Une nouvelle architecture de calcul pour résoudre des diagrammes d'influence

From Influence Diagrams to Multi-operator Cluster DAGs *

Cédric Pralet[1] Thomas Schiex[2] Gérard Verfaillie[3]

[1] LAAS-CNRS & INRA/BIA, Toulouse, cpralet@laas.fr
[2] INRA/BIA, Toulouse, tschiex@toulouse.inra.fr
[3] ONERA/DCSD, Toulouse, Gerard.Verfaillie@onera.fr

Résumé

Différentes architectures de calcul existent pour résoudre des diagrammes d'influence via des calculs locaux, telles celles connues sous les dénominations de Shenoy-Shafer*,* HUGIN *ou* Lazy Propagation*. Toutes étendent les algorithmes classiques d'élimination de variables grâce à l'utilisation de* potentiels*. Dans cet article, nous introduisons une nouvelle architecture de calcul, basée sur un graphe orienté sans circuit de clusters multi-opérateurs (MC-DAG), qui peut produire des décompositions ayant une plus faible* largeur induite contrainte *et induire de ce fait des gains potentiellement exponentiels. Son principe consiste à bénéficier de la nature composite des diagrammes d'influence, au lieu d'utiliser des potentiels uniformes, de façon à mieux analyser la structure du problème.*

Mots Clef

Diagrammes d'influence, Élimination de variables, Décomposition en clusters.

Abstract

There exist several computing architectures to solve influence diagrams using local computations, such as the Shenoy-Shafer*, the* HUGIN*, or the* Lazy Propagation *architectures. They all extend usual variable elimination algorithms thanks to the use of so-called* potentials*. In this paper, we introduce a new computing architecture, called the* Multi-operator Cluster DAG *architecture, which can produce decompositions with an improved* constrained induced-width*, and therefore induce potentially exponential gains. Its principle is to benefit from the composite nature of influence diagrams, instead of using uniform potentials, in order to better analyze the problem structure.*

Keywords

Influence Diagrams, Variable elimination, Cluster decomposition.

1 Introduction

Depuis les premiers algorithmes basés sur des *arbres de décision* ou sur des opérations d'*inversion d'arc* [11], différentes méthodes exactes ont été proposées pour résoudre des diagrammes d'influence via des calculs locaux, comme celles utilisant les architectures de calcul *Shenoy-Shafer*, *HUGIN* ou *Lazy Propagation* (LP) [13, 4, 6]. Ces méthodes ont permis d'adapter les algorithmes classiques d'*élimination de variables* (VE) (qui sont basiquement dédiés au calcul d'un type de marginalisation sur une combinaison de fonctions locales avec un type d'opérateur de combinaison) à la manipulation de plusieurs types d'information (probabilités et utilités), plusieurs types de marginalisation ($+$ pour les variables aléatoires et $\max$ pour les variables de décision) et plusieurs types de combinaison ($\times$ pour les probabilités et $+$ pour les utilités), tels qu'ils apparaissent dans un diagramme d'influence. Le mécanisme clé d'une telle extension est l'utilisation d'éléments appelés *potentiels* [7].

Dans ce papier, nous définissons une nouvelle architecture de calcul, appelée *Multi-operator Cluster DAG* (MCDAG), qui n'utilise pas de potentiels, mais repose toujours sur un algorithme de type VE. Comparés aux schémas existants, les MCDAG exploitent activement la nature composite des diagrammes d'influence. Nous présentons d'abord l'approche à base de potentiels et nous motivons le besoin d'une nouvelle architecture de calcul (section 2). Puis, les MCDAG sont introduits (section 3) et un algorithme de type VE est défini (section 4). Finalement, ce travail est comparé avec des approches existantes (section 5) et étendu à d'autres cadres (section 6). Toutes les preuves sont disponibles à [10].

*Cet article a été soumis en anglais à UAI06. Le travail décrit a par ailleurs été partiellement mené dans le cadre du projet européen COGNIRON (*The Cognitive Companion*).

2 Motivations

Notations et définitions Un *diagramme d'influence* [3] est un modèle graphique composite défini sur trois ensembles de variables organisées en un *graphe orienté sans circuit* (DAG) G : (1) un ensemble C de *variables aléatoires* avec, pour chaque variable $c \in C$, la spécification d'une *distribution de probabilité conditionnelle* $P_{c\,|\,pa(c)}$ sur c étant donnés ses parents $pa(c)$ dans G ; (2) un ensemble $D = \{D_1, \ldots, D_n\}$ de *variables de décision*, sachant que les indices représentent l'*ordre* dans lequel les décisions sont prises et que, pour chaque variable $d \in D$, $pa(d)$ représente l'ensemble des variables qui sont *observées* avant de prendre la décision d ; (3) un ensemble Γ de *variables d'utilité* avec, pour chaque variable $u \in \Gamma$, la spécification d'une *fonction locale d'utilité* $U_{pa(u)}$ définie sur ses parents $pa(u)$ dans G (les variables d'utilité sont des *feuilles* du DAG).

L'ensemble des distributions de probabilité conditionnelles (une pour chaque $c \in C$) est noté P et l'ensemble des fonctions locales d'utilité (une pour chaque $u \in \Gamma$) est noté U. Chaque fonction locale $\phi \in P \cup U$ est définie sur un ensemble de variables $sc(\phi)$ appelée sa *portée* ($sc(P_{x\,|\,pa(x)}) = \{x\} \cup pa(x)$ et $sc(U_{pa(u)}) = pa(u)$). L'ensemble des variables aléatoires observées avant la première décision D_1 est noté I_0, l'ensemble des variables aléatoires observées entre les décisions D_k and D_{k+1} est noté I_k et l'ensemble des variables aléatoires non observées avant la dernière décision D_n est noté I_n. Nous utilisons $dom(x)$ pour désigner le domaine d'une variable $x \in C \cup D$ et, par extension, pour tout $W \subset C \cup D$, $dom(W) = \prod_{x \in W} dom(x)$.

