From influence diagrams to multioperator cluster DAGs

Cédric Pralet
LAAS-CNRS
Toulouse, France

Gérard Verfaillie
ONERA
Toulouse, France

Thomas Schiex
INRA
Castanet Tolosan, France

Abstract

There exist several 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 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.

1 INTRODUCTION

Since the first algorithms based on decision trees or arc-reversal operations [Shachter, 1986], several exact methods have been proposed to solve influence diagrams using local computations, such as the ones based on the Shenoy-Shafer, the HUGIN, or the Lazy Propagation architectures [Shenoy, 1992; Jensen *et al.*, 1994; Madsen and Jensen, 1999]. These methods have succeeded in adapting classical variable elimination techniques (which are basically designed to compute one type of marginalization on a combination of local functions with only one type of combination operator), in order to handle the multiple types of information (probabilities and utilities), the multiple types of marginalizations (sum and max), and the multiple types of combination ($\times$ for probabilities, $+$ for utilities) involved in an influence diagram. The key mechanism used for such an extension consists in using elements known as *potentials* [Ndilikilikesha, 1994].

In this paper, we define a new architecture, called the Multi-operator Cluster DAG (MCDAG) architecture, which does not use potentials, but still benefits from variable elimination. Compared to existing schemes, MCDAGs actively exploit the composite nature of influence diagrams. We first present the potential approach and motivate the need for a new architecture (Section 2). Then, MCDAGs are introduced (Section 3) and a variable elimination algorithm is defined (Section 4). Finally, this work is compared with existing approaches (Section 5) and extended to other frameworks (Section 6). The proofs are available in [Pralet *et al.*, 2006].

2 MOTIVATIONS

Notations and definitions An *influence diagram* [Howard and Matheson, 1984] is a composite graphical model defined on three sets of variables organized in a Directed Acyclic Graph (DAG) G: (1) a set C of *chance variables* $x \in C$, for each of which a conditional probability distribution $P_{x \mid pa(x)}$ on x given its parents in G is specified; (2) a set $D = \{D_1, \ldots, D_n\}$ (indices represent the order in which decisions are made) of *decision variables* $x \in D$, for each of which $pa(x)$ is the set of variables observed before decision x is made; (3) a set Γ of *utility variables* $u \in \Gamma$, each of which is associated with a utility function $U_{pa(u)}$ on $pa(u)$ (and utility variables are leaves in the DAG).

The set of conditional probability distributions (one for each $x \in C$) is denoted P and the set of utility functions (one for each $u \in \Gamma$) is denoted U. Each function $\phi \in P \cup U$ holds on a set of variables $sc(\phi)$ called its scope, and is consequently called a *scoped function* ($sc(P_{x \mid pa(x)}) = \{x\} \cup pa(x)$ and $sc(U_{pa(u)}) = pa(u)$). The set of chance variables observed before the first decision is denoted I_0, the set of chance variables observed between decisions D_k and D_{k+1} is denoted I_k, and the set of chance variables unobserved before the last decision is denoted I_n. We use $dom(x)$ to denote the domain of a variable $x \in C \cup D$, and by extension, for $W \subset C \cup D$, $dom(W) = \prod_{x \in W} dom(x)$.

The usual problem associated with an influence diagram is to find *decision rules* maximizing the expected utility (a decision rule for a decision D_k is a function associating a value in $dom(D_k)$ with any assignment of the variables observed before making decision D_k) As shown in [Jensen *et al.*, 1994], this is equivalent to computing optimal decision rules for the quantity

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

2.1 THE POTENTIAL APPROACH

With this approach, Equation 1 is reformulated using so-called *potentials* in order to use only one combination and one marginalization operator. A potential on a set of variables W is a pair $\pi_W = (p_W, u_W)$, where p_W and u_W are respectively a positive real function and a real function, whose scopes are included in W. The initial conditional probability distributions $P_i \in P$ are transformed into potentials $(P_i, 0)$, whereas the initial utility functions $U_i \in U$ are transformed into potentials $(1, U_i)$. On these potentials, a *combination* operation $\otimes$ and a *marginalization* (or *elimination*) operation $\downarrow$ are defined:

- the combination of $\pi_{W_1} = (p_{W_1}, u_{W_1})$ and $\pi_{W_2} = (p_{W_2}, u_{W_2})$ is the potential on $W_1 \cup W_2$ given by $\pi_{W_1} \otimes \pi_{W_2} = (p_{W_1} \times p_{W_2}, u_{W_1} + u_{W_2})$;
- the marginalization of $\pi_W = (p_W, u_W)$ onto $W_1 \subset C$ equals $\pi_W^{\downarrow W_1} = \left(\sum_{W - W_1} p_W, \frac{\sum_{W - W_1} p_W u_W}{\sum_{W - W_1} p_W} \right)$ (with the convention $0/0 = 0$), whereas the marginalization of $\pi_W = (p_W, u_W)$ onto $W_1 \subset D$ is given by $\pi_W^{\downarrow W_1} = (p_W, \max_{W_1} u_W)$.

