Contextual Variable Elimination with Overlapping Contexts

Wannes Meert, Jan Struyf and Hendrik Blockeel
Katholieke Universiteit Leuven, Department of Computer Science, Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract

Belief networks (BNs) extracted from statistical relational learning formalisms often include variables with conditional probability distributions (CPDs) that exhibit a local structure (e.g, decision trees and noisy-or). In such cases, naively representing CPDs as tables and using a general purpose inference algorithm such as variable elimination (VE) results in redundant computation. Contextual variable elimination (CVE) partly addresses this problem by representing the BN in terms of smaller units called confactors. This leads to a more compact representation and faster inference. CVE requires that a variable's confactors are mutually-exclusive and exhaustive. We propose CVE-OC (CVE with overlapping contexts), which lifts these restrictions. This seemingly simple step shows to be powerful and allows for a more efficient encoding of confactors and a reduction of the computational cost. Experiments show that CVE-OC outperforms CVE on multiple problems.

1 Introduction

Belief networks (BNs) (Pearl, 1988) are a well-known formalism to represent probabilistic relations between variables. Their main advantage is that they allow one to graphically represent which variables are (in)dependent of which other variables. A BN takes the form of a directed acyclic graph in which the variables are nodes and the dependence information is encoded by means of the edges. The BN defines the variables' joint probability distribution, which is compactly represented as a product of factors. This product includes for each variable one factor, which is typically encoded as a table that represents the conditional probability distribution (CPD) of the variable given its parents in the BN. BN inference (computing the conditional probability of a query variable given certain evidence) can be performed by summing out all non-query non-evidence variables from the factorization. This is essentially what the variable elimination (VE) algorithm (Zhang and Poole, 1996) does.

A field where BNs are applied is *statistical relational learning* (SRL) (Getoor and Taskar, 2007). This is a research area that combines the elegant handling of uncertainty from probability theory with the capability of representing complex relational domains of first-order logic. A large number of the formalisms used in SRL can be converted into BNs (e.g., CP-logic, ProbLog, ICL, PRISM, BLP). For these formalisms, only exploiting the notion of independence does not yield the most efficient representation possible. The CPDs resulting from the conversion exhibit a particular internal structure, which we will call *local structure*. For example, a CPD may be given as a decision tree (Ramon et al., 2008), may express that the different conditions influence the variable independently (noisy-or or noisy-max), or may impose constraints on the range of a variable in a certain context (Meert et al., 2008). In these cases, a table based representation contains redundancies; an alternative representation that avoids these redundancies may yield more efficient inference.

Several methods have been proposed to exploit local structure. For instance, contextual variable elimination (CVE) (Poole and Zhang, 2003) uses a more compact representation for decision trees and more complex local structures that exhibit contextual independence. This reduces the tree-width of the network, thus al-

lowing for more efficient inference. CVE is a generalization of probability trees and a comparison is made in (Poole and Zhang, 2003). Another example is multiplicative factorization (MF) (Díez and Galán, 2003) that can exploit local structures like independent causation (e.g. noisy-or/and). CVE and MF each exploit one particular type of structure, but cannot handle the other. CVE relies on contexts being mutually-exclusive and exhaustive (we will call this the MEE-restriction), which makes it unsuitable to combine it with MF.

There are also methods that utilize a preprocessing phase to compile the belief network into a different structure, which is optimized for answering multiple queries and allows efficient inference with particular types of local structure. Some known methods are the Arithmetic Circuits (AC) of (Chavira and Darwiche, 2007) and the AND/OR-trees of (Mateescu and Dechter, 2008). In SRL, for each query a new network is built (every time requiring a compilation), therefore, in this paper, we focus on the compilation-free methods.

The main contribution of this paper is that we show how in CVE the MEE-restriction can be lifted. This leads to a new method, *CVE-OC*: CVE with overlapping constraints. Lifting the MEE-restriction has two important consequences: (a) contexts can be encoded more compactly, with increased efficiency as a result, and (b) factorizations like MF can also be handled, which means that CVE-OC can exploit all structures that CVE and MF can exploit. An additional contribution, is that CVE-OC handles constraints on the range of multi-valued variables. This is an extension of CVE for which no concrete solution has been presented up till now.

This paper is organized as follows. We start by providing background on CVE and MF. After this we presents CVE-OC, the paper's main contribution. Next, we discuss its usefulness in the context of statistical relational learning. We present an experimental evaluation in that context, and finally present our conclusions.

2 Preliminaries

2.1 Contextual variable elimination (CVE)

CVE makes use of a more specific form of conditional independence known as *contextual independence* (Boutilier et al., 1996; Poole and Zhang, 2003).

Definition 1 (Contextual Independence)**.** Assume that $\mathbf{X}$ $\mathbf{Y}$, $\mathbf{Z}$ and $\mathbf{C}$ are sets of variables. $\mathbf{X}$ and $\mathbf{Y}$ are contextually independent given $\mathbf{Z}$ and context $\mathbf{C} = \mathbf{c}$, with $\mathbf{c} \in \mathrm{dom}(\mathbf{C})$, iff

$$Pr(\mathbf{X}|\mathbf{Y} = \mathbf{y} \wedge \mathbf{Z} = \mathbf{z} \wedge \mathbf{C} = \mathbf{c}) = Pr(\mathbf{X}|\mathbf{Z} = \mathbf{z} \wedge \mathbf{C} = \mathbf{c})$$

for all $\mathbf{y} \in \mathrm{dom}(\mathbf{Y})$ and $\mathbf{z} \in \mathrm{dom}(\mathbf{Z})$ such that $Pr(\mathbf{Y} = \mathbf{y} \wedge \mathbf{Z} = \mathbf{z} \wedge \mathbf{C} = \mathbf{c}) > 0$. We also say that $\mathbf{X}$ is *contextually independent* of $\mathbf{Y}$ given $\mathbf{Z}$ and context $\mathbf{C} = \mathbf{c}$ (if we drop $\mathbf{C} = \mathbf{c}$, we say $\mathbf{X}$ is conditionally independent from $\mathbf{Y}$ given $\mathbf{Z}$).

VE (Zhang and Poole, 1996) represents the joint distribution as a product of factors, in which each factor is a conditional probability table. CVE (Poole and Zhang, 2003) factorizes the joint distribution further by replacing each factor by a set of *contextual factors* or *confactors*. A confactor r_i consists of two parts: a *context* and a *table*:

$$\langle \underbrace{V_1 = v_{1,i} \wedge \ldots \wedge V_{k-1} = v_{k-1,i}}_{\text{context}} , \underbrace{factor_i(V_k, \ldots, V_m)}_{\text{table}} \rangle$$

The context is a conjunction of variable-value tests ($V_j = v_{j,i}$), which indicates the condition under which the table is applicable (if the context is "*true*" the table is always applicable). The context is used to split up factors into confactors based on Def. 1. The table stores probabilities for all value assignments of a set of zero or more variables $(V_k, \ldots, V_m)$.

The set of confactors that together represent the CPD of a variable V (the confactors *for* V) is mutually-exclusive and exhaustive (MEE). This means that for each possible value assignment for a variable V and its parents $\mathbf{pa}(V)$,

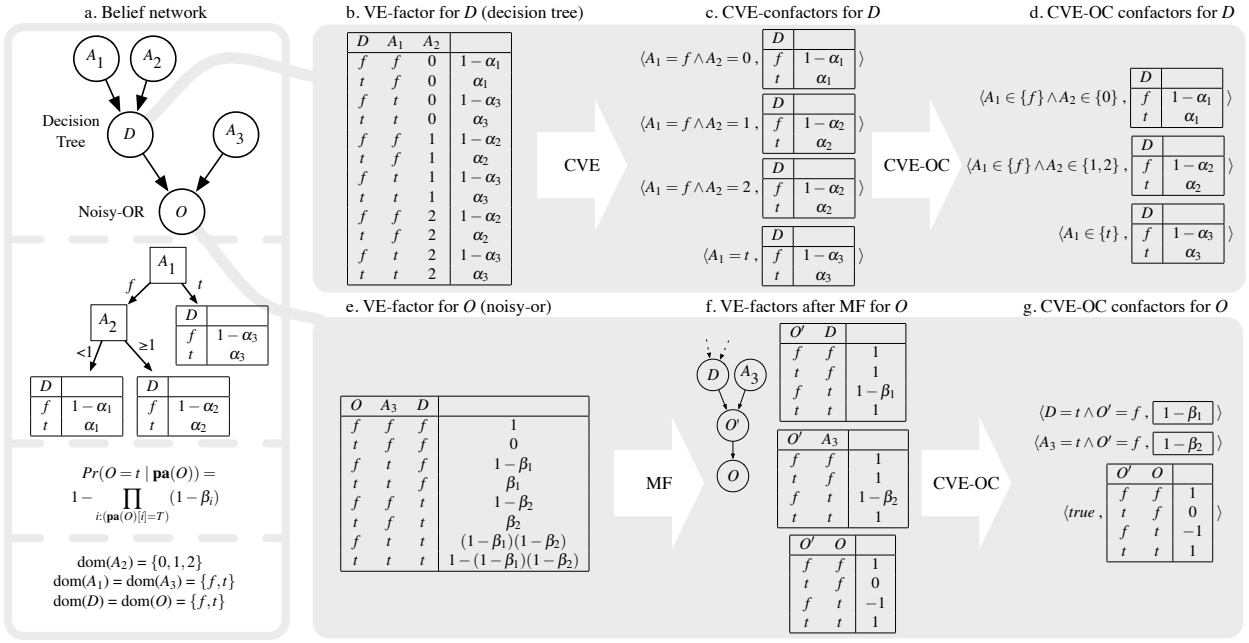

Figure 1: (a) Belief network with local structure; on top the graphical model, then the decision tree for node D and the definition of noisy-or for node O, and at the bottom the domains for the variables; (b) CPD for D represented as a table like in VE; (c) CPD for D represented by confactors for use in CVE; (d) CPD for D represented by confactors for CVE-OC; (e) CPD for O; (f) CPD for O factorized with MF for use in VE; (g) CPD for noisy-or represented by confactors for CVE-OC after MF.

there is precisely one confactor of which the table includes the parameter $Pr(V = v \mid \mathbf{pa}(V) = \mathbf{v_{pa}})$. These conditions ensure that confactors for the same variable do not overlap, therefore, the set of all confactors for a variable is identical to the original factor.

Fig. 1.c shows a confactor representation of a BN for which the CPD for D can be represented by a decision tree. This CPD can be compactly represented as a set of confactors of which each context is the conjunction of variable-value tests on a path from the decision tree root to one of its leaves. (As a note, the converse is not true; a decision tree cannot always represent confactors equally compactly.)

We describe CVE at a high level (the complete algorithm can be found in (Poole and Zhang, 2003)). Similar to VE, CVE eliminates the non-query, non-evidence variables one by one from the joint distribution. To eliminate a variable E, it relies on four basic operations:

1. $\langle c, t_1\rangle \otimes \langle c, t_2\rangle \equiv \langle c, t_1 \otimes t_2\rangle$, multiplying two confactors with identical contexts c.
2. $\sum_E \langle c, t\rangle \equiv \langle c, \sum_E t\rangle$, summing out a variable E that appears in the table of a confactor.
3. $\sum_E (\langle c \wedge E = e_1, t_1\rangle, \ldots, \langle c \wedge E = e_k, t_k\rangle) \equiv \langle c, \sum t_i\rangle$, summing out a variable E, with domain $e_1, \ldots, e_k$, that appears in the contexts.
4. $\langle c, t\rangle \equiv \langle c \wedge X = x_1, t(X = x_1)\rangle, \ldots, \langle c \wedge X = x_k, t(X = x_k)\rangle$, *splitting* a factor; $t(X = x_i)$ is table t but elements for which $X \neq x_i$ are removed.

The first three operations are only possible if the contexts are identical (indicated with c) except for the variable to eliminate (E). To make the contexts identical, CVE uses the fourth operator (splitting). Given two confactors, repeated splitting can be used to create two confactors with identical contexts. The order in

which these operators are applied is chosen by the so-called *absorption* algorithm for CVE. Splitting creates extra confactors, and therefore heuristics are used to avoid this operation as much as possible.

Confactors can represent CPDs more compactly than tables, but, as the previous discussion illustrates, at the cost of more complicated basic operations.

2.2 Multiplicative factorization of noisy-max

Fig. 1.e shows how noisy-or can be represented in terms of a table that is used by VE. This representation has the disadvantage that, while the inputs independently cause the output, this independence is not reflected in the factorization. Confactors do not offer a solution for this type of local structure, so another technique should be used.

(Díez and Galán, 2003) propose a state-of-the-art multiplicative[1] factorization (MF) for a factor representing noisy-or (and its generalization noisy-max). In this new set of factors, each factor only involves one of the inputs. In general, this leads to faster inference with VE. It is not necessary for this paper to fully understand the method or the example in Fig. 1.f, but important to note is that this method uses multiple factors to represent the CPD for a variable (in this case O'). Because of the MEE-restriction, these two factors cannot be represented by confactors.

3 CVE with overlapping contexts

Our main contribution is the CVE-OC algorithm. The CVE-OC algorithm removes the restrictions on the confactors imposed by the CVE algorithm:

First, CVE expects that the confactors for a variable V are MEE. This condition ensures that the parameters in the confactors are identical to those in V's original conditional probability table. It is also a pre-condition for the algorithm CVE uses to combine confactors while eliminating a variable (the absorption algorithm). As mentioned in the introduction, this pre-condition has certain disadvantages (e.g., it is incompatible with MF (Díez and Galán, 2003), and it may make expressing logical constraints more complicated). Therefore, we choose to remove this pre-condition. As a result, a parameter in the original table is not guaranteed to be equal to a parameter in a single confactor (like in CVE) but is equal to the multiplication of different parameters found in different confactors with non-mutually-exclusive contexts. As a consequence the absorption algorithm can no longer be used and we need a new technique to decide which confactors to combine when.

Second, the equality tests in the contexts can be replaced with set membership tests. This allows for a more compact representation in domains with multi-valued variables. This representation was already proposed by Poole and Zhang (Poole and Zhang, 2003), but not supported in their algorithm and implementation as it requires one to extend the splitting operation.

A confactor r_i now has the following form:

$$\langle V_1 \in v_{1,i} \wedge \ldots \wedge V_k \in v_{k,i} \wedge \ldots \wedge V_n \in v_{n,i} \quad , \\ factor_i(V_k, \ldots, V_n, \ldots, V_m) \rangle$$

The context is a conjunction of set membership tests ($V_j \in v_{j,i}$, $v_{j,i} \subseteq dom(V_j)$, with $1 \leq j \leq n$), which indicates the condition under which the table is applicable. The table stores probabilities for given value assignments for a set of zero or more variables ($V_k \ldots V_m$). Note that a variable can now appear both in the context and in the table.

The interpretation of a set of confactors with overlapping contexts for a variable V can be given in terms of the multiplication of their parameters. Given a value assignment for a variable and its parents, it is now possible that multiple confactors for V have contexts that are a applicable and each of these confactors has a parameter in its table that is consistent with the value assignment. The product of these parameters is equal to the parameter in the original

[1] The term 'multiplicative' is used because the summation that is typical for noisy-or is transformed into a multiplication causing further factorization.

table representing the CPD. If the set is not exhaustive, we assume that the value assignments not covered by a context correspond to parameters that are equal to 1.0, which are irrelevant in a multiplication.

Based on the above modifications, we can convert the CPDs in Fig. 1.c and 1.f into the more compact representation in Fig. 1.d and 1.g, which is the input to CVE-OC. This shows three new uses of confactors: (a) it is possible to use set membership to express value ranges of a variable (e.g., the conditions on A_2); (b) noisy-or can be represented more efficiently by using MF (Díez and Galán, 2003) (possibly combined with set membership in case of multi-valued variables); (c) logical constraints can be more compactly expressed and will be exploited during variable elimination.

3.1 The CVE-OC algorithm

Recall from the explanation of CVE that its core operations are not only multiplication and sum-out, but also *compatibility checking* and *splitting*. As explained in (Poole and Zhang, 2003), compatibility checking and splitting must be performed very often and may therefore be computationally expensive.

Since CVE-OC allows overlapping contexts, it cannot use *absorption* to reduce the number of compatibility checks. Moreover, the use of set membership tests requires even more types of splitting. To improve the efficiency of compatibility checking and splitting we propose a temporary tree-based index structure to represent the set R_E of all confactors that contain the variable E that is being eliminated.

Fig. 2 shows an example of this index structure. Each internal node contains a variable that appears in a context of a confactor in R_E and the outgoing edges of the node are labeled with subsets of the variable's domain. The leaf nodes contain the tables.

Before explaining the index construction procedure, we define the *rank* of a variable. Each variable is given a different rank. $rank(E) = 0$, and the ranks of the other variables are positive and increase monotonically with the number of confactors they appear in (ties are broken

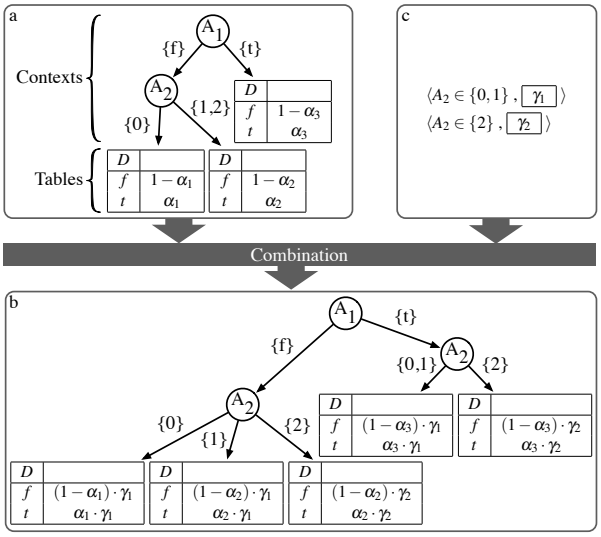

Figure 2: (a) Tree index structure used by CVE-OC to eliminate A_2, after adding the confactors from Fig. 1.d. (b) The tree structure after adding the confactors shown in (c) to (a).

at random). The index will have the property that variables with a higher rank occur higher up in the tree.

To construct the index, CVE-OC starts with a tree that consists of a single leaf that contains a table $t = 1.0$. Then it absorbs all confactors from R_E one by one into the tree. In the end, the tree will contain for any context a correct CPD. To absorb a confactor $r = \langle c, t \rangle$, it moves the confactor down the tree. When it encounters an internal node containing variable N, one of the following five cases may occur (CVE-OC acts according to the first case that applies).

1. c's top-ranked variable V has a higher rank than rank(N), i.e., V should appear above N in the tree. V occurs as $V \in v$ in c. CVE-OC then replaces node N by a new node labeled V with two outgoing edges: one labeled v, and one labeled with its complement $(\mathrm{dom}(V) - v)$. The original subtree rooted at N is duplicated and becomes both the left and right subtree of the new node V. CVE-OC removes variable V from r's context c and sorts the resulting confactor down both subtrees.

2. N appears in r's context, i.e., $(N \in n) \in c$. If one of the outgoing edges from N is labeled n, then CVE-OC sorts r down that branch. If not, then it must perform the splitting operation. To this end, it processes all N's outgoing edges in turn. For a given edge i, if its label n_i is disjoint from n, the corresponding subtree is incompatible with r and no further computation is required. Otherwise, n_i must be split. CVE-OC removes the edge labeled n_i from N and replaces it with two new edges, one labeled $n_i^+ = n_i \cap n$ and one labeled $n_i^- = n_i - n$. The original subtree rooted at n_i is duplicated below these two new edges. Next, CVE-OC removes N from r's context and projects its table on the condition $N \in n_i^+$. Then it sorts the resulting confactor down edge n_i^+.

3. N appears in r's table. Let $n_1, \ldots, n_k$ be the labels of N's outgoing edges. CVE-OC sorts r down the tree via each n_i after projecting r's table on the condition $N \in n_i$.

4. r represents a logical constraint ($t = 0$) and $c = \emptyset$. In this case, CVE-OC replaces node N by a new leaf and initializes the table in that leaf to zero. We call this step pruning the tree based on a constraint.

5. CVE-OC sorts r down to all subtrees of N.

If r reaches a leaf, which stores a table t_l, then the following happens. If $t_l = 0$, then this leaf represents a constraint and no further computation is required. If $t_l \neq 0$, then there are two cases: either $c = \emptyset$ or $c \neq \emptyset$. In the former case, CVE-OC replaces the table in the leaf by the product of t_l and t. In the latter case, it must introduce a new node for the top ranked variable in c into the tree. This is done in precisely the same way as in step one above.

After all confactors in R_E are absorbed into the index, CVE-OC can sum-out the variable E. This is done in two steps. In the first step, E is summed out from all the tables in the leaves of the index. The next step applies to all internal nodes that are labeled with E. Because $\text{rank}(E) = 0$, all children of such nodes are leaves. To sum-out E, a node that contains E is simply replaced by a leaf with a table equal to the sum of the tables in E's children.

After E is eliminated, the tree is converted back into a set of confactors, by creating one confactor for every leaf. Together with the confactors not in R_E, these form the set of confactors for the following elimination step. Note that we cannot reuse the same index for eliminating a different variable because the index is specific to the variable that is being eliminated. Also, converting all confactors into one single tree is to be avoided as a tree is in general not the most compact representation for a set of confactors. Therefore, we include as few as possible confactors in the tree (by means of the variable ordering), and we only use the tree to make compatibility checking and splitting more straightforward.

4 Experiments

We evaluate the inference methods on the task of inferring the marginal distribution of one designated variable in four types of BNs of varying complexity. We always select the variable with the highest inference cost and do not include any evidence (i.e., we consider the most difficult case). The software and BNs are available online (dtai.cs.kuleuven.be/corporal). We compare five algorithm/input combinations: VE, VE with multiplicative factorization (VE+MF), CVE$^{\in}$, CVE-OC, and CVE-OC with multiplicative factorization (CVE-OC+MF). CVE$^{\in}$ is our own C++ implementation of CVE; it is the second algorithm described in (Poole and Zhang, 2003) (which uses absorption), extended with set membership tests (so we can accurately assess the contribution of the overlapping confactors). CVE-OC uses exactly the same input (confactors) as CVE$^{\in}$; CVE-OC+MF has additional confactors for noisy-or/and structures. We use the minimum deficiency elimination ordering (Bertele and Brioschi, 1972), which is a simple greedy heuristic that performs well for VE. Fig. 3 presents the results. Additionally, we have added in the graphs results obtained using the ACE system (Chavira and Darwiche,

2007), which is a representative and state-of-the-art compilation-based approach. Version 2.0 is used, with default settings. The input consists of conditional probability tables except for noisy-or/and nodes which are encoded using the noisy-max syntax.

The BNs in (a) and (b) are constructed from artificial CP-logic theories (Vennekens et al., 2009). The BNs in (a) only include interconnected noisy-or (white) and and (black) nodes with a linearly increasing number of parents. VE runs out of memory when the the number of noisy-or nodes is larger than 8, while CVE can handle larger BNs because of its compact confactor representation. VE+MF and CVE-OC+MF are much faster than the other methods because they efficiently factorize the noisy-or nodes. For experimental comparison of MF with other methods for noisy-or networks we refer to (Díez and Galán, 2003; Savicky and Vomlel, 2007). The ACE system also exploits the presence of noisy-or/and nodes but is a factor 100 slower because of the compilation where it optimizes for all variables. After crossing the curves for non-MF methods, the ACE-curve stops because the tables used as input become too large to generate. The network in (b) contains many CPDs structured as decision trees (black) with a linearly increasing number of parents. CVE and CVE-OC are well suited to handle such a representation and are therefore faster than VE (MF has no influence). The ACE system seems to be unable to fully exploit the interconnected decision tree structures in the CPDs and is also slower because it tries to optimize the structure for all variables.

The BNs in (c) are constructed from a CP-theory that was learned from the UW-CSE dataset (Richardson and Domingos, 2006). The UW-CSE BNs include all structures also included in (a) and (b). For such networks CVE-OC+MF excels as it can efficiently represent all these structures. The other methods run out of memory because they cannot represent one of the local structures efficiently. The noisy-or/and nodes are no problem for the ACE system and when the decision tree structures do not interact too heavily it can handle them efficiently. The curve for CVE-OC+MF fluctuates because the heuristic used for the elimination ordering is too agnostic about local structure when combining different structures. A better heuristic is an interesting future research topic.

The networks in (d) are the randomized networks used in Fig. 11 of (Poole and Zhang, 2003) and created with the original Java code available from the author. This experiment compares CVE$^{\in}$ with the new CVE-OC algorithm on the same data as Poole and Zhang used. This shows that CVE-OC, although more generally applicable, is about as fast as CVE$^{\in}$ even when there are no additional structures to be exploited.

5 Conclusions and future work

We presented the algorithm CVE-OC (CVE with overlapping contexts), which extends contextual variable elimination (CVE). The introduction of overlapping contexts is a simple but powerful step. From the representation point of view, it offers an elegant combination of deterministic and probabilistic knowledge. From the computational point of view, the need for equality testing is reduced, invalid combinations of values are pruned, and the loosened restrictions allow for better optimizations. CVE-OC generalizes over both CVE and MF, and provide optimization opportunities beyond the union of what those methods offer.

The experiments show that CVE-OC, while more generally applicable, can handle input for CVE without any loss in efficiency. Because of its generality, the input for CVE-OC can be more compact than for CVE and other known optimization methods for VE can be integrated. For example, the integration of MF is shown to be faster and more compact than only VE, CVE or MF.

In future work, we intend to perform more experiments on the propagation of constraints in CVE-OC and would like to investigate more the influence of heuristics for elimination orderings. With respect to SRL, we would like to investigate further the relationship between CVE-OC and other SRL inference algorithms and see how

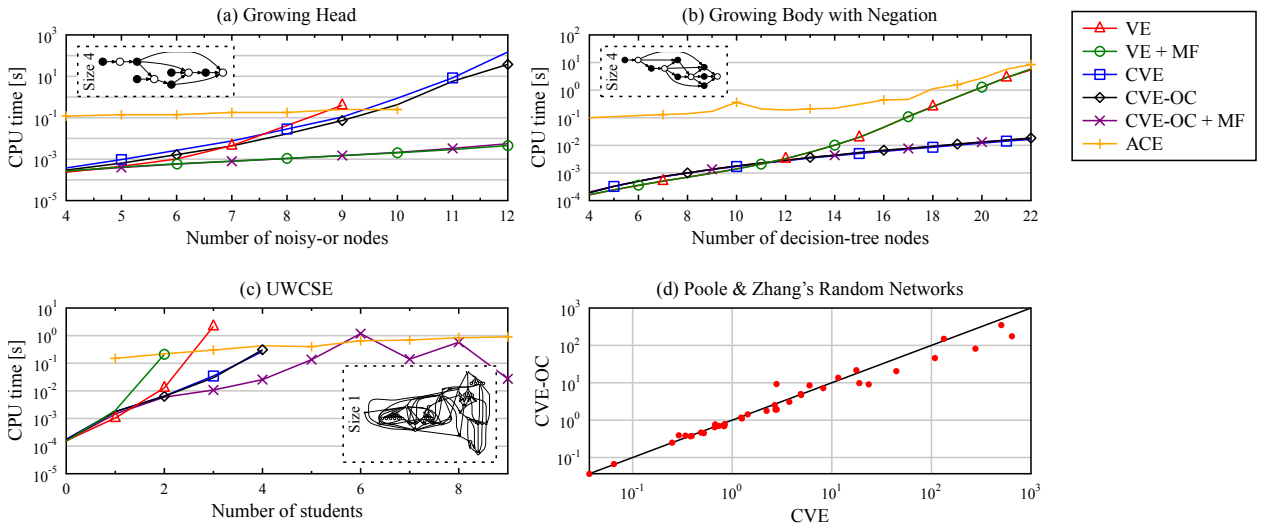

Figure 3: Inference times for networks originating from CP-theories and random networks. The horizontal axis of (a)-(c) indicates BN complexity.

we can integrate this into learning methods for SRL.

Acknowledgements

Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) to Wannes Meert. GOA/08/008 'Probabilistic Logic Learning'.

References

U. Bertele and F. Brioschi. 1972. *Nonserial Dynamic Programming*. Academic Press.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. 1996. Context-specific independence in Bayesian networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 115–123.

M. Chavira and A. Darwiche. 2007. Compiling Bayesian networks using variable elimination. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2443–2449.

F.J. Díez and S.F. Galán. 2003. Efficient computation for the noisy MAX. *Intelligent Systems*, 18(2):165–177.

L. Getoor and B. Taskar, editors. 2007. *Statistical Relational Learning*. MIT Press.

R. Mateescