— Une fonction principale, appelée $BTD\text{-}mcdag()$, qui renvoie la réponse $Ans(Q)$ à une requête Q. Si r correspond à la racine du MCDAG considéré, cette fonction appelle simplement $evalSons(r, \emptyset, \Phi(r), \mathcal{B}_0)$ avec $\mathcal{B}_0 = (\bot^-, \top^+, 1_E, 1_E, 0_E, 0_E)$ une borne complexe "vide" qui n'impose aucune exigence.

La définition formelle de ces fonctions est laissée à la version anglaise de la thèse. Notons que du fait de l'élagage de l'espace de recherche, il se peut que l'on "rentre" dans un cluster c et que l'on en ressorte sans connaître la valeur exacte de c. De ce fait, la structure de mémorisation utilisée n'enregistre pas la valeur exacte d'un cluster pour une affectation donnée, mais seulement des bornes inférieures et supérieures sur cette valeur. La valeur exacte du cluster est connue lorsque ces bornes inférieures et supérieures sont égales. Notons également que la structure utilisée pour la mémorisation peut être creuse : la structure de mémorisation peut être une table de hachage, une structure du type diagramme de décision binaire [1, 17]... et non forcément une table.

Théorème 8.9. *Si la fonction* bound *est correcte et complète, alors l'algorithme* ***BTD-mcdag*** *est correct et complet, c'est-à-dire qu'il renvoie* $Ans(Q)$.

Proposition 8.10. *Soit* M *un MCDAG associé à une requête* $Q = (Sov, (V, G, P, \emptyset, U))$. *La complexité temporelle de l'algorithme* ***BTD-mcdag*** *est* $O(m \cdot \mu \cdot d^h)$, *avec* h *la hauteur du MCDAG,* μ *le nombre maximal de parents pour un cluster du MCDAG (*$\mu = 1$ *si le MCDAG est un arbre) et* $m = |P \cup U|$ *dans le cas semi-anneau et* $m = (1 + |P|)(1 + |U|)$ *dans le cas semi-goupe. La*

complexité spatiale est $O(N \cdot s \cdot d^s)$, avec N le nombre de clusters dans le MCDAG et s la taille maximale des séparateurs.

La complexité théorique de l'algorithme **BTD-mcdag** est pire que celle de l'algorithme **RecTS-mcdag** et les deux algorithmes ont la même complexité spatiale. Cependant, cela ne signifie pas que l'algorithme **RecTS-mcdag** est meilleur que l'algorithme **BTD-mcdag** en pratique, car les complexités fournies ici sont uniquement théoriques. En pratique, une recherche arborescente qui utilise des bornes est bien plus efficace qu'une recherche sans bornes malgré une moins bonne complexité théorique.

L'algorithme ainsi défini est applicable pour résoudre des problèmes aussi variés que des problèmes de satisfiabilité stochastique, des CSP stochastiques, des QBF, des QCSP, des MDP factorisés, des diagrammes d'influence probabilistes ou possibilistes, des problèmes MAP (Maximum A Posteriori hypothesis) ou encore des problèmes de planification stochastique ou de conformant planning. Ceci montre l'intérêt de définir des algorithmes génériques dans un cadre générique.

8.5 Utilisation d'opérateurs de différence et de division

Les bornes complexes permettent de définir un algorithme de recherche arborescente structurée utilisant des bornes. Cependant, utiliser des bornes complexes n'est pas gratuit, car par exemple pour chaque test impliquant LB une opération $\otimes$ et une opération $\oplus$ sont réalisées, en plus du test de comparaison pour savoir si LB est satisfaite. Lorsque la structure algébrique est plus riche, il est possible de revenir à des bornes simples (LB, UB) imposant des exigences simples du type $(LB \prec ub) \wedge (lb \prec UB)$, au lieu d'utiliser des bornes complexes $(LB, UB, lb_\otimes, ub_\otimes, lb_\oplus, ub_\oplus)$ et des comparaisons complexes telles que $(LB \prec ub_\otimes \otimes ub \oplus ub_\oplus) \wedge (lb_\otimes \otimes lb \oplus lb_\oplus \prec UB)$.

Les hypothèses algébriques supplémentaires qui permettent d'utiliser des bornes simples sont essentiellement liées à l'existence d'opérations inverses pour $\otimes$ et $\oplus$. Certaines de ces hypothèses se rapprochent des hypothèses faites dans [20] pour les *fair* VCSP ou dans [8] pour les *semiring-based CSP*. Ces hypothèses supplémentaires s'écrivent de la manière suivante :

— Hypothèse supplémentaire sur $\oplus$, notée "$Ax^{\ominus}$"
  Pour tous $x, y \in E$ tels que $x \preceq y$, l'ensemble $\{z \in E \mid y = z \oplus x\}$ admet un élément maximum noté $y \ominus x$. En d'autres termes, nous supposons l'existence d'une différence maximale entre y et x.

— Hypothèse supplémentaire sur $\otimes$, notée "$Ax^{\oslash}$", avec deux versions disjointes
  – $Ax_1^{\oslash}$: ou bien $1_E = \top$ et pour tous $x, y \in E$ tels que $x \preceq y$, l'ensemble $\{z \in E \mid x = z \otimes y\}$ admet un élément maximum noté $x \oslash y$ (c'est-à-dire qu'il existe une différence maximale entre x et y).
  – $Ax_2^{\oslash}$: ou bien $1_E \neq \top$ et $\top = \top^+$ (ce qui signifie que $\top$ a été ajouté à la structure de départ) et pour tous $x, y \in E$ tels que $y \notin \{0_E, \top\}$, il existe un unique $z \in E$, noté $x \oslash y$, tel que $x = y \otimes z$.