Solving the problem associated with an influence diagram is then equivalent to computing $Q = \left(\left(\cdots \left(\left(\pi_{C \cup D}{}^{\downarrow I_n} \right)^{\downarrow D_n} \right)^{\downarrow I_{n-1}} \cdots \right)^{\downarrow D_1} \right)^{\downarrow I_0}$, where $\pi_{C \cup D} = (\otimes_{P_i \in P} (P_i, 0)) \otimes (\otimes_{U_i \in U} (1, U_i))$ is the combination of the initial potentials. As $\otimes$ and $\downarrow$ satisfy the Shenoy-Shafer axioms defined in [Shenoy, 1991], Q can be computed using usual variable elimination algorithms [Jensen *et al.*, 1994]. This explains why potentials are used by existing architectures for local computations on influence diagrams: Shenoy-Shafer, HUGIN, or Lazy Propagation (LP).[1]

2.2 QUANTIFYING THE COMPLEXITY

In the case of influence diagrams, the alternation of sum and max marginalizations, which do not generally commute, prevents from eliminating variables in any order. The complexity can then be quantified using the *constrained induced-width* [Jensen *et al.*, 1994; Park and Darwiche, 2004] (instead of the induced-width [Dechter and El Fattah, 2001]).

Definition 1. *Let $G = (V_G, H_G)$ be a hypergraph*[2] *and let $\preceq$ be a partial order on V_G. The constrained induced-width of G with constraints on the elimination order given by $\preceq$ ("$x \prec y$" stands for "y must be eliminated before x") is a parameter denoted $w_G(\preceq)$. It is defined as $w_G(\preceq) = \min_{o \in lin(\preceq)} w_G(o)$, $lin(\preceq)$ being the set of linearizations of $\preceq$ to a total order on V_G and $w_G(o)$ being the induced-width of G for the elimination order o (i.e. the size of the largest hyperedge created when eliminating variables in the order given by o).*

The constrained induced-width can be used to give an upper bound on the complexity of existing algorithms which use potentials. Let $G_p = (C \cup D, \{sc(P_i) | P_i \in P\} \cup \{sc(U_i) | U_i \in U\})$ be the hypergraph corresponding to the "untyped" influence diagram. Let $\preceq_p$ be the partial order defined by $I_0 \prec_p D_1$, $(I_k \neq \emptyset) \rightarrow (D_k \prec_p I_k \prec_p D_{k+1})$, and $D_n \prec_p I_n$. Finally, let d be the maximum size of the variables domains. Then, with classical approaches based on potentials and *strong junction trees* [Jensen *et al.*, 1994], which are junction trees with constraints on the marginalization order, the theoretical complexity is $O(|P \cup U| \cdot d^{1 + w_{G_p}(\preceq_p)})$ (the number of elements of a finite set E is denoted $|E|$).

2.3 DECREASING THE CONSTRAINED INDUCED-WIDTH

The constrained-induced width is a guideline to show how the complexity can be decreased. In this direction, one can work on the two parameters on which it depends: the partial order $\preceq$, and the hypergraph G.

Weakening the partial order $\preceq$

Proposition 1. *Let $G = (V_G, H_G)$ be a hypergraph and let $\preceq_1$, $\preceq_2$ be two partial orders on V_G such that $\forall (x, y) \in V_G \times V_G$, $(x \preceq_2 y) \rightarrow (x \preceq_1 y)$ ($\preceq_2$ is weaker than $\preceq_1$). Then, $w_G(\preceq_1) \geq w_G(\preceq_2)$.*

Proposition 1 means that if one weakens $\preceq$, i.e. if one reveals some extra freedoms in the elimination order (e.g. by proving that some marginalizations with sum and max can commute), then the theoretical complexity may decrease. Though such a technique is known to be useless in contexts like Maximum A Posteriori hypothesis [Park and Darwiche, 2004], where there is only one alternation of max and sum marginalizations,

[1] The LP architecture actually uses potentials defined as pairs of set of functions (instead of pairs of functions).

[2] This means that V_G is the set of variables (or vertices), and H_G is a set of hyperedges on V_G, i.e. a subset of 2^{V_G}.

it can lead to an exponential gain as soon as there are more than two levels of alternation.

Indeed, assume that one wants to compute $\max_{x_1,\ldots,x_n} \sum_y \max_{x_{n+1}} P_y \cdot \left(U_{x_1,y} + \sum_{i\in[1,n]} U_{x_i,x_{n+1}}\right)$. On one hand, using $\preceq_1$ defined by $\{x_1,\ldots,x_n\} \prec_1 y \prec_1 x_{n+1}$ provides us with the constrained induced-width $w_G(\preceq_1) = n$, since x_{n+1} is then necessarily eliminated first. On the other hand, the scopes of the functions involved enable us to infer that with $\preceq_2$ defined by $x_1 \prec_2 y$, one is guaranteed to compute the same value, since y is "linked" only with x_1. The constrained induced-width is then $w_G(\preceq_2) = 1$, e.g. with the elimination order $x_1 \prec y \prec x_{n+1} \prec x_n \prec \ldots \prec x_2$. Therefore, the theoretical complexity decreases from $O((n+2)\cdot d^{n+1})$ to $O((n+2)\cdot d^2)$, thanks to the weakening of the partial order (the $(n+2)$ factor corresponds to the number of scoped functions).