Le problème classique associé à un diagramme d'influence consiste à trouver les *règles de décision* qui maximisent l'utilité espérée (une règle de décision pour une décision D_k est une fonction qui associe une valeur de $dom(D_k)$ à toute affectation des variables observées avant de prendre la décision D_k). Comme cela est montré dans [4], ceci est équivalent à calculer les règles de décision optimale associées à la quantité :

$$\sum_{I_0} \max_{D_1} \ldots \sum_{I_{n-1}} \max_{D_n} \sum_{I_n} ((\prod_{P_i \in P} P_i) \times (\sum_{U_i \in U} U_i)) \quad (1)$$

2.1 L'approche à base de potentiels

Avec cette approche, l'équation 1 est reformulée en utilisant ce qu'on appelle des *potentiels* de façon à n'utiliser qu'un seul opérateur de *combinaison* et un seul opérateur de *marginalisation* (ou *élimination*). Un potentiel sur un ensemble de variables W est une paire $\pi_W = (p_W, u_W)$, où p_W et u_W sont respectivement une fonction réelle positive et une fonction réelle, dont les portées sont incluses dans W. Les distributions initiales de probabilité conditionnelle $P_i \in P$ sont transformées en des potentiels $(P_i, 0)$, tandis que les fonctions initiales d'utilité $U_i \in U$ sont transformées en des potentiels $(1, U_i)$. Sur ces potentiels, une opération de combinaison $\otimes$ et une opération d'élimination $\downarrow$ sont définies :

— la combinaison de $\pi_{W_1} = (p_{W_1}, u_{W_1})$ et de $\pi_{W_2} = (p_{W_2}, u_{W_2})$ est le potentiel sur $W_1 \cup W_2$ donné par $\pi_{W_1} \otimes \pi_{W_2} = (p_{W_1} \times p_{W_2}, u_{W_1} + u_{W_2})$;

— l'élimination de $\pi_W = (p_W, u_W)$ sur $W_1 \subset C$ est le potentiel sur W_1 donné par $\pi_W^{\downarrow W_1} = \left(\sum_{W-W_1} p_W, \frac{\sum_{W-W_1} p_W u_W}{\sum_{W-W_1} p_W}\right)$ (avec la convention $0/0 = 0$), tandis que l'élimination de $\pi_W = (p_W, u_W)$ sur $W_1 \subset D$ est le potentiel sur W_1 donné par $\pi_W^{\downarrow W_1} = (p_W, \max_{W-W_1} u_W)$.

Résoudre le problème associé à un diagramme d'influence est alors équivalent au calcul de $Q = ((\cdots((\pi_{C\cup D}{}^{\downarrow I_n})^{\downarrow D_n})^{\downarrow I_{n-1}} \cdots)^{\downarrow D_1})^{\downarrow I_0}$, où $\pi_{C\cup D} = (\otimes_{P_i \in P}(P_i, 0)) \otimes (\otimes_{U_i \in U}(1, U_i))$ est la combinaison des potentiels initiaux. Comme $\otimes$ et $\downarrow$ satisfont les axiomes de Shenoy-Schafer définis dans [12], Q peut être calculé en utilisant un algorithme classique de type VE [4]. Cela explique pourquoi les potentiels sont utilisés par les architectures de calcul existantes qui visent à résoudre les diagrammes d'influence via des calculs locaux : Shenoy-Shafer, HUGIN et LP. [1]

2.2 Mesurer la complexité

Dans le cas des diagrammes d'influence, l'alternance d'éliminations de type $\sum$ et $\max$, qui généralement ne commutent pas, interdit d'éliminer les variables dans un ordre quelconque. La complexité d'un algorithme de type VE peut alors être quantifiée en utilisant la *largeur induite contrainte* [4, 8] (au lieu de la *largeur induite* [1]).

Definition 1. *Soit $G = (V_G, H_G)$ un hyper-graphe* [2] *et $\preceq$ un ordre partiel sur V_G. La largeur induite contrainte de G avec les contraintes sur l'ordre d'élimination imposées par $\preceq$ ("$x \prec y$" signifie que "y doit être éliminé avant x") est un paramètre noté $w_G(\preceq)$. Il est défini par $w_G(\preceq) = \min_{o \in lin(\preceq)} w_G(o)$, $lin(\preceq)$ étant l'ensemble des linéarisations de $\preceq$ en un ordre total sur V_G et $w_G(o)$ étant la largeur induite de G pour l'ordre d'élimination o (c'est-à-dire la taille de la plus grande hyper-arête créée en éliminant les variables suivant l'ordre o).*

La largeur induite contrainte peut être utilisée pour produire un majorant de la complexité des algorithmes de type VE à base de potentiels. Soit $G_p = (C \cup D, \{sc(P_i) | P_i \in P\} \cup \{sc(U_i) | U_i \in U\})$ l'hyper-graphe correspondant au diagramme d'influence "non typé". Soit $\preceq_p$ l'ordre partiel défini par $I_0 \prec_p D_1$, $(I_k \neq \emptyset) \rightarrow (D_k \prec_p I_k \prec_p D_{k+1})$ et $D_n \prec_p I_n$. Finalement, soit d la taille maximum des domaines des variables. Alors, avec des approches classiques à base de potentiels et des *arbres de jonction forte* [4], qui sont des arbres de jonction avec des contraintes sur

1. L'architecture LP utilise en fait des potentiels définis comme des paires d'ensembles de fonctions (plutôt que des paires de fonctions).

2. Cela signifie que V_G est un ensemble des variables (ou sommets) et H_G un ensemble d'hyper-arêtes sur V_G, c'est-à-dire un sous-ensemble de 2^{V_G}.

l'ordre d'élimination, la complexité théorique est $O(|P \cup U| \cdot d^{1+w_{G_p}(\preceq_p)})$ (le cardinal d'un ensemble E est noté $|E|$).

2.3 Diminuer la largeur induite contrainte

La largeur induite contrainte est un guide pour montrer comment diminuer la complexité. Pour cela, on peut travailler sur les deux paramètres dont elle dépend : l'ordre partiel $\preceq$ et l'hyper-graphe G.

Relâcher l'ordre partiel

Proposition 1. *Soit $G = (V_G, H_G)$ un hyper-graphe et $\preceq_1$, $\preceq_2$ deux ordres partiels V_G tels que $\forall (x, y) \in V_G \times V_G$, $(x \preceq_2 y) \rightarrow (x \preceq_1 y)$ ($\preceq_2$ est plus faible que $\preceq_1$). Alors, $w_G(\preceq_1) \geq w_G(\preceq_2)$.*

La proposition 1 signifie que si l'on relâche $\preceq$, c'est-à-dire si l'on fait apparaître des libertés supplémentaires dans l'ordre d'élimination (par exemple en prouvant que certaines éliminations avec $+$ et $\max$ peuvent commuter), alors la complexité théorique peut décroître. Bien qu'une telle technique soit inutile dans des contextes tels que le problème MAP (Maximum A Posteriori hypothesis [8]), où il y a seulement une alternance d'éliminations avec $+$ et $\max$, elle peut conduire à des gains exponentiels dès qu'il y a plus de deux niveaux d'alternances.
En effet, supposons que l'on veuille calculer $\max_{x_1,\ldots,x_n} \sum_y \max_{x_{n+1}} P_y \cdot \left(U_{x_1,y} + \sum_{i\in[1,n]} U_{x_i,x_{n+1}}\right)$. D'une part, l'ordre partiel $\preceq_1$ défini par $\{x_1, \ldots, x_n\} \prec_1 y \prec_1 x_{n+1}$ donne une largeur induite contrainte $w_G(\preceq_1) = n$, puisque x_{n+1} est forcément éliminé en premier. D'autre part, les portées des fonctions nous permettent d'établir qu'avec l'ordre partiel $\preceq_2$ défini par $x_1 \prec_2 y$, on obtient le même résultat, puisque y n'est liée qu'avec x_1. La largeur induite contrainte est alors $w_G(\preceq_2) = 1$, par exemple avec l'ordre d'élimination $x_1 \prec y \prec x_{n+1} \prec x_n \prec \ldots \prec x_2$. En conséquence, la complexité théorique décroît de $O((n+2) \cdot d^{n+1})$ à $O((n+2) \cdot d^2)$ grâce au relâchement de l'ordre partiel (le facteur $(n+2)$ correspond au nombre de fonctions locales).

Travailler sur l'hyper-graphe Le second mécanisme possible consiste à travailler sur l'hyper-graphe G, soit en éliminant des variables dites *stériles* (calculer $\sum_x P_{x\,|\,pa(x)}$ est inutile à cause de la normalisation), soit en décomposant mieux le problème. Pour illustrer cette seconde option, supposons que l'on veuille calculer $\max_{x_1,\ldots,x_n} \sum_y P_y \cdot (U_{y,x_1} + \ldots + U_{y,x_n})$. L'hyper-graphe initial $G_1 = (\{x_1, \ldots, x_n, y\}, \{\{y, x_1\}, \ldots, \{y, x_n\}\})$, avec l'ordre partiel $\preceq_1$ défini par $\{x_1, \ldots, x_n\} \prec_1 y$, donne une complexité théorique $O((n+1) \cdot d^{w_{G_1}(\preceq_1)+1}) = O((n+1) \cdot d^{n+1})$. On peut cependant écrire :

$$\max_{x_1,\ldots,x_n} \sum_y P_y \cdot (U_{y,x_1} + \ldots + U_{y,x_n})$$
$$= (\max_{x_1} \sum_y P_y \cdot U_{y,x_1}) + \ldots + (\max_{x_n} \sum_y P_y \cdot U_{y,x_n})$$

Ainsi, une *duplication* implicite de y donne une complexité théorique $O((n+1)d^2) = O((n+1)d^{1+w_{G_2}(\preceq_2)})$, où G_2 est l'hyper-graphe défini par les variables $\{x_1, \ldots, x_n, y^{(1)}, \ldots, y^{(n)}\}$ et par les hyper-arêtes $\{\{x_1, y^{(1)}\}, \ldots, \{x_n, y^{(n)}\}\}$ et où $\preceq_2$ est l'ordre partiel défini par $x_1 \prec_2 y^{(1)}, \ldots, x_n \prec_2 y^{(n)}$. Cette méthode, qui utilise la propriété $\sum_S (U_1 + U_2) = (\sum_S U_1) + (\sum_S U_2)$, duplique les variables "quantifiées" avec $+$ pour aboutir à des calculs plus locaux. La proposition 2 montre le gain potentiellement exponentiel résultant de la duplication.

Proposition 2. *Soit ϕ_{x,S_i} une fonction locale de portée $\{x\} \cup S_i$ pour tout $i \in [1, m]$. Le calcul brut de $\sum_x (\phi_{x,S_1} + \ldots + \phi_{x,S_m})$ nécessite toujours plus d'opérations $+$ que le calcul brut $(\sum_x \phi_{x,S_1}) + \ldots + (\sum_x \phi_{x,S_m})$. De plus, le calcul brut de $\sum_x (\phi_{x,S_1} + \ldots + \phi_{x,S_m})$ a une complexité $O(m \cdot d^{1+|S_1 \cup \ldots \cup S_m|})$, tandis que celui des m quantités de l'ensemble $\{\sum_x \phi_{x,S_i} \,|\, 1 \leq i \leq m\}$ a une complexité $O(m \cdot d^{1+\max_{i\in[1,m]} |S_i|})$.* □

Pourquoi ne pas utiliser des potentiels ? Bien que relâcher les contraintes sur l'ordre d'élimination soit possible avec des potentiels, dupliquer des variables ne l'est pas. En effet, on ne peut pas écrire $(\pi_{W_1} \otimes \pi_{W_2})^{\downarrow W_3} = (\pi_{W_1}^{\downarrow W_3}) \otimes (\pi_{W_2}^{\downarrow W_3})$, même si $W_3 \subset C$. De plus, le mécanisme de duplication lui-même peut relâcher certaines contraintes sur l'ordre d'élimination. En conséquence, nous introduisons une nouvelle architecture de calcul qui n'utilise pas les potentiels pour résoudre des diagrammes d'influence via des calculs locaux. Cette architecture est appelée *Multi-operator Cluster DAG* (MCDAG).

3 L'architecture MCDAG

3.1 Macrostructurer un diagramme d'influence

La première étape de la construction d'une architecture MCDAG consiste à analyser la *macrostructure* du diagramme d'influence, en détectant les possibilités de relâchement de l'ordre d'élimination, tout en utilisant les possibilités de duplication et les conditions de normalisation sur les distributions de probabilité conditionnelle. Cette macrostructure est représentée par un DAG de *nœuds de calcul*.

Definition 2. *Un nœud de calcul n est :*
- *soit une fonction locale ϕ de $P \cup U$; dans ce cas, la valeur de n est donnée par $val(n) = \phi$ et sa portée est donnée par $sc(n) = sc(\phi)$;*
- *soit un triplet $(Sov, \circledast, N)$, où Sov est une séquence de paires opérateur-variables, $\circledast$ est un opérateur associatif et commutatif disposant d'un élément neutre et N un ensemble de nœuds de calcul ; dans ce cas, la valeur de n est donnée par $val(n) = Sov\,(\circledast_{n'\in N}\, val(n'))$ et sa portée est donnée par $sc(n) = (\cup_{n'\in N}\, sc(n')) - \{x \,|\, op_x \in Sov\}$.*

Informellement, un nœud de calcul $(Sov, \circledast, N)$ définit une séquence d'éliminations sur une combinaison de nœuds de

calcul avec un opérateur $\circledast$ spécifique. Il peut être représenté comme dans la figure 1. Étant donné un ensemble N de nœuds de calcul, nous définissons N^{+x} (respectivement N^{-x}) comme l'ensemble des nœuds de N dont la portée contient x (respectivement ne contient pas x) : $N^{+x} = \{n \in N \mid x \in sc(n)\}$ (respectivement $N^{-x} = \{n \in N \mid x \notin sc(n)\}$).

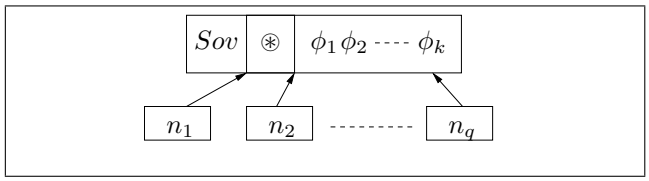

FIGURE 1: Un nœud de calcul $(Sov, \circledast, N)$, où $N \cap (P \cup U) = \{\phi_1, \ldots, \phi_k\}$ et $N - (P \cup U) = \{n_1, \ldots, n_q\}$.

Des diagrammes d'influence aux nœuds de calcul. Sans perte de généralité, nous supposons que $U \neq \emptyset$ (si ce n'est pas le cas, nous pouvons ajouter $U_0 = 1$ à U).

Proposition 3. *Soit Sov_0 la séquence initiale $\sum_{I_0} \max_{D_1} \ldots \sum_{I_{n-1}} \max_{D_n} \sum_{I_n}$ de paires opérateur-variables définie par le diagramme d'influence. La valeur de l'équation 1 est égale à la valeur du nœud de calcul $n_0 = (Sov_0, +, \{(\emptyset, \times, P \cup \{U_i\}), U_i \in U\})$.*

Par exemple, pour le diagramme d'influence associé au calcul de $\max_d \sum_{r_2,r_1} P_{r_1} \cdot P_{r_2|r_1} \cdot (U_{d,r_1} + U_{d,r_2} + U_d)$, n_0 correspond au premier nœud de calcul dans la figure 2.

Macrostructurer le nœud initial. De façon à mettre en évidence la macrostructure du diagramme d'influence, nous analysons la séquence de calculs réalisée par n_0. Pour cela, nous considérons successivement les éliminations de Sov_0 de la droite vers la gauche et nous utilisons trois types de *règles de réécriture*, qui préservent toutes la valeur des nœuds : (1) des *règles de décomposition* qui décomposent la structure en utilisant le mécanisme de duplication ; (2) des *règles de recomposition* qui révèlent des libertés dans l'ordre d'élimination ; (3) des *règles de simplification* qui éliminent les calculs inutiles en utilisant les conditions de normalisation. Ces règles de réécriture sont présentées d'abord dans le cas d'éliminations de type $\sum_x$, puis dans le cas d'éliminations de type $\max_x$. Une règle de réécriture peut être précédée de pré-conditions qui restreignent son champ d'application.

Règles de réécriture dans le cas d'une élimination de type $\sum_x$ Quand une élimination de type $\sum_x$ doit être réalisée, une règle de décomposition D_Σ, une règle de recomposition R_Σ et deux règles de simplification S^1_Σ et S^2_Σ sont utilisées. Elles sont illustrées dans la figure 2 qui correspond au diagramme d'influence introduit plus haut.

$\boxed{D_\Sigma}$ $\quad (Sov.\sum_x, +, \{(\emptyset, \times, N), N \in \mathfrak{N}\})$
$\rightsquigarrow \left(Sov, +, \left\{\left(\emptyset, \times, N^{-x} \cup \left\{\left(\sum_x, \times, N^{+x}\right)\right\}\right), N \in \mathfrak{N}\right\}\right)$

$\boxed{R_\Sigma}$ [Prec. : $(S' \cap (S \cup sc(N_1)) = \emptyset) \wedge (N_1 \cap N_2 = \emptyset)$]
$(\sum_S, \times, N_1 \cup \{(\sum_{S'}, \times, N_2)\}) \rightsquigarrow (\sum_{S \cup S'}, \times, N_1 \cup N_2)$

$\boxed{S^1_\Sigma}$ [Prec. : $x \notin S \cup sc(N)$]
$(\sum_{\{x\} \cup S}, \times, N \cup \{P_{x \mid pa(x)}\}) \rightsquigarrow (\sum_S, \times, N)$

$\boxed{S^2_\Sigma}$ $\quad (\emptyset, \times, N \cup \{(\sum_\emptyset, \times, \emptyset)\}) \rightsquigarrow (\emptyset, \times, N)$

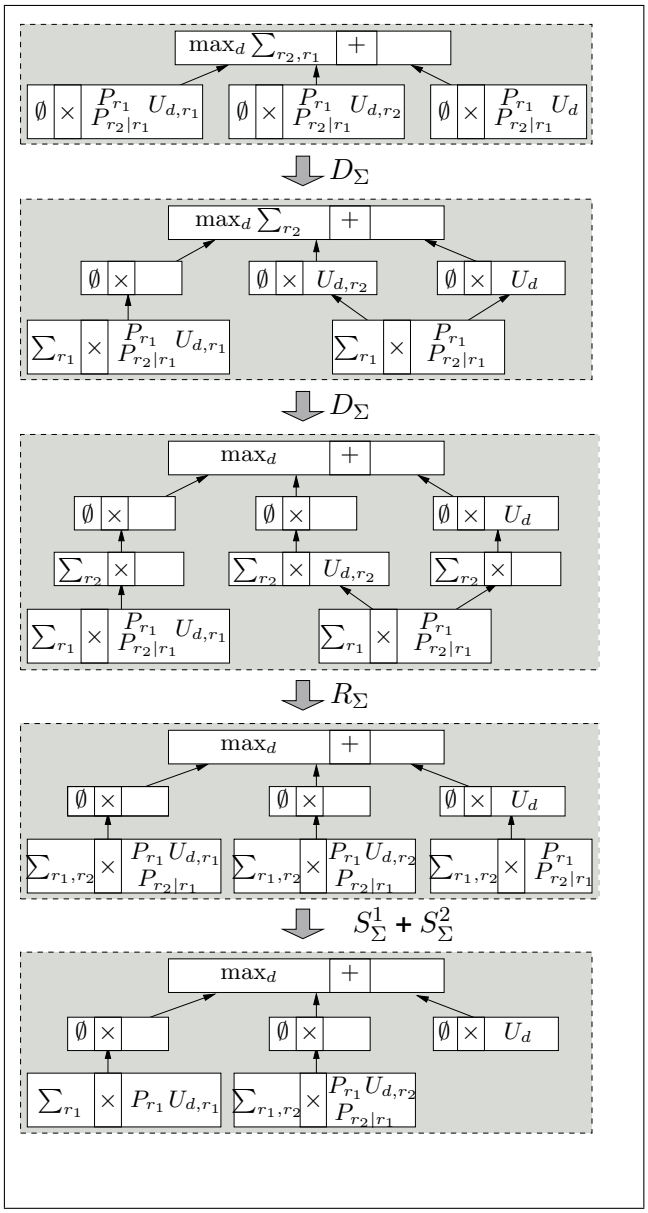

FIGURE 2: Application des règles de réécriture dans le cas d'éliminations de type $\sum_x$.

Exemple Dans l'exemple de la figure 2, la première règle à appliquer est la règle de décomposition D_Σ, qui

traite la paire opérateur-variable $\sum_{r_1}$.[3] Une telle règle utilise le mécanisme de duplication et la distributivité de $\times$ sur $+$. Il produit un DAG de nœuds de calcul. C'est un DAG puisque des nœuds de calcul identiques sont réunis (ce n'est pas difficile de détecter de tels nœuds en appliquant les règles). Puis, D_Σ peut de nouveau être appliqué sur $\sum_{r_2}$. On peut inférer de l'architecture obtenue qu'il n'y a pas de raison que r_1 soit éliminé avant r_2. L'utilisation de la règle de recomposition R_Σ le rend explicite dans la structure. Basiquement, R_Σ utilise la distributivité de $\times$ sur $+$. Finalement, l'application de S^1_Σ et de S^2_Σ, qui utilisent la normalisation des distributions de probabilité conditionnelle, simplifie certains nœuds de calcul. À la fin, aucun cluster n'implique plus de deux variables (si on élimine r_1 en premier dans le nœud $(\sum_{r_1,r_2}, \times, \{P_{r_1}, P_{r_2|r_1}, U_{d,r_2}\})$), alors qu'avec une approche à base de potentiels, il aurait été nécessaire de considérer trois variables simultanément (car r_1 serait impliqué dans les potentiels $(P_{r_1}, 0), (P_{r_2|r_1}, 0), (1, U_{d,r_1})$ s'il était éliminé en premier et r_2 dans les potentiels $(P_{r_2|r_1}, 0), (1, U_{d,r_2})$ si c'était lui qui était éliminé en premier).

Règles de réécriture dans le cas d'une élimination de type $\max_x$ Quand une élimination de type $\max_x$ doit être réalisée, une règle de décomposition $D_{\max}$ et une règle de recomposition $R_{\max}$ sont utilisées (il n'y a pas de règle de simplification puisqu'il n'y a pas de condition de normalisation à utiliser sur les variables de décision). Ces règles, qui sont un peu plus complexes que les précédentes, sont illustrées dans la figure 3 qui correspond au diagramme d'influence $\max_{d_1} \sum_{r_2} \max_{d_2} \sum_{r_1} \max_{d_3} P_{r_1} \cdot P_{r_2|r_1} \cdot (U_{d_1} + U_{d_2,d_3} + U_{r_2,d_1,d_3} + U_{r_1,d_2})$.

$\boxed{D_{\max}}$ [Prec. : $\forall N \in \mathfrak{N}^{+x} \forall n \in N^{-x}$, $val(n) \geq 0$]

$$(Sov.\max_x, +, \{(\emptyset, \times, N), N \in \mathfrak{N}\})$$
$$\rightsquigarrow \begin{cases} (Sov, +, \{(\emptyset, \times, N), N \in \mathfrak{N}\}) \text{ if } \mathfrak{N}^{+x} = \emptyset \\ (Sov, +, \{(\emptyset, \times, N), N \in \mathfrak{N}^{-x}\} \\ \quad \cup \{(\emptyset, \times, N_1 \cup \{(\max_x, +, N_2)\})\}) \text{ otherwise} \end{cases}$$
$$\text{where } \begin{cases} N_1 = \cap_{N \in \mathfrak{N}^{+x}} N^{-x} \\ N_2 = \{(\emptyset, \times, N - N_1), N \in \mathfrak{N}^{+x}\} \end{cases}$$

$\boxed{R_{\max}}$ [Prec. : $(S' \cap (S \cup sc(N_1) \cup sc(N_2)) = \emptyset) \wedge (\forall N_3 \in \mathfrak{N},\ N_2 \cap N_3 = \emptyset) \wedge (\forall n \in N_2, val(n) \geq 0)$]

$$(\max_S, +, N_1 \cup \{(\emptyset, \times, N_2 \cup \{(\max_{S'}, +, \{(\emptyset, \times, N_3), N_3 \in \mathfrak{N}\})\})\})$$
$$\rightsquigarrow (\max_{S \cup S'}, +, N_1 \cup \{(\emptyset, \times, N_2 \cup N_3), N_3 \in \mathfrak{N}\})$$

Exemple Dans l'exemple de la figure 3, on applique d'abord la règle de décomposition $D_{\max}$ pour traiter la paire opérateur-variable $\max_{d_3}$. Une telle règle utilise d'abord la monotonie de $+$ ($\max(a+b, a+c) = a +$

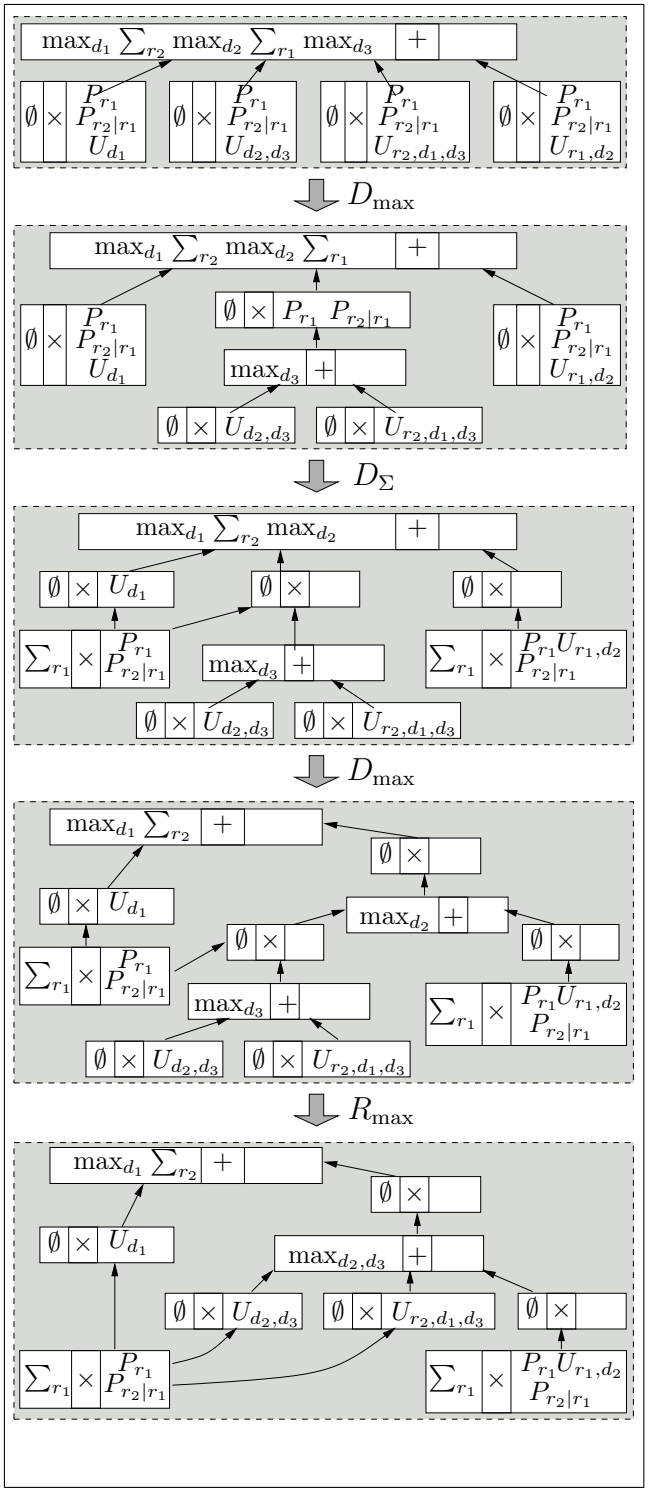

FIGURE 3: Application des règles de réécriture dans le cas d'éliminations de type $\max_x$ (l'application de ces règles peut créer des nœuds de calcul de la forme $(\emptyset, \times, \{n\})$, qui ne font aucun calcul ; ces nœuds peuvent être éliminés dans une étape finale).

3. Pour les besoins de l'exemple, r_1 est éliminé avant r_2, même si l'inverse aurait été plus efficace.

$\max(b,c)$), puis la distributivité de $\times$ sur $+$ et la monotonie de $\times$ (de façon à écrire des choses comme $\max_{d_3}((P_{r_1} \cdot P_{r_2|r_1} \cdot U_{d_2,d_3}) + (P_{r_1} \cdot P_{r_2|r_1} \cdot U_{r_2,d_1,d_3})) = P_{r_1} \cdot P_{r_2|r_1} \cdot \max_{d_3}(U_{d_2,d_3} + U_{r_2,d_1,d_3})$). Puis, D_Σ peut être utilisée sur $\sum_{r_1}$ et $D_{\max}$ sur $\max_{d_2}$. Ensuite, la règle de recomposition $R_{\max}$, qui utilise la monotonie de $\times$ et de $+$, fait apparaître que l'ordre d'élimination entre d_2 et d_3 est en fait libre. Cela n'était pas évident sur la séquence initiale Sov_0. L'approche à base de potentiels est incapable de faire apparaître de telles libertés, ce qui peut induire un rapport exponentiel en termes de complexité, comme cela a été montré en 2.3.

Ordre d'application des règles Une application chaotique des règles ne converge pas, puisque par exemple les règles $D_{\max}$ et $R_{\max}$ peuvent être appliquées alternativement de façon infinie. En conséquence, nous spécifions un ordre suivant lequel les règles peuvent être utilisées pour converger un unique DAG de nœuds de calcul (nous avons implicitement utilisé cet ordre dans les exemples précédents). Nous considérons successivement chaque paire opérateur-variable de la séquence initiale Sov_0 de la droite vers la gauche (des éliminations comme $\sum_{x_1,\ldots,x_n}$ peuvent être découpées en $\sum_{x_1} \cdots \sum_{x_n}$).
Si l'élimination la plus à droite dans la séquence Sov du nœud racine est de type $\sum_x$, alors la règle D_Σ est appliquée une fois. Elle crée de nouveaux nœuds petit-fils de la racine. Sur chacun d'entre eux, nous essayons d'appliquer la règle R_Σ de façon à faire apparaître des libertés dans l'ordre d'élimination. Si R_Σ est appliquée, elle crée de nouveaux nœuds. Sur chacun d'entre eux, les règles S^1_Σ puis S^2_Σ sont appliquées (jusqu'à ce qu'elles ne puissent plus l'être).
Si l'élimination la plus à droite dans la séquence Sov du nœud racine est de type $\max_x$, alors la règle $D_{\max}$ est appliquée une fois. Elle crée un nouveau nœud fils et un nouveau nœud petit-fils de la racine. Sur le nœud petit-fils, nous essayons de relâcher les contraintes sur l'ordre d'élimination en utilisant la règle $R_{\max}$.
La correction de la macrostructure obtenue résulte de la correction des règles de réécriture :

Proposition 4. *Les règles de réécritureD_Σ, R_Σ, S^1_Σ, S^2_Σ, $D_{\max}$ et $R_{\max}$ sont correctes, ce qui veut dire que, pour chacune de ces règles $n_1 \rightsquigarrow n_2$, si les pré-conditions sont satisfaites, alors $val(n_1) = val(n_2)$. De plus, les règles $D_{\max}$ et $R_{\max}$ ne modifient pas l'ensemble des règles de décision optimale.*

Complexité de la construction de l'architecture Une architecture de calcul est utilisable s'il est raisonnable de la construire. La proposition 5 permet d'éliminer certains tests lors de l'application des règles de réécriture et la proposition 6 fournit des majorants sur la complexité de construction de cette architecture.

Proposition 5. *Sauf en ce qui concerne S^1_Σ, les pré-conditions des règles de réécriture sont toujours satisfaites.*

Proposition 6. *La complexité en temps et en espace de l'application des règles de réécriture est respectivement $O(|U| \cdot (1+|P|)^2)$ et $O(|U| \cdot (1+|P|))$.*

3.2 Vers des MCDAG

Les règles de réécriture nous permettent de transformer le nœud de calcul initial multi-opérateur n_0 en un DAG de nœuds de calcul mono-opérateur de type $(\max_S, +, N)$, $(\sum_S, \times, N)$, $(\emptyset, \times, N)$ ou $\phi \in P \cup U$. Pour les nœuds de type $(\max_S, +, N)$ ou $(\sum_S, \times, N)$, il est alors possible d'utiliser la liberté dans l'ordre d'élimination. Pour cela, les techniques classiques de construction d'*arbres de jonction* peuvent être utilisées puisque, d'une part, $(\mathbb{R}, \max, +)$ et $(\mathbb{R}, +, \times)$ sont des semi-anneaux commutatifs et, d'autre part, il n'y a aucune contrainte sur l'ordre d'élimination à l'intérieur de ces nœuds (la seule légère différence avec les arbres de jonction classiques est que seul un sous-ensemble des variables impliquées dans le nœud peut avoir à être éliminé, mais il est aisé de traiter cette situation).
Pour obtenir une bonne décomposition pour les nœuds n de type $(\max_S, +, N)$ ou $(\sum_S, \times, N)$, on peut construire un arbre de jonction pour éliminer S de l'hyper-graphe $G = (sc(N), \{sc(n') \mid n' \in N\})$. La largeur induite optimale qui peut être obtenue pour n est $w(n) = w_{G,S}$, la largeur induite de G pour l'élimination des variables de S.[4] La largeur induite de l'architecture MCDAG est alors définie par $w_{mcdag} = \max_{n \in \mathfrak{N}} w(n)$, où $\mathfrak{N}$ est l'ensemble des nœuds de type $(\max_S, +, N)$ ou $(\sum_S, \times, N)$.
Après la décomposition de chaque nœud de calcul mono-opérateur, on obtient un DAG de *clusters*, que nous appelons *Multi-operator Cluster DAG* (MCDAG). La définition suivante est plus générale que nécessaire dans le cas des diagrammes d'influence, mais une telle généricité est utile pour la discussion de la section 6.

Definition 3. *Un MCDAG est un DAG où chaque sommet c (appelé cluster) est étiqueté par quatre éléments : (1) un ensemble de variables $V(c)$, (2) un ensemble de fonctions locales $\Psi(c)$ prenant leurs valeurs dans un ensemble E, (3) un ensemble de clusters fils $Sons(c)$ et (4) un couple $(\oplus_c, \otimes_c)$ d'opérateurs sur E tel que $(E, \oplus_c, \otimes_c)$ soit un semi-anneau commutatif.*

Definition 4. *La valeur d'un cluster c d'un MCDAG est donnée par :*

$$val(c) = \bigoplus_{V(c)-V(pa(c))}^{c} \Big(\big(\bigotimes_{\psi \in \Psi(c)}^{c} \psi \big) \otimes_c \big(\bigotimes_{s \in Sons(c)}^{c} val(s) \big) \Big)$$

4. Pour des nœuds de type $(\max_S, +, N)$ qui sont en fait toujours de la forme $(\max_S, +, \{(\emptyset, \times, N'), N' \in \mathfrak{N}\})$, de meilleures décompositions peuvent être obtenues en considérant un autre hyper-graphe. En fait, pour chaque $N' \in \mathfrak{N}$, il existe un nœud unique $n \in N'$, noté $N'[u]$, tel que n ou un de ses fils implique au moins une fonction d'utilité. Il est alors préférable de considérer l'hyper-graphe $G' = (sc(N), \{sc(N'[u]) \mid N' \in \mathfrak{N}\})$. Intuitivement, cela permet de se rendre compte que, par exemple, si on élimine x en premier dans un nœud de la forme $(\max_{xy}, +, N) = (\max_{xy}, +, \{(\emptyset, \times, U_{y,z}), (\emptyset, \times, \{n_z, U_{x,y}\}), (\emptyset, \times, \{n_z, U_x\})\})$, seules deux variables (x et y) devront être considérées, puisque n_z est un facteur de $U_{x,y}$ et de U_x. Nous ne développons pas ce point technique plus avant ici.

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

Grâce à la proposition 7, travailler sur des MCDAG est suffisant pour résoudre des diagrammes d'influence.

Proposition 7. *La valeur du MCDAG obtenue après avoir décomposé la macrostructure est égale à l'utilité espérée maximale. De plus, pour chaque variable de décision D_k, l'ensemble des règles de décision optimale pour D_k dans le diagramme d'influence est égal à l'ensemble des règles de décision optimale pour D_k dans le MCDAG.*

3.3 Réunir certains calculs

Il peut exister dans le MCDAG des clusters qui réalisent exactement les mêmes calculs, même s'ils proviennent de nœuds de calcul différents. Par exemple, un nœud de calcul $n_1 = (\sum_{x,y}, \times, \{P_x, P_{y|x}, U_{y,z})$ peut être décomposé pour donner les clusters $c_1 = (\{x\}, \{P_x, P_{y|x}\}, \emptyset, (+, \times))$ et $c'_1 = (\{y\}, \{U_{y,z}\}, \{c'_1\}, (+, \times))$. Un nœud de calcul $n_2 = (\sum_{x,y}, \times, \{P_x, P_{y|x}, U_{y,t})$ peut être décomposé pour donner les clusters $c_2 = (\{x\}, \{P_x, P_{y|x}\}, \emptyset, (+, \times))$ et $c'_2 = (\{y\}, \{U_{y,t}\}, \{c'_2\}, (+, \times))$. Comme $c_1 = c_2$, des calculs peuvent être évités en réunissant les deux clusters dans le MCDAG. Détecter des clusters identiques est cependant moins aisé que détecter des nœuds de calcul identiques.

3.4 Résumé

Trois étapes sont nécessaires pour construire l'architecture de calcul. D'abord, le nœud de calcul initial multi-opérateur est transformé en un DAG de nœuds de calcul mono-opérateur (via des règles de réécriture correctes). Puis, chaque nœud de calcul est décomposé via une classique construction d'arbre de jonction. Cela nous donne un MCDAG dans lequel certains clusters peuvent être réunis.

4 Un algorithme de type VE sur des MCDAG

Définir un algorithme de type VE sur un MCDAG est simple. La seule différence avec les algorithmes de type VE existants est l'aspect multi-opérateur pour les opérateurs de combinaison et d'élimination utilisés. Comme dans les architectures de calcul classiques, les nœuds envoient des messages à leurs parents. Quand un nœud c a reçu tous les messages $val(s)$ de ses fils, il peut calculer sa propre valeur $val(c) = \oplus_{cV(c)-V(pa(c))} \left(\left(\otimes_{c\psi \in \Psi(c)} \psi \right) \otimes_c \left(\otimes_{cs \in Sons(c)} val(s) \right) \right)$ et l'envoyer à ses parents. Les messages vont des feuilles vers la racine et la valeur calculée par le cluster racine est l'utilité espérée maximale

5 Comparaison avec les architectures de calcul existantes

Comparés aux architectures de calcul existantes sur les diagrammes d'influence, les MCDAG peuvent être exponentiellement plus efficaces en réduisant fortement la largeur induite contrainte (voir la section 2.3), grâce (1) au mécanisme de duplication, (2) à l'analyse des libertés supplémentaires dans l'ordre d'élimination et (3) à l'utilisation des conditions de normalisation. Sur ces trois points, on peut faire le lien avec des travaux existants :

— L'idée qui se trouve derrière le mécanisme de duplication est l'utilisation toutes les décompositions (indépendances) présentes dans un diagramme d'influence. Un diagramme d'influence exprime en fait des indépendances, d'une part sur la distribution globale de probabilité $P(C \mid D)$ et d'autre part sur la fonction globale d'utilité. Les MCDAG utilisent séparément ces deux types d'indépendance, alors qu'une approche à base de potentiels utilise une sorte de relation d'indépendance *mixte* plus faible.

— Relâcher les contraintes sur l'ordre d'élimination peut être mis en relation avec la notion classique d'information *pertinente* pour les variables de décision. Avec les MCDAG, cette notion n'est pas utilisée uniquement pour des raisons de concision des règles de décision. Elle est aussi utilisée pour révéler des libertés dans l'ordre d'élimination et réduire ainsi la complexité temporelle. Notons que certaines de ces libertés sont obtenues par synergie avec le mécanisme de duplication qui ne peut pas être utilisé avec les potentiels.

— Grâce à la règle de simplification S^1_Σ, les conditions de normalisation nous permettent non seulement d'éviter des calculs inutiles, mais aussi d'améliorer l'architecture de calcul (S^1_Σ peut indirectement relâcher certaines contraintes sur l'ordre d'élimination). Ceci est plus puissant que les architectures LP [6], qui n'utilisent que le premier point durant la phase de transmission de messages. Notons qu'avec les MCDAG, une fois que le DAG des nœuds de calcul est construit, il n'y a plus de conditions de normalisation à utiliser.

Finalement, l'architecture MCDAG est toujours meilleure que les schémas existants en termes de largeur induite contrainte, comme le met en évidence le théorème 1.

Theorem 1. *Soit $w_{G_p}(\preceq_p)$ la largeur induite contrainte associée à l'approche à base de potentiels (voir la section 2.2) et w_{mcdag} la largeur induite associée au MCDAG (voir la section 3.2). On a : $w_{mcdag} \leq w_{G_p}(\preceq_p)$.*

L'architecture MCDAG va donc à l'encontre d'une croyance courante selon laquelle utiliser des potentiels et des opérations de division est indispensable pour résoudre des diagrammes d'influence avec des algorithmes de type VE.

6 Extensions possibles

L'architecture de calcul MCDAG a en fait été développée dans un cadre algébrique générique qui inclut les diagrammes d'influence. Ce cadre, appelé *réseaux de Plausibilité-Faisabilité-Utilité* (PFU) [9], est un cadre générique pour la décision séquentielle avec éventuellement incertitudes (partie plausibilité), asymétries dans le pro-

cessus de décision (partie faisabilité) et utilités (partie utilité). Le cadre PFU couvre des formalismes allant des *formules booléennes quantifiées* (QBF) ou des *réseaux bayésiens* aux *problèmes de satisfaction de contraintes stochastiques* (SCSP) et définit de nouveaux formalismes comme les *diagrammes d'influence possibilistes*. Cette couverture est possible parce que les questions posées dans de nombreux formalismes existants se réduisent souvent à une séquence d'éliminations (utilisant éventuellement divers opérateurs d'élimination) sur une combinaison de fonctions locales (utilisant éventuellement divers opérateurs de combinaison). De telles séquences, dont l'équation 1 est un exemple, peuvent être structurées en utilisant des règles de réécriture similaires à celles présentées, qui utilisent activement les propriétés algébriques des opérateurs impliqués.
Grâce à la nature générique du cadre PFU, étendre le travail précédent à une version possibiliste des diagrammes d'influence est trivial. Si on utilise l'utilité espérée possibiliste pessimiste [2], l'utilité optimale peut être définie par (les distributions de probabilité P_i deviennent des distributions de possibilité et les utilités U_i deviennent des degrés de préférence dans $[0, 1]$) :

$$\min_{I_0} \max_{D_1} \ldots \min_{I_{n-1}} \max_{D_n} \min_{I_n} (\max(\max_{P_i \in P} (1 - P_i), \min_{U_i \in U} U))$$

Cette séquence d'éliminations peut être structurée et calculée via des MCDAG. La seule différence est que $\times$ devient $\max$ et $\sum$ et $+$ deviennent $\min$ dans les règles de réécriture. Les nœuds de calcul sont alors de type $(\min, \max, N)$, $(\max, \min, N)$, ou $(\emptyset, \max, N)$ et les clusters dans le MCDAG utilisent $(\oplus_c, \otimes_c) = (\min, \max)$, $(\oplus_c, \otimes_c) = (\max, \min)$ ou $(\oplus_c, \otimes_c) = (\emptyset, \max)$.

7 Conclusion

Pour résoudre des diagrammes d'influence, utiliser des potentiels permet de réutiliser les algorithmes existants à base d'élimination de variables, mais peut être exponentiellement sous-optimal. Le point crucial est que tirer avantage de la nature composite de modèles graphiques tels que les diagrammes d'influence et des propriétés algébriques des opérateurs de combinaison et d'élimination impliqués est essentiel pour obtenir une architecture de calcul efficace à base de calcul locaux. Pour cela, une solution consiste à bâtir une architecture composite impliquant différents opérateurs de combinaison et d'élimination. Le résultat est l'architecture MCDAG qui garantie une diminution de la largeur induite contrainte.
Les auteurs travaillent actuellement à obtenir des résultats formels et expérimentaux sur les MCDAG dans le cadre PFU (la construction automatique d'architectures de calcul MCDAG est actuellement implémentée). De futures directions de travail pourraient être d'abord l'adaptation de l'architecture de calcul MCDAG au cas des diagrammes d'influence à *mémoire limitée* (LIMID) [5], puis l'utilisation de cette architecture dans un contexte de résolution approchée.

Références

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

[2] 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.

[3] 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.

[4] 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.

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

[6] 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)*, Stockholm, Sweden, 1999.

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

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

[9] C. Pralet, G. Verfaillie, and T. Schiex. An Algebraic Graphical Model for Decision with Uncertainties, Feasibilities, and Utilities. Technical report, LAAS-CNRS, 2005. http ://www.laas.fr/∼cpralet/praletverfschiex.ps.

[10] C. Pralet, G. Verfaillie, and T. Schiex. From Influence Diagrams to Multioperator Cluster DAGs : Extended version. Technical report, LAAS-CNRS, 2006. http ://www.laas.fr/∼cpralet/uai06ext.ps.

[11] R. Shachter. Evaluating Influence Diagrams. *Operations Research*, 34(6) :871–882, 1986.

[12] P. Shenoy. Valuation-based Systems for Discrete Optimization. In *Proc. of the 6th International Conference on Uncertainty in Artificial Intelligence (UAI-90)*, pages 385–400, Cambridge, MA, USA, 1990.

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