Ces axiomes sont satisfaits dans de nombreux cas classiques. Par exemple, l'axiome $Ax^{\ominus}$ est satisfait pour $(E, \preceq, \oplus) = ([0, +\infty], \leq, +)$, $(E, \preceq, \oplus) = ([0, +\infty], \geq, \min)$ et $(E, \preceq, \oplus) = ([0, 1], \leq, \max)$. L'hypothèse supplémentaire sur $\otimes$ est satisfaite avec $(E, \preceq, \otimes) = ([0, +\infty], \geq, +)$ et $(E, \preceq, \otimes) = ([0, 1], \leq, \min)$ dans le premier cas ($1_E = \top$) et avec $(E, \preceq, \otimes) = ([0, +\infty], \leq, \times)$ dans le

second cas ($1_E \neq \top$).

Comme indiqué dans le tableau 8.1, dès que $Ax^{\ominus}$ et $Ax^{\oslash}$ sont vérifiés, il est possible d'utiliser des bornes simples. Ce tableau montre que si *val* est une quantité à calculer, des exigences telles que $\alpha \oplus val \prec UB$, $\alpha \oplus val \succ LB$, $\alpha \otimes val \prec UB$ ou $\alpha \otimes val \succ LB$ peuvent être transformées en des exigences pour lesquelles il suffit de comparer *val* avec une borne inférieure actualisée LB' ou avec une borne supérieure actualisée UB'.

Cas	Exigence complexe	Condition	Exigence simple
$Ax^{\ominus}$	$\alpha \oplus val \prec UB$	$UB \preceq \alpha$	$val \prec \bot^-$
		$\alpha \prec UB$	$val \prec UB \ominus \alpha$
	$LB \prec \alpha \oplus val$	$LB \prec \alpha$	$val \succ \bot^-$
		$\alpha \preceq LB$	$val \succ LB \ominus \alpha$
$Ax_1^{\oslash}$	$\alpha \otimes val \prec UB$	$UB \preceq \alpha$	$val \prec UB \oslash \alpha$
		$\alpha \prec UB$	$val \prec \top^+$
	$LB \prec \alpha \otimes val$	$LB \prec \alpha$	$val \succ LB \oslash \alpha$
		$\alpha \preceq LB$	$val \succ \top^+$
$Ax_2^{\oslash}$	$\alpha \otimes val \prec UB$	$\alpha \notin \{0_E, \top\}$	$val \prec UB \oslash \alpha$
		$\alpha = 0_E$	$val \prec \top^+$
	$LB \prec \alpha \otimes val$	$\alpha \notin \{0_E, \top\}$	$val \succ LB \oslash \alpha$
		$(\alpha = 0_E) \wedge (LB \neq \bot^-)$	$val \succ \top^+$
		$(\alpha = 0_E) \wedge (LB = \bot^-)$	$val \succ \bot^-$
		$\alpha = \top$	$val \succ \bot^-$

TABLE 8.1 – De bornes complexes à des bornes simples en utilisant des opérations de différence et de division, avec $(\alpha, val) \in E^2$ et $LB \prec UB$. Pour $Ax_2^{\oslash}$, le cas de l'exigence $\alpha \otimes val \prec UB$ combinée avec $\alpha = \top$ n'est pas considéŕe car il n'apparaîtra jamais en pratique (informellement, lorsqu'une exigence de la forme $\alpha \otimes val \prec UB$ sera imposée, α correspondra à une borne inférieure sur une quantité dans $E - \{\top\}$, et donc $\alpha \neq \top$).

De nouvelles versions des fonctions *evalClusterMin*, *evalClusterMax*, *evalClusterPlus* et *evalSons* peuvent alors être définies. Ces nouvelles versions utilisent uniquement des bornes simples (LB, UB) et doivent calculer des évaluations de quantités $val(c, A, V, \Phi)$ bornées par des bornes simples, comme défini ci-dessous.

Définition 8.11. *Une évaluation de* $val(c, A, V, \Phi)$ *bornée par une borne simple* (LB, UB) *est un couple* $(lb, ub) \in E^2$ *tel que* $lb \preceq val(c, A, V, \Phi) \preceq ub$ *et* $(lb = ub) \vee (UB \preceq lb) \vee (ub \preceq LB)$.

En d'autres termes, une évaluation de $val(c, A, V, \Phi)$ bornée par (LB, UB) est tout simplement une couple de bornes inférieures et supérieures sur $val(c, A, V, \Phi)$ qui soit donnent la valeur exacte de $val(c, A, V, \Phi)$, soit prouvent qu'une des exigences imposées par LB et UB est violée.

La nouvelle fonction principale est appelée **BTD-answerQ**().

8.6 Calcul de bornes

Les algorithmes précédemment introduits utilisent une fonction nommée *bound* qui permet de calculer des bornes sur certaines valeurs. Afin de ne pas retourner des bornes naïves telles que $(\bot, \top)$, de nombreuses techniques peuvent être utilisées :

— *Calcul de bornes par propagation de contraintes* [63, 13, 63, 10, 20, 59].

— *Inversion de quantificateurs* : inverser certains quantificateurs peut donner une borne inférieure ou supérieure et le calcul à réaliser après inversion de quantificateurs peut être très simple en termes de largeur induite, même si la quantité à calculer avant inversion est compliquée à calculer en termes de largeur.

— *Changement de quantificateurs* : une autre technique peut être de remplacer des min par des max, des max par des min, des $\oplus$ par des max... Pourvu que les opérations de remplacement soit diminuent toujours la quantité à calculer, soit l'augmentent toujours, la quantité obtenue après changement de quantificateurs peut être très facile à calculer et fournir une borne inférieure ou supérieure. Ce type de techniques a déjà été utilisé dans [92] sur des QBF : dans ce travail, l'idée consiste à remplacer des quantificateurs $\forall$ par des quantificateurs $\exists$ pour obtenir une forme ne faisant intervenir que des quantificateurs $\exists$, et donc pour laquelle aucune contrainte sur l'ordre d'élimination n'est imposée (d'où un calcul potentiellement beaucoup plus simple).