Working on the hypergraph The second possible mechanism is to work on the hypergraph G, either by eliminating so-called barren variables (computing $\sum_x P_{x\,|\,pa(x)}$ is useless because of normalization), or by better decomposing the problem. To illustrate the latter, assume that one wants to compute $\max_{x_1,\ldots,x_n} \sum_y P_y \cdot (U_{y,x_1} + \ldots + U_{y,x_n})$. The basic hypergraph $G_1 = (\{x_1,\ldots,x_n,y\},\{\{y,x_1\},\ldots,\{y,x_n\}\})$, together with $\preceq_1$ defined by $\{x_1,\ldots,x_n\} \prec_1 y$, gives a theoretical complexity $O((n+1)\cdot d^{w_{G_1}(\preceq_1)+1}) = O((n+1)\cdot d^{n+1})$. However, one can write

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

Thus, an implicit *duplication* of y makes the complexity decrease to $O((n+1)d^2) = O((n+1)d^{1+w_{G_2}(\preceq_2)})$, where G_2 is the hypergraph defined by the variables $\{x_1,\ldots,x_n,y^{(1)},\ldots,y^{(n)}\}$ and by the hyperedges $\{\{x_1,y^{(1)}\},\ldots,\{x_n,y^{(n)}\}\}$, and where $\preceq_2$ is given by $x_1 \prec_2 y^{(1)}$, …, $x_n \prec_2 y^{(n)}$. This method, which uses the property $\sum_S (U_1+U_2) = (\sum_S U_1) + (\sum_S U_2)$, duplicates variables "quantified" with $\sum$, so that computations become more local. Proposition 2 shows the possible exponential gain obtained by duplication.

Proposition 2. *Let ϕ_{x,S_i} be a scoped function of scope $\{x\}\cup S_i$ for any $i \in [1,m]$. The raw computation of $\sum_x (\phi_{x,S_1} + \ldots + \phi_{x,S_m})$ always requires more sums than the raw computation of $(\sum_x \phi_{x,S_1}) + \ldots + (\sum_x \phi_{x,S_m})$. Moreover, the raw computation of $\sum_x (\phi_{x,S_1} + \ldots + \phi_{x,S_m})$ results in a complexity $O(m\cdot d^{1+|S_1\cup\ldots\cup S_m|})$, whereas the raw computation of the m quantities in the set $\{\sum_x \phi_{x,S_i} \,|\, 1 \le i \le m\}$ results in a complexity $O(m\cdot d^{1+\max_{i\in[1,m]}|S_i|})$.* □

Why not use potentials? Though weakening the constraints on the elimination order could be done with potentials, the duplication mechanism cannot be used if potentials are. Indeed, one cannot write $(\pi_{W_1}\otimes\pi_{W_2})^{\downarrow W_3} = (\pi_{W_1}^{\downarrow W_3}) \otimes (\pi_{W_2}^{\downarrow W_3})$ even if $W_3 \subset C$. Moreover, the duplication mechanism itself may weaken some constraints on the elimination order. Consequently, we introduce a new architecture which does not use potentials to solve influence diagrams using local computations. This architecture is called the *Multi-operator Cluster DAG* (MCDAG) architecture.

3 THE MCDAG ARCHITECTURE

3.1 MACROSTRUCTURING AN INFLUENCE DIAGRAM

The first step to build the MCDAG architecture is to analyze the *macrostructure* of the influence diagram, by detecting the possible reordering freedoms in the elimination order, while using the duplication technique and the normalization conditions on conditional probability distributions. This macrostructure is represented with a DAG of *computation nodes.*

Definition 2. *A computation node n is:*

- *either a scoped fonction ϕ in $P\cup U$; in this case, the value of n is given by $val(n) = \phi$, and its scope is given by $sc(n) = sc(\phi)$;*
- *or a triple $(Sov, \circledast, N)$, where Sov is a sequence of operator-variables pairs, $\circledast$ is an associative and commutative operator with an identity, and where N is a set of computation nodes; in this case, the value of n is given by $val(n) = Sov\,(\circledast_{n'\in N}\, val(n'))$, and its scope is given by $sc(n) = (\cup_{n'\in N}\, sc(n')) - \{x \,|\, op_x \in Sov\}$.*

Informally, a computation node $(Sov, \circledast, N)$ defines a sequence of marginalizations on a combination of computation nodes with a specific operator. It can be represented as in Figure 1. Given a set of computation nodes N, we define N^{+x} (resp N^{-x}) as the set of nodes of N whose scope contains x (resp. does not contain x): $N^{+x} = \{n \in N \,|\, x \in sc(n)\}$ (resp. $N^{-x} = \{n \in N \,|\, x \notin sc(n)\}$).

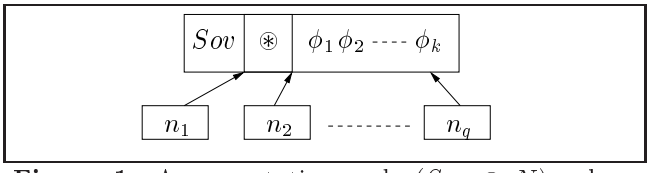

Figure 1: A computation node $(Sov, \circledast, N)$, where $N \cap (P\cup U) = \{\phi_1,\ldots,\phi_k\}$ and $N - (P\cup U) = \{n_1,\ldots,n_q\}$.

3.1.1 From influence diagrams to computation nodes

Without loss of generality, we assume that $U \neq \emptyset$ (if this is not the case, one can add $U_0 = 1$ to U).

Proposition 3. *Let Sov_0 be the initial sequence $\sum_{I_0} \max_{D_1} \dots \sum_{I_{n-1}} \max_{D_n} \sum_{I_n}$ of operator-variables pairs defined by the influence diagram. The value of Equation 1 is equal to the value of the computation node $n_0 = (Sov_0, +, \{(\emptyset, \times, P \cup \{U_i\}), U_i \in U\})$.*

For the influence diagram associated with the computation of $\max_d \sum_{r_2,r_1} P_{r_1} \cdot P_{r_2|r_1} \cdot (U_{d,r_1} + U_{d,r_2} + U_d)$, n_0 corresponds to the first computation node in Figure 2.

3.1.2 Macrostructuring the initial node

In order to exhibit the macrostructure of the influence diagram, we analyze the sequence of computations performed by n_0. To do so, we successively consider the eliminations in Sov_0 from the right to the left and use three types of *rewriting rules*, preserving nodes values, to make the macrostructure explicit: (1) *decomposition rules*, which decompose the structure using namely the duplication technique; (2) *recomposition rules*, which reveal freedoms in the elimination order; (3) *simplification rules*, which remove useless computations from the architecture, by using normalization conditions. Rewriting rules are presented first for the case of sum-marginalizations, and then for the case of max-marginalizations. A rewriting rule may be preceded by some preconditions for it to be applicable.

Rewriting rules for $\sum_x$ When a sum-marginalization must be performed, a decomposition rule D_Σ, a recomposition rule R_Σ, and two simplification rules S^1_Σ and S^2_Σ are used. These are illustrated in Figure 2, which corresponds to the influence diagram example introduced in 3.1.1.

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

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

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

Example In the example of Figure 2, the first rule to be applied is the decomposition rule D_Σ, which treats the operator-variable pair $\sum_{r_1}$.[3] Such a rule uses the duplication mechanism and the distributivity property of $\times$ over $+$. It provides us with a DAG of computation nodes. It is a DAG since common computation nodes are merged (and it is not hard to detect such nodes when applying the rules). Then, D_Σ can be applied again for $\sum_{r_2}$. One can infer from the obtained architecture that there is no reason for r_1 to be eliminated before r_2. Using the recomposition rule R_Σ makes this clear in the structure. Basically, R_Σ uses the distributivity of $\times$ over $+$. Last, applying S^1_Σ and S^2_Σ, which use the normalization of conditional probability distributions, simplifies some nodes in the architecture. In the end, computations

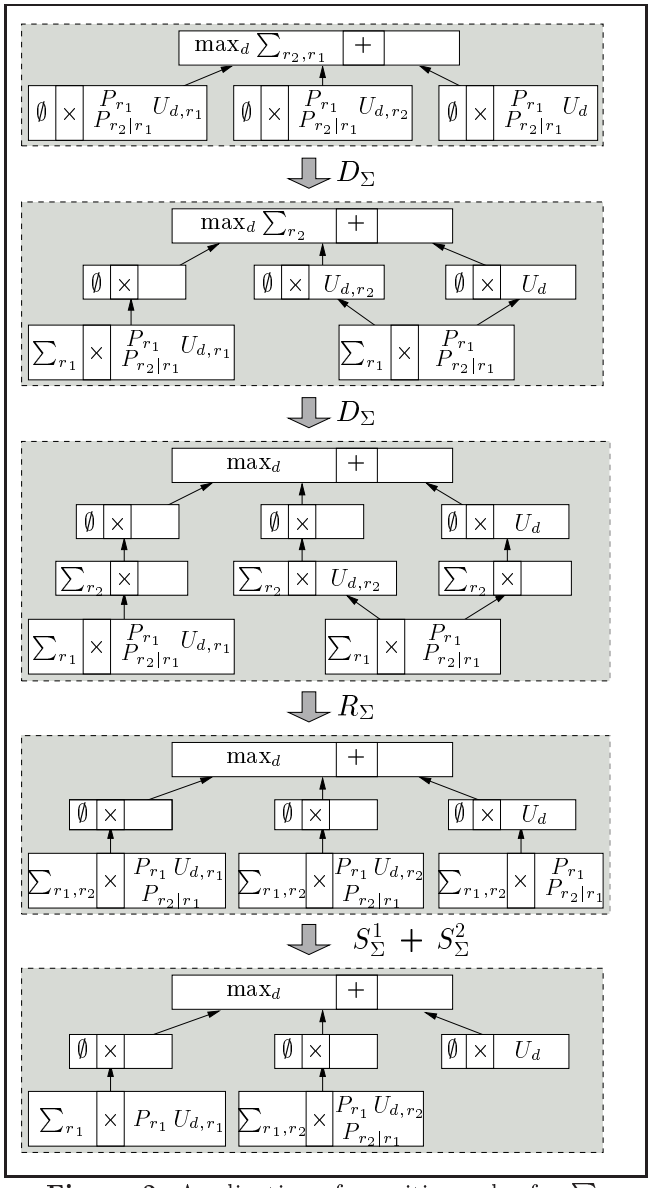

Figure 2: Application of rewriting rules for $\sum$.

[3] For the example needs, r_1 is eliminated before r_2, even if the opposite would be more efficient.

can be made by considering at the most two variables simultaneously (if one eliminates r_1 first in the node $(\sum_{r_1,r_2}, \times, \{P_{r_1}, P_{r_2|r_1}, U_{d,r_2}\})$, whereas with a potential approach, considering three variables simultaneously would have been necessary (because r_1 would be involved in the potentials $(P_{r_1}, 0), (P_{r_2|r_1}, 0), (1, U_{d,r_1})$ if eliminated first, and r_2 would be involved in the potentials $(P_{r_2|r_1}, 0), (1, U_{d,r_2})$ if eliminated first).

Rewriting rules for $\max_x$ When a max-marginalization must be performed, a decomposition rule $D_{\max}$ and a recomposition rule $R_{\max}$ are used (there is no simplification rule since there is no normalization condition to use for decision variables). These rules, which are a bit more complex than the previous ones, are illustrated in Figure 3, which corresponds to the influence diagram $\max_{d_1} \sum_{r_2} \max_{d_2} \sum_{r_1} \max_{d_3} P_{r_1} \cdot P_{r_2|r_1} \cdot (U_{d_1} + U_{d_2,d_3} + U_{r_2,d_1,d_3} + U_{r_1,d_2})$.

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

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

Example In the example of Figure 3, one first applies the decomposition rule $D_{\max}$, in order to treat the operator-variable pair $\max_{d_3}$. Such a rule uses first the monotonicity of $+$ ($\max(a+b, a+c) = a + \max(b, c)$), and then both the distributivity of $\times$ over $+$ and the monotonicity of $\times$ (so as to write things like $\max_{d_3}((P_{r_1} \cdot P_{r_2|r_1} \cdot U_{d_2,d_3}) + (P_{r_1} \cdot P_{r_2|r_1} \cdot U_{r_2,d_1,d_3})) = P_{r_1} \cdot P_{r_2|r_1} \cdot \max_{d_3}(U_{d_2,d_3} + U_{r_2,d_1,d_3})$). Then, D_Σ can be used for $\sum_{r_1}$, and $D_{\max}$ can be used for $\max_{d_2}$. After those steps, the recomposition rule $R_{\max}$, which uses the monotonicities of $\times$ and $+$, reveals that the elimination order between d_2 and d_3 is actually free. This was not obvious from the initial Sov sequence. The approach using potentials is unable to make such freedoms explicit, which may induce exponential increase in complexity as shown in 2.3.

Order of application of the rules A chaotic iteration of the rules does not converge, since e.g., rules

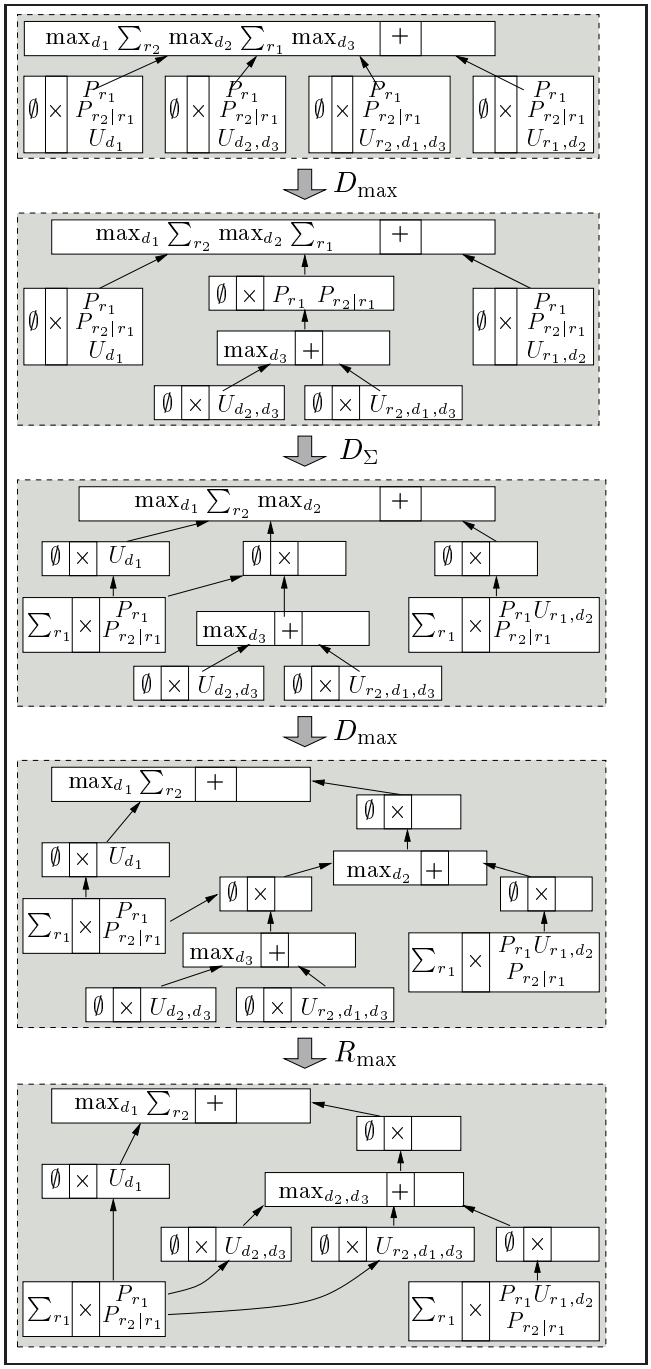

Figure 3: Application of rewriting rules for max (the application of the rules may create nodes looking like $(\emptyset, \times, \{n\})$, which perform no computations; these nodes can be eliminated at a final step).

$D_{\max}$ and $R_{\max}$ may be infinitely alternately applied. As a result, we specify an order in which rules can be used to converge to a unique final DAG of computation nodes (we have implicitly used this order in the previous examples). We successively consider each operator-variable pair of the initial sequence Sov_0 from the right to the left (marginalizations like $\sum_{x_1,\ldots,x_n}$

can be split into $\sum_{x_1} \cdots \sum_{x_n}$).

If the rightmost marginalization in the *Sov* sequence of the root node is $\sum_x$, then rule D_Σ is applied once. It creates new grandchildren nodes for the root, for each of which, we try to apply rule R_Σ in order to reveal freedoms in the elimination order. If R_Σ is applied, this creates new computation nodes, on each of which simplification rules S^1_Σ and then S^2_Σ are applied (until they cannot be applied anymore).

If the rightmost marginalization in the *Sov* sequence of the root node is $\max_x$, then rule $D_{\max}$ is applied once. This creates a new child and a new grandchild for the root. For the created grandchild, we try to weaken constraints on the elimination order using rule $R_{\max}$.

The soundness of the macrostructure obtained is provided by the soundness of the rewriting rules:

Proposition 4. *Rewriting rules D_Σ, R_Σ, S^1_Σ, S^2_Σ, $D_{\max}$ and $R_{\max}$ are sound, i.e. for any of these rules $n_1 \rightsquigarrow n_2$, if the preconditions are satisfied, then $val(n_1) = val(n_2)$ holds. Moreover, rules $D_{\max}$ and $R_{\max}$ leave the set of optimal decision rules unchanged.*

Complexity issues An architecture is usable only if it is reasonable to build it. Proposition 5 makes it possible to save some tests during the application of the rewriting rules, and Proposition 6 gives upper bounds on the complexity.

Proposition 5. *Except for S^1_Σ, the preconditions of the rewriting rules are always satisfied.*

Proposition 6. *The time and space complexity of the application of the rewriting rules are $O(|U| \cdot (1+|P|)^2)$ and $O(|U| \cdot (1+|P|))$ respectively.*

3.2 TOWARDS MCDAGS

The rewriting rules enable us to transform the initial multi-operator computation node n_0 into a DAG of mono-operator computation nodes looking like $(\max_S, +, N)$, $(\sum_S, \times, N)$, $(\emptyset, \times, N)$, or $\phi \in P \cup U$. For nodes $(\max_S, +, N)$ or $(\sum_S, \times, N)$, it is time to use freedoms in the elimination order. To do so, usual junction tree construction techniques can be used, since on one hand, $(\mathbb{R}, \max, +)$ and $(\mathbb{R}, +, \times)$ are commutative semirings, and since on the other hand, there are no constraints on the elimination order inside each of these nodes (the only slight difference with usual junction trees is that only a subset of the variables involved in a computation node may have to be eliminated, but it is quite easy to cope with this).

To obtain a good decomposition for nodes n like $(\max_S, +, N)$ or $(\sum_S, \times, N)$, one can build a junction tree to eliminate S from the hypergraph $G = (sc(N), \{sc(n') \mid n' \in N\})$. The optimal induced-width which can be obtained for n is $w(n) = w_{G,S}$, the induced-width of G for the elimination of the variables in S.[4] The induced-width of the MCDAG architecture is then defined by $w_{mcdag} = \max_{n \in \mathfrak{N}} w(n)$, where $\mathfrak{N}$ is the set of nodes looking like $(\max_S, +, N)$ or $(\sum_S, \times, N)$.

After the decomposition of each mono-operator computation node, one obtains a *Multi-operator Cluster DAG*. The definition below is more general than needed for the influence diagram case, but such a genericity is useful for the discussion in Section 6.

Definition 3. *A Multi-operator Cluster DAG is a DAG where every vertex c (called a cluster) is labelled with four elements: a set of variables $V(c)$, a set of scoped functions $\Psi(c)$ taking values in a set E, a set of son clusters $Sons(c)$, and a couple $(\oplus_c, \otimes_c)$ of operators on E such that $(E, \oplus_c, \otimes_c)$ is a commutative semiring.*

Definition 4. *The value of a cluster c of a MCDAG is given by*

$$val(c) = \bigoplus_{c}_{V(c)-V(pa(c))} \left(\left(\bigotimes_{c}_{\psi \in \Psi(c)} \psi \right) \otimes_c \left(\bigotimes_{c}_{s \in Sons(c)} val(s) \right) \right)$$

The value of a MCDAG is the value of its root node.

Thanks to Proposition 7, working on MCDAGs is sufficient to solve influence diagrams.

Proposition 7. *The value of the MCDAG obtained after having decomposed the macrostructure is equal to the maximal expected utility. Moreover, for any decision variable D_k, the set of optimal decision rules for D_k in the influence diagram is equal to the set of optimal decision rules for D_k in the MCDAG.*

3.3 MERGING SOME COMPUTATIONS

There may exist MCDAG clusters performing exactly the same computations, even if the computation nodes they come from are distinct. For instance, a computation node $n_1 = (\sum_{x,y}, \times, \{P_x, P_{y|x}, U_{y,z})$ may be decomposed into clusters $c_1 = (\{x\}, \{P_x, P_{y|x}\}, \emptyset, (+, \times))$

[4] For nodes like $(\max_S, +, N)$, which actually always look like $(\max_S, +, \{(\emptyset, \times, N'), N' \in \mathfrak{N}\})$, better decompositions can be obtained by considering another hypergraph. In fact, for each $N' \in \mathfrak{N}$, there exists a unique node $n \in N'$, denoted $N'[u]$, such that n or its children involve at least one utility function. It is then better to consider the hypergraph $G' = (sc(N), \{sc(N'[u]) \mid N' \in \mathfrak{N}\})$. Intuitively, this enables to figure out that e.g., if one eliminates x first in a node like $(\max_{xy}, +, N) = (\max_{xy}, +, \{(\emptyset, \times, U_{y,z}), (\emptyset, \times, \{n_z, U_{x,y}\}), (\emptyset, \times, \{n_z, U_x\})\})$, only two variables ($x$ and y) must be considered, since n_z is a factor of both $U_{x,y}$ and U_x. We do not further develop this technical point here.

and $c'_1 = (\{y\}, \{U_{y,z}\}, \{c'_1\}, (+, \times))$. A computation node $n_2 = (\sum_{x,y}, \times, \{P_x, P_{y|x}, U_{y,t})$ may be decomposed into clusters $c_2 = (\{x\}, \{P_x, P_{y|x}\}, \emptyset, (+, \times))$ and $c'_2 = (\{y\}, \{U_{y,t}\}, \{c'_2\}, (+, \times))$. As $c_1 = c_2$, some computations can be saved by merging clusters c_1 and c_2 in the MCDAG. Detecting common clusters is not as easy as detecting common computation nodes.

3.4 SUMMARY

There are three steps to build the architecture. First, the initial multi-operator computation node is transformed into a DAG of mono-operator computation nodes (via sound rewriting rules). Then, each computation node is decomposed with a usual junction tree contruction. It provides us with a MCDAG, in which some clusters can finally be merged.

4 VARIABLE ELIMINATION ALGORITHM ON MCDAGs

Defining a variable elimination algorithm on a MCDAG is simple. The only difference with existing variable elimination algorithms is the multi-operator aspect for both the marginalization and the combination operators used. As in usual architectures, nodes send messages to their parents. Whenever a node c has received all messages $val(s)$ from its children, c can compute its own value $val(c) = \oplus_{cV(c)-V(pa(c))} \left(\left(\otimes_{c\psi \in \Psi(c)} \psi \right) \otimes_c \left(\otimes_{cs \in Sons(c)} val(s) \right) \right)$ and send it to its parents. As a result, messages go from leaves to root, and the value computed by the root is the maximal expected utility.

5 COMPARISON WITH EXISTING ARCHITECTURES

Compared to existing architectures on influence diagrams, MCDAGs can be exponentially more efficient by strongly decreasing the constrained induced-width (cf Section 2.3), thanks to (1) the duplication technique, (2) the analysis of extra reordering freedoms, and (3) the use of normalizations conditions. One can compare these three points with existing works:

- The idea behind duplication is to use all the decompositions (independences) available in an influence diagram. An influence diagram actually expresses independences on one hand on the global probability distribution $P(C \mid D)$, and on the other hand on the global utility function. MCDAGs separately use these two kinds of independences, whereas a potential approach uses a kind of weaker "mixed" independence relation.
- Weakening the constraints on the elimination order can be linked with the usual notion of *relevant* information for decision variables. With MCDAGs, this notion is not used only for decision rules conciseness reasons: it is also used to reveal reordering freedoms, which can decrease the temporal complexity. Note that some of the ordering freedom here is obtained by synergism with the duplication, which cannot be used with potentials.
- Thanks to simplification rule S^1_Σ, the normalization conditions enable us not only to avoid useless computations, but also to improve the architecture structure (S^1_Σ may indirectly weaken some constraints on the elimination order). This is stronger than Lazy Propagation architectures [Madsen and Jensen, 1999], which use the first point only, during the message passing phase. Note that with MCDAGs, once the DAG of computation nodes is built, there are no more normalization conditions to be used.

In the end, the MCDAG architecture is always better than existing schemes in terms of constrained induced-width, as Theorem 1 shows.

Theorem 1. *Let $w_{G_p}(\preceq_p)$ be the constrained induced-width associated with the potential approach (cf Section 2.2) and let w_{mcdag} be the induced-width associated with the MCDAG (cf Section 3.2). Then, $w_{mcdag} \leq w_{G_p}(\preceq_p)$.*

Last, the MCDAG architecture contradicts a common belief that using potentials and division operations is necessary to solve influence diagrams with variable elimination algorithms.

6 POSSIBLE EXTENSIONS

The MCDAG architecture has actually been developed in a generic algebraic framework which subsumes influence diagrams. This framework, called the Plausibility-Feasibility-Utility networks (PFUs) framework [Pralet *et al.*, 2005], is a generic framework for sequential decision making with possibly uncertainties (plausibility part), assymmetries in the decision process (feasibility part), and utilities. PFUs subsume formalisms from quantified boolean formulas or Bayesian networks to stochastic constraint satisfaction problems, and even define new frameworks like possibilistic influence diagrams. This subsumption is possible because the questions raised in many existing formalisms can often be reduced to a sequence of marginalizations (with possibly multiple marginalization operators) on a combination of scoped functions

(with possibly multiple combination operators). Such sequences, a particular case of which is Equation 1, can be structured using rewriting rules as the ones previously presented, which actively exploit the algebraic properties of the operators at stake.

Thanks to the generic nature of PFUs, extending the previous work to a possibilistic version of influence diagrams is trivial. If one uses the possibilistic pessimistic expected utility [Dubois and Prade, 1995], the optimal utility can be defined by (the probability distributions P_i become possibility distributions, and the utilities U_i become preference degrees in $[0,1]$):

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

The above sequence of marginalizations can be structured and computed via MCDAGs. The only difference is that $\times$ becomes max, and $\sum$ and $+$ become min in the rewriting rules. The computation nodes then look like $(\min, \max, N)$, $(\max, \min, N)$, or $(\emptyset, \max, N)$, and the clusters in the MCDAG use $(\oplus_c, \otimes_c) = (\min, \max)$, $(\oplus_c, \otimes_c) = (\max, \min)$, or $(\oplus_c, \otimes_c) = (\emptyset, \max)$.

7 CONCLUSION

To solve influence diagrams, using potentials allows one to reuse existing variable elimination schemes, but may be exponentially sub-optimal. The key point is that taking advantage of the composite nature of graphical models such as influence diagrams, and namely of the algebraic properties of the elimination and combination operators at stake, is essential to obtain an efficient architecture for local computations. To do so, a solution is to design a kind of composite architecture involving several elimination operators and several combination operators. The result is the MCDAG architecture, which guarantees a decrease in the constrained induced-width.

The authors are currently working to obtain more formal and experimental results on MCDAGs in the context of the PFU framework (the construction of MCDAG architectures is currently implemented). Future directions could be first to try and adapt the MCDAG architecture, and namely the duplication mechanism, to the case of Limited Memory Influence Diagrams (LIMIDs) [Lauritzen and Nilsson, 2001], and then to use the MCDAG architecture in the context of an approximate resolution.

References

[Dechter and El Fattah, 2001] R. Dechter and Y. El Fattah. Topological parameters for time-space tradeoff. *Artificial Intelligence*, 125(1-2):93–118, 2001.

[Dubois and Prade, 1995] 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*, pages 1925–1930, 1995.

[Howard and Matheson, 1984] R. Howard and J. Matheson. Influence Diagrams. In *Readings on the Principles and Applications of Decision Analysis*, pages 721–762. 1984.

[Jensen *et al.*, 1994] F. Jensen, F. Jensen, and S. Dittmer. From influence diagrams to junction trees. In *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pages 367–373, 1994.

[Lauritzen and Nilsson, 2001] S. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information, 2001.

[Madsen and Jensen, 1999] A. Madsen and F. Jensen. Lazy evaluation of symmetric bayesian decision problems. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 382–390, 1999.

[Ndilikilikesha, 1994] P. Ndilikilikesha. Potential influence diagrams. *International Journal of Approximated Reasoning*, 10(3):251–285, 1994.

[Park and Darwiche, 2004] J. Park and A. Darwiche. Complexity results and approximation strategies for map explanations. *Journal of Artificial Intelligence Reasearch*, 21:101–133, 2004.

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

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

[Shachter, 1986] R. Shachter. Evaluating influence diagrams. *Operations research*, 40:463–484, 1986.

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

[Shenoy, 1992] P.P. Shenoy. Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40(3):463–484, 1992.