— *Mini-buckets* [33] : l'idée des techniques *mini-bucket* est de conserver des fonctions locales dont la portée a une taille inférieure à un certain seuil. La largeur induite est ainsi contrôlée et même si le résultat obtenu n'est pas exact, il peut fournir des bornes sur la quantité à calculer.

— *Simplification de la structure algébrique* : on peut travailler sur une formulation du problème utilisant une structure algébrique plus simple, résoudre ce problème plus simple et enfin transformer le résultat obtenu en bornes sur le problème initial [7, 24]

8.7 Résumé et perspectives

Ce chapitre a montré comment un algorithme générique de recherche arborescente structurée utilisant des bornes pouvait être défini pour calculer la valeur d'un MCDAG. Les points centraux sont la gestion de l'aspect multi-opérateur pour les combinaisons et les éliminations et la gestion des bornes élaguant l'espace de recherche. Des résultats de complexité théorique ont également été établis en fonction de paramètres divers que sont la largeur du MCDAG, sa hauteur, ou encore la taille maximale d'un séparateur entre deux clusters.

D'un autre côté, on pourrait envisager de définir des algorithmes de résolution approchée utilisant des techniques d'échantillonnage [66, 87] ou de recherche locale [67] : échantillonnage pour des éliminations avec l'opérateur + (+, et non $\oplus$) et recherche locale pour des éliminations par les opérateurs min et max. Ceci est une des voies à explorer pour définir de nouveaux algorithmes génériques.

D'un point de vue plus pragmatique, les algorithmes définis dans ce chapitre font en fait intervenir plusieurs paramètres dont l'influence reste à étudier :

— Heuristiques pour le choix de la prochaine variable à affecter dans le cluster courant, pour le choix de la valeur à affecter à une variable, pour le choix du prochain cluster fils à considérer...

— Calcul de bornes : quelques pistes ont été données concernant le calcul de bornes, mais beaucoup de travail reste à accomplir pour trouver de bons réglages (par exemple concernant le degré de cohérence locale à établir lors d'une propagation de contraintes).

Beaucoup d'études sur tous ces paramètres ont déjà été réalisées dans chacun des formalismes

couverts par le cadre PFU. Afin d'acquérir une meilleure connaissance concernant leur "influence générique" et afin de tester l'efficacité pratique des algorithmes définis, il est nécessaire d'obtenir des résultats expérimentaux. C'est pourquoi un outil de résolution générique permettant de répondre à des requêtes PFU génériques a été développé.

Chapitre 9

Un outil générique pour répondre à des requêtes PFU

Ce chapitre décrit brièvement l'outil de résolution générique qui a été implémenté pour résoudre des requêtes sur des réseaux PFU. Il présente d'abord les formats utilisés pour décrire des réseaux PFU et des requêtes, avant de présenter l'outil générique développé. Le but principal de ce chapitre est de montrer que le cadre PFU n'est pas juste une abstraction. Voir la version anglaise de la thèse pour plus de détails.

Quelques résultats expérimentaux préliminaires ont été obtenus sur un problème réel de déploiement et de maintien d'une constellation de satellites, mais poursuivre les expérimentations sur d'autres problèmes est nécessaire pour approfondir les points suivants :

- Comparer les algorithmes définis précédemment en termes de complexité pratique :
  - quantifier les gains obtenus grâce aux techniques de structuration des requêtes (qui fournissant l'architecture MCDAG),
  - comparer les algorithmes d'élimination de variable avec les méthodes de recherche arborescente structurée,
  - comparer les algorithmes de recherche arborescente structurée pour plusieurs réglages : mémorisation ou non, bornes simples ou bornes complexes, heuristiques pour les choix de variables, de valeurs ou de cluster fils à explorer, techniques utilisées pour calculer des bornes (cohérence locale souple, inversion de quantificateurs...)
- Comparer les algorithmes implémentés avec des algorithmes développés dans des cadres spécifiques
- Evaluer la complexité induite par l'utilisation d'une structure d'utilité espérée donnée, pour quantifier par exemple le coût engendré par l'utilisation de modèles de plausibilités/utilités plus ou moins qualitatifs ou quantitatifs (par exemple, pour une même forme de réseau PFU et pour une même requête, comparer les temps de calcul obtenus si l'on utilise un modèle d'utilité espérée additive probabiliste, un modèle de satisfation espérée, un modèle d'utilité espérée possibiliste...).

Conclusion

Synthèse des contributions

Au cours des dernières décennies, de nombreux formalismes ont été définis pour modéliser et résoudre des problèmes de décision. Dans cette thèse, nous avons introduit un nouveau cadre de représentation générique pour des problèmes de décision séquentielle mettant en jeu des incertitudes, des infaisabilités et des utilités. Ce cadre flexible couvre non seulement de nombreuses approches existantes, telles que les CSP durs, valués, quantifiés, mixtes et stochastiques, les réseaux bayésiens, les champs de Markov, les processus décisionnels markoviens probabilistes ou possibilistes, ou encore les diagrammes d'influence, mais il permet aussi de définir directement de nouveaux formalismes correspondant à certaines de ses instanciations.

Au final, le cadre obtenu, appelé le cadre PFU, est à la fois algébrique et justifié d'un point de vue de la théorie de la décision. Ces deux facettes, qui sont le résultat d'une conception prenant en compte à la fois des enjeux en termes d'expressivité et des enjeux algorithmiques, apparaissent clairement dans le théorème 5.4, qui établit une équivalence formelle entre d'une part la définition de la réponse à une requête à partir d'arbre de décision et d'autre part la définition opérationnelle de la valeur d'une requête. Comparée aux approches algébriques génériques existantes [97, 25, 56], le cadre PFU est le seul qui manipule explicitement plusieurs types de variables (variables de décision et variables d'environnement), plusieurs types de fonctions locales (plausibilités, faisabilités et utilités) et plusieurs types d'opérateurs de combinaison et d'élimination.

D'un point de vue opérationnel, des algorithmes génériques à base de recherche arborescente ou d'élimination de variables ont également été définis. Certaines conditions dites de décomposabilité, permettant d'utiliser toutes les factorisations de quantités globales en fonctions locales, ont été identifiées et utilisées au sein d'un algorithme unifié d'élimination de variables (utilisant éventuellement des éléments appelés potentiels). Une autre voie a ensuite été explorée avec des mécanismes d'optimisation de la structure d'une requête. Cette voie nous a conduit à définir deux nouvelles architectures de calcul permettant de répondre à des requêtes de manière plus efficace, l'architecture des *arbre de clusters multi-opérateurs* (MCTree) et l'architecture des *DAGs de clusters multi-opérateurs* (MCDAG). Ces architectures ont été construites en utilisant un mécanisme de structuration en deux temps utilisant d'une part des règles de réécriture et d'autre part des techniques de décomposition en arbres de clusters. Cette structuration permet d'améliorer un paramètre connu sous le nom de largeur induite (ou largeur d'arbre) qui influence les complexités spatiales et temporelles de résolution d'une requête. A partir de ces architectures, nous avons défini des algorithmes de recherche arborescente plus ou moins sophistiqués suivant s'ils utilisent des techniques de mémorisation ou d'élagage par des bornes. La principale difficulté a été de réussir à gérer la

nature multi-opérateur des requêtes PFU, à la fois en terme d'élimination et de combinaison. Naturellement, certaines hypothèses faites par le cadre PFU sont discutables. Mais il est nécessaire de garder à l'esprit que ces mêmes hypothèses ont permis d'explorer des approches algorithmiques variées. Enfin, un outil générique de résolution a été implémenté.

Toutes ces contributions sont résumées à la figure 9.1

D'un point de vue plus général, les conclusions de cette thèse sont les suivantes :

1. Construire un cadre générique englobant de nombreux formalismes existants est possible, et le cadre obtenu n'est pas un monstre incompréhensible et ingérable. Il correspond juste à une forme générique de *modèle graphique algébrique composite*.
2. Des algorithmes génériques unifiés peuvent être définis dans ce cadre et, comme pour les approches algébriques existantes, ces algorithmes offrent des complexité théoriques spatiales et temporelles fonctions de paramètres topologiques tels que la largeur d'arbre. En termes de largeur, une analyse précise de la structure des requêtes multi-opérateur considérées peut être fortement bénéfique.
3. Répondre à des requêtes multi-opérateurs est équivalent à répondre à plusieurs requêtes mono-opérateurs organisées dans une architecture générique appelée l'architecture MCDAG. Cette dernière peut être obtenue de manière systématique et dès qu'elle est disponible, les algorithmes de résolution du cas mono-opérateur peuvent être réutilisés. La principale difficulté concerne la gestion des bornes, qui doivent être utilisés en synergie avec les multiples opérateurs de combinaison et d'élimination. En réalité, les MCDAGs peuvent être utilisés quelle que soit la méthode de résolution utilisée (élimination de variables, recherche arborescente ou algorithmes approchés), car ils expriment juste des décompositions.

Perspectives

Les perspectives de ce travail sont multiples :

— Comme indiqué au chapitre 9, obtenir des résultats expérimentaux est l'un des objectifs à court terme, ce pour pouvoir obtenir une meilleure connaissance concernant les algorithmes proposés.

— Les méthodes de structuration raisonnent en fonction des variables. Nous pourrions aussi tenter d'exploiter des structures présentes à des niveaux plus fins, par example au niveau de la valeur des fonctions locales, en utilisant des approches telles que les diagrammes de décision binaires (*Binary Decision Diagrams*, BDDs [1, 17]) ou les *Negation Normal Forms* (NNFs [23]).

— Nous pourrions également étudier plus précisément les résultats fournis par les méthodes de structuration pour des requêtes et des réseaux répliqués sur plusieurs pas de temps, comme dans un processus décisionnel markovien.

— Beaucoup de choses restent à faire concernant l'utilisation de bornes, une des idées maîtresses pouvant être de définir une sorte d'*arc cohérence quantifiée généralisée*.

— D'un point de vue plus général, deux attitudes opposées peuvent être adoptées concernant le cadre lui-même. Ces deux attitudes ne sont pas incompatibles et correspondent respectivement à une démarche de généralisation et à une démarche de spécialisation :

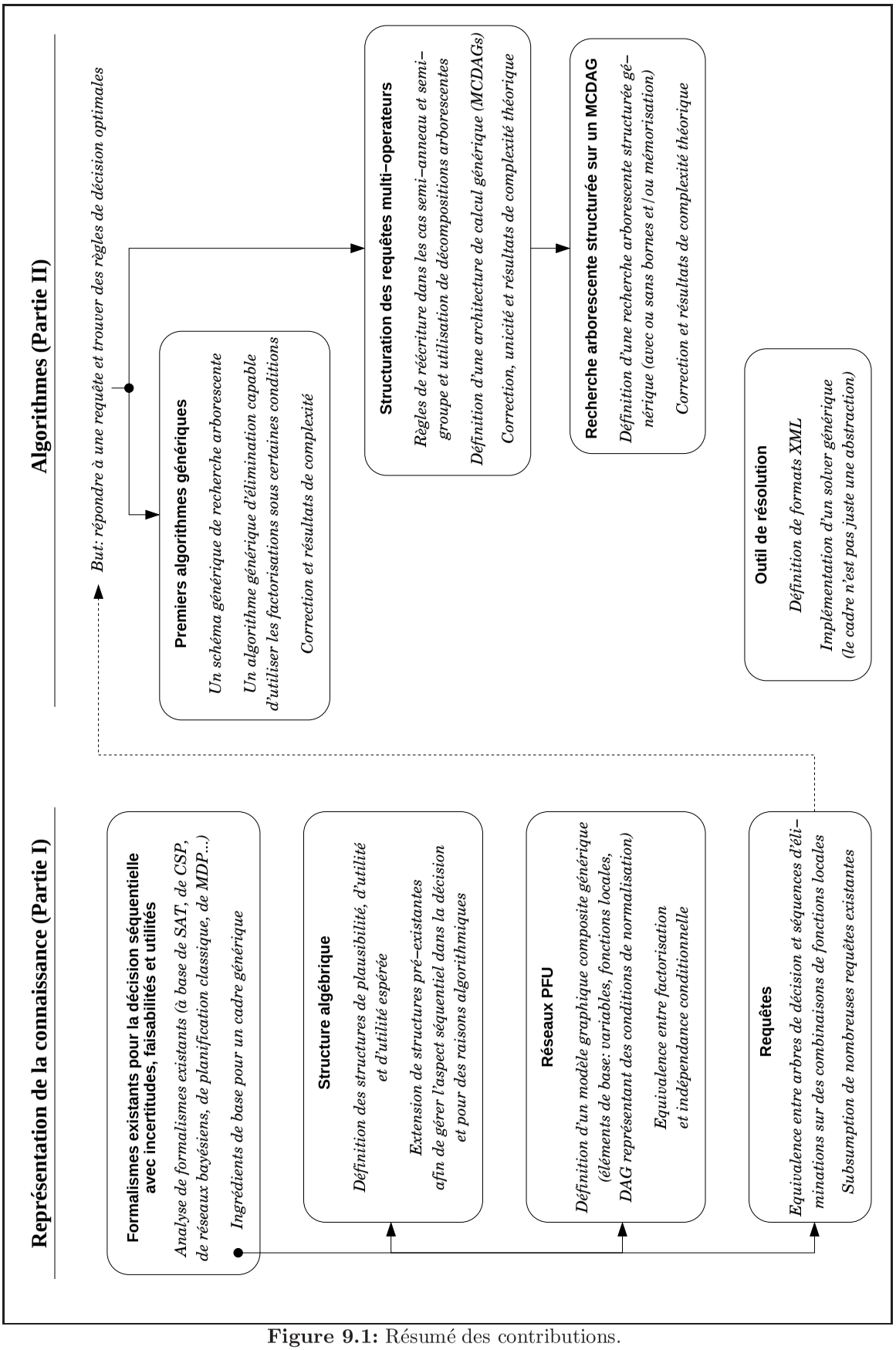

Figure 9.1: Résumé des contributions.

- Une première approche consiste à poursuivre la quête de la généricité, de manière à accroître le pouvoir d'expression du cadre PFU. Nous pourrions aussi définir des sortes de "multi-requêtes" permettant de poser plusieurs requêtes simultanément. C'est ce qui est par exemple implicitement fait dans les réseaux bayésiens pour calculer plusieurs distributions de probabilités marginales simultanément.
- La seconde approche consiste à identifier certains problèmes de base à résoudre et à se concentrer sur ces problèmes. Plus précisément, l'architecture MCDAG montre que les problèmes élémentaires à résoudre consistent souvent à calculer des quantités du type $\sum_S(\prod_{\varphi\in\Phi}\varphi)$, $\max_S(\sum_{\varphi\in\Phi}\varphi)$, ou $\max_S(\min_{\varphi\in\Phi}\varphi)$. Ces trois problèmes élémentaires correspondent aux types de calculs réalisés dans des réseaux bayésiens [73], des CSP pondérés [60], et des CSPs flous [35] respectivement. Une méthode possible pour justifier cette démarche de spécialisation est d'exhiber des morphismes entre des MCDAGs génériques et des MCDAGs utilisant juste les trois problèmes élémentaires mentionnés précédemment [20]. Une fois ce pré-requis algébrique effectué (s'il est possible de l'effectuer), l'architecture MCDAG peut être vue comme un entrelacement de ces trois problèmes élémentaires, à la frontière entre réseaux bayésiens et CSPs valués.

Dans les prochaines années, le cadre PFU permettra peut-être d'intégrer de nouvelles idées algorithmiques dans un outil de résolution flexible et générique. Il représente une opportunité de rassembler les nombreux efforts réalisés dans différentes communautés et de bénéficier des liens féconds entre algèbre, modèles graphiques et optimisation combinatoire.

Liste des tableaux

Table des figures

Bibliographie

[1] S.B. Akers. Binary Decision Diagrams. *IEEE Transactions on Computers*, 27(6), 1978.

[2] S.A. Arnborg. Efficient Algorithms for Combinatorial Problems on Graphs with Bounded Decomposability - A Survey. *BIT*, 25 :2–23, 1985.

[3] F. Bacchus and A. Grove. Graphical Models for Preference and Utility. In *Proc. of the 11th International Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 3–10, Montréal, Canada, 1995.

[4] B.W. Ballard. The *-Minimax Search Procedure for Trees Containing Chance Nodes. *Artificial Intelligence*, 21(3) :327–350, 1983.

[5] M. Benedetti. Quantifier Trees for QBF. In *Proc. of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT-05)*, St. Andrews, Scotland, 2005.

[6] U. Bertelé and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.

[7] S. Bistarelli, P. Codognet, and F. Rossi. Abstracting Soft Constraints : Framework, Properties, Examples. *Artificial Intelligence*, 139 :175–211, 2002.

[8] S. Bistarelli and F. Gadducci. Enhancing Constraints Manipulation in Semiring-based Formalisms. In *Proc. of the 17th European Conference on Artificial Intelligence (ECAI-06)*, Riva del Garda, Italy, 2006.

[9] S. Bistarelli, U. Montanari, and F. Rossi. Constraint Solving over Semirings. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 624–630, Montréal, Canada, 1995.

[10] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Satisfaction and Optimization. *Journal of ACM*, 44(2) :201–236, 1997.

[11] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-Based CSPs and Valued CSPs : Frameworks, Properties and Comparison. *Constraints*, 4(3) :199–240, 1999.

[12] H. L. Bodlaender. A Tourist Guide through Treewidth. *Acta Cybernetica*, 11 :1–21, 1993.

[13] L. Bordeaux and E. Monfroy. Beyond NP : Arc-consistency for Quantified Constraints. In *Proc. of the 8th International Conference on Principles and Practice of Constraint Programming (CP-02)*, Ithaca, New York, USA, 2002.

[14] C. Boutilier, R. Brafman, H. Hoos, and D. Poole. Reasoning With Conditional Ceteris Paribus Preference Statements. In *Proc. of the 15th International Conference on Uncertainty in Artificial Intelligence (UAI-99)*, Stockholm, Sweden, 1999.

[15] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning : Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11 :1–94, 1999.

[16] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic Dynamic Programming with Factored Representations. *Artificial Intelligence*, 121(1-2) :49–107, 2000.

[17] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8) :677–691, 1986.

[18] R. Chellappa and A. Jain. Markov Random Fields : Theory and Applications. Academic Press, 1993.

[19] F. Chu and J. Halpern. Great Expectations. Part I : On the Customizability of Generalized Expected Utility. In *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, 2003.

[20] M. Cooper and T. Schiex. Arc Consistency for Soft Constraints. *Artificial Intelligence*, 154(1-2) :199–227, 2004.

[21] A. Darwiche. Recursive Conditioning. *Artificial Intelligence*, 126(1-2) :5–41, 2001.

[22] A. Darwiche and M.L. Ginsberg. A Symbolic Generalization of Probability Theory. In *Proc. of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pages 622–627, San Jose, CA, USA, 1992.

[23] A. Darwiche and P. Marquis. A Knowledge Compilation Map. *Artificial Intelligence*, 17 :229–264, 2002.

[24] S. de Givry, G. Verfaillie, and T. Schiex. Bounding the Optimum of Constraint Optimization Problems. In *Proc. of the 3rd International Conference on Principles and Practice of Constraint Programming (CP-97)*, Schloss Hagenberg, Austria, 1997.

[25] R. Dechter. Bucket Elimination : a Unifying Framework for Reasoning. *Artificial Intelligence*, 113(1-2) :41–85, 1999.

[26] R. Dechter. A New Perspective on Algorithms for Optimizing Policies under Uncertainty. In *Proc. of the 5th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-00)*, pages 72–81, Breckenridge, CO, USA, 2000.

[27] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[28] R. Dechter and Y. El Fattah. Topological Parameters for Time-Space Tradeoff. *Artificial Intelligence*, 125(1-2) :93–118, 2001.

[29] R. Dechter and D. Larkin. Hybrid Processing of Beliefs and Constraints. In *Proc. of the 17th International Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 112–119, Seattle, WA, USA, 2001.

[30] R. Dechter and R. Mateescu. Mixtures of Deterministic-Probabilistic Networks and their AND/OR Search Space. In *Proc. of the 20th International Conference on Uncertainty in Artificial Intelligence (UAI-04)*, Banff, Canada, 2004.

[31] R. Dechter and R. Mateescu. AND/OR Search Spaces for Graphical Models. *To appear in Artificial Intelligence Journal*, 2006.

[32] R. Dechter, I. Meiry, and J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49 :61–95, 1991.

[33] R. Dechter and I. Rish. Mini-Buckets : A General Scheme for Bounded Inference. *Journal of the ACM*, 50(2) :107 – 153, 2003.

[34] R. Demirer and P.P. Shenoy. Sequential Valuation Networks : A New Graphical Technique for Asymmetric Decision Problems. In *Proc. of the 6th European Conference on Symbolic and Quantitavive Approaches to Reasoning with Uncertainty (ECSQARU-01)*, pages 252–265, London, UK, 2001.

[35] D. Dubois, H. Fargier, and H. Prade. The Calculus of Fuzzy Restrictions as a Basis for Flexible Constraint Satisfaction. In *Proc. of the 2nd IEEE Conference on Fuzzy Sets*, pages 1131–1136, San Francisco, CA, 1993.

[36] D. Dubois and H. Prade. Possibility Theory : An Approach to Computerized Processing of Uncertainty. Plenum Press, 1988.

[37] D. Dubois and H. Prade. Possibility Theory as a Basis for Qualitative Decision Theory. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1925–1930, Montréal, Canada, 1995.

[38] H. Fargier, J. Lang, and T. Schiex. Mixed Constraint Satisfaction : a Framework for Decision Problems under Incomplete Knowledge. In *Proc. of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pages 175–180, Portland, OR, USA, 1996.

[39] H. Fargier and P. Perny. Qualitative Models for Decision Under Uncertainty without the Commensurability Assumption. In *Proc. of the 15th International Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 188–195, Stockholm, Sweden, 1999.

[40] R. Fikes and N. Nilsson. STRIPS : a New Approach to the Application of Theorem Proving. *Artificial Intelligence*, 2(3-4) :189–208, 1971.

[41] N. Friedman and J. Halpern. Plausibility Measures : A User's Guide. In *Proc. of the 11th International Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 175–184, Montréal, Canada, 1995.

[42] M. Frydenberg. The Chain Graph Markov Property. *Scandinavian Journal of Statistics*, 17 :333–353, 1990.

[43] L. Garcia and R. Sabbadin. Possibilistic Influence Diagrams. In *Proc. of the 17th European Conference on Artificial Intelligence (ECAI-06)*, pages 372–376, Riva del Garda, Italy, 2006.

[44] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning : Theory and Practice*. Morgan Kaufmann, 2004.

[45] P.H. Giang and P.P. Shenoy. A Qualitative Linear Utility Theory for Spohn's Theory of Epistemic Beliefs. In *Proc. of the 16th International Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 220–229, Stanford, California, USA, 2000.

[46] R.P. Goldman and M.S. Boddy. Expressive Planning and Explicit Knowledge. In *Proc. of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 110–117, Edinburgh, Scotland, 1996.

[47] J. Halpern. Conditional Plausibility Measures and Bayesian Networks. *Journal of Artificial Intelligence Research*, 14 :359–389, 2001.

[48] R. Howard and J. Matheson. Influence Diagrams. In *Readings on the Principles and Applications of Decision Analysis*, pages 721–762. Strategic Decisions Group, Menlo Park, CA, USA, 1984.

[49] P. Jégou and C. Terrioux. Hybrid Backtracking bounded by Tree-decomposition of Constraint Networks. *Artificial Intelligence*, 146(1) :43–75, 2003.

[50] F. Jensen, F.V. Jensen, and S. Dittmer. From Influence Diagrams to Junction Trees. In *Proc. of the 10th International Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 367–373, Seattle, WA, USA, 1994.

[51] F.V. Jensen, T.D. Nielsen, and P.P. Shenoy. Sequential Influence Diagrams : A Unified Asymmetry Framework. In *Proceedings of the Second European Workshop on Probabilistic Graphical Models (PGM-04)*, pages 121–128, Leiden, Netherlands, 2004.

[52] F.V. Jensen and M. Vomlelova. Unconstrained Influence Diagrams. In *Proc. of the 18th International Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 234–241, Seattle, WA, USA, 2002.

[53] L. Khatib, P. Morris, R. Morris, and F. Rossi. Temporal Constraint Reasoning with Preferences. In *Proc. of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, Seattle, WA, USA, 2001.

[54] U. Kjaerulff. Triangulation of Graphs - Algorithms Giving Small Total State Space. Technical Report Tech. Report. R 90-09, Dept. of Mathematics and Computer Science, Aalborg University, Denmark, 1990.

[55] D. Knuth and R. Moore. An Analysis of Alpha-Beta Pruning. *Artificial Intelligence*, 8(4) :293–326, 1975.

[56] J. Kolhas. *Information Algebras : Generic Structures for Inference*. Springer, 2003.

[57] A.M.C.A. Koster, H.L. Bodlaender, and S.P.M. Van Hoesel. Treewidth : Computational Experiments. Technical report, Zentrum für Informationstechnik, Berlin, 2001.

[58] N. Kushmerick, S. Hanks, and D. Weld. An Algorithm for Probabilistic Planning. *Artificial Intelligence*, 76(1-2) :239–286, 1995.

[59] J. Larrosa and T. Schiex. In the quest of the best form of local consistency for weighted csp. In *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, 2003.

[60] J. Larrosa and T. Schiex. In the Quest of the Best Form of Local Consistency for Weighted CSP. In *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 239–244, Acapulco, Mexico, 2003.

[61] S. Lauritzen and D. Nilsson. Representing and Solving Decision Problems with Limited Information. *Management Science*, 47(9) :1235–1251, 2001.

[62] M. Littman, S. Majercik, and T. Pitassi. Stochastic Boolean Satisfiability. *Journal of Automated Reasoning*, 27(3) :251–296, 2001.

[63] A. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8(1) :99–118, 1977.

[64] A. Madsen and F.V. Jensen. Lazy Evaluation of Symmetric Bayesian Decision Problems. In *Proc. of the 15th International Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 382–390, Stockholm, Sweden, 1999.

[65] D. McDermott. PDDL, the Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control, 1998.

[66] N. Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 44, 1949.

[67] S. Minton, M. Johnston, A. Philips, and P. Laird. Minimizing Conflicts : a Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence*, 58 :160–205, 1992.

[68] G. Monahan. A Survey of Partially Observable Markov Decision Processes : Theory, Models, and Algorithms. *Management Science*, 28(1) :1–16, 1982.

[69] U. Montanari and F. Rossi. Constraint Relaxation may be Perfect. *Artificial Intelligence*, 48 :143–170, 1991.

[70] P. Ndilikilikesha. Potential Influence Diagrams. *International Journal of Approximated Reasoning*, 10 :251–285, 1994.

[71] T.D. Nielsen and F.V. Jensen. Representing and solving asymmetric decision problems. *International Journal of Information Technology and Decision Making*, 2 :217–263, 2003.

[72] J. Park and A. Darwiche. Complexity Results and Approximation Strategies for MAP Explanations. *Journal of Artificial Intelligence Research*, 21 :101–133, 2004.

[73] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[74] P. Perny, O. Spanjaard, and P. Weng. Algebraic Markov Decision Processes. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, Edinburgh, Scotland, 2005.

[75] C. Pralet, T. Schiex, and G. Verfaillie. Algorithmes et Complexités Génériques pour Différents Cadres de Décision Séquentielle dans l'Incertain. *Revue d'Intelligence Artificielle, à paraître*.

[76] C. Pralet, T. Schiex, and G. Verfaillie. Decomposition of Multi-Operator Queries on Semiring-based Graphical Models. In *Proc. of the 12th International Conference on Principles and Practice of Constraint Programming (CP-06)*, pages 437–452, Nantes, France, 2006.

[77] C. Pralet, T. Schiex, and G. Verfaillie. From Influence Diagrams to Multioperator Cluster DAGs. In *Proc. of the 22nd International Conference on Uncertainty in Artificial Intelligence (UAI-06)*, Cambridge, MA, USA, 2006.

[78] C. Pralet, T. Schiex, and G. Verfaillie. Une Nouvelle Architecture de Calcul pour Résoudre des Diagrammes d'Influence. In *Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA-06)*, Toulouse, France, 2006.

[79] C. Pralet, G. Verfaillie, and T. Schiex. An Algebraic Graphical Model for Decision with Uncertainties, Feasibilities, and Utilities. *Journal of Artificial Intelligence Research, to appear*.

[80] C. Pralet, G. Verfaillie, and T. Schiex. Un Cadre Graphique et Algébrique pour les Problèmes de Décision incluant Incertitudes, Faisabilités et Utilités. *Revue d'Intelligence Artificielle, à paraître*.

[81] C. Pralet, G. Verfaillie, and T. Schiex. Composite Graphical Models for Reasoning about Uncertainties, Feasibilities, and Utilities. In *Proc. of the CP-05 International Workshop on "Preferences and Soft Constraints"*, Sitges, Spain, 2005.

[82] C. Pralet, G. Verfaillie, and T. Schiex. Requêtes Complexes sur des Réseaux de Croyance-Faisabilité-Désir. In *Journées Francophones de Programmation par Contraintes (JFPC-05)*, Lens, France, 2005.

[83] C. Pralet, G. Verfaillie, and T. Schiex. Décision avec Incertitudes, Faisabilités et Utilités : vers un Cadre Algébrique Unifié. In *Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes (JFPDA-06)*, Toulouse, France, 2006.

[84] C. Pralet, G. Verfaillie, and T. Schiex. Decision with Uncertainties, Feasibilities, and Utilities : Towards a Unified Algebraic Framework. In *Proc. of the 17th European Conference on Artificial Intelligence (ECAI-06)*, pages 427–431, Riva del Garda, Italy, 2006.

[85] C. Pralet, G. Verfaillllie, and T. Schiex. Belief and Desire Networks for Answering Complex Queries. In *Proc. of the CP-04 Workshop on "Constraint Solving under Change and Uncertainty"*, Toronto, Canada, 2004.

[86] M. Puterman. *Markov Decision Processes, Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[87] C.P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, second edition, 2004.

[88] N. Robertson and P.D. Seymour. Graph Minors ii : Algorithmic Aspects of Treewidth. *Journal of Algorithms*, 7 :309–322, 1986.

[89] D.J. Rose. Triangulated Graphs and the Elimination Process. *Journal of Mathematical Analysis and Applications*, 32, 1970.

[90] F. Rossi, B. Venable, and N. Yorke-Smith. Simple Temporal Problems with Preferences and Uncertainty. In *Proc. of the CP-03 Workshop on "Handling Change and Uncertainty"*, Cork, Ireland, 2003.

[91] R. Sabbadin. A Possibilistic Model for Qualitative Sequential Decision Problems under Uncertainty in Partially Observable Environments. In *Proc. of the 15th International Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 567–574, Stockholm, Sweden, 1999.

[92] H. Samulowitz and F. Bacchus. Using SAT in QBF. In *Proc. of the 11th International Conference on Principles and Practice of Constraint Programming (CP-05)*, pages 578–592, Sitges, Spain, 2005.

[93] T. Sang, P. Beame, and H. Kautz. Solving Bayesian Networks by Weighted Model Counting. In *Proc. of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pages 475–482, Pittsburgh, PA, USA, 2005.

[94] T. Schiex, H. Fargier, and G. Verfaillie. Valued Constraint Satisfaction Problems : Hard and Easy Problems. In *Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 631–637, Montréal, Canada, 1995.

[95] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1998.

[96] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[97] P. Shenoy. Valuation-based Systems for Discrete Optimization. *Uncertainty in Artificial Intelligence*, 6 :385–400, 1991.

[98] P. Shenoy. Valuation-based Systems for Bayesian Decision Analysis. *Operations Research*, 40(3) :463–484, 1992.

[99] P. Shenoy. Conditional Independence in Valuation-Based Systems. *International Journal of Approximated Reasoning*, 10(3) :203–234, 1994.

[100] P.P. Shenoy. Valuation Network Representation and Solution of Asymmetric Decision Problems. *European Journal of Operational Research*, 121 :579–608, 2000.

[101] J.E. Smith, S. Holtzman, and J.E. Matheson. Structuring Conditional Relationships in Influence Diagrams. *Operations Research*, 41 :280–297, 1993.

[102] W. Spohn. A General Non-Probabilistic Theory of Inductive Reasoning. In *Proc. of the 6th International Conference on Uncertainty in Artificial Intelligence (UAI-90)*, pages 149–158, Cambridge, MA, USA, 1990.

[103] T.Vidal and M.Ghallab. Dealing with Uncertain Durations in Temporal Constraint Networks dedicated to Planning. In *Proc. of the 12th European Conference on Artificial Intelligence (ECAI-96)*, Budapest, Hungary, 1996.

[104] G. Verfaillie and C. Pralet. The Basic Ingredients of a Constraint-based Framework for Decision-making under Uncertainty. In *Proc. of the CP-05 International Workshop on "Constraint solving under Change and Uncertainty"*, Sitges, Spain, 2005.

[105] T. Vidal and H. Fargier. Handling Contingency in Temporal Constraint Networks : From Consistency to Controllabilities. *Journal of Experimental and Theoretical Artificial Intelligence*, 11(1) :23–45, 1999.

[106] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press, 1944.

[107] T. Walsh. Stochastic Constraint Programming. In *Proc. of the 15th European Conference on Artificial Intelligence (ECAI-02)*, pages 111–115, Lyon, France, 2002.

[108] E. Weydert. General Belief Measures. In *Proc. of the 10th International Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 575–582, 1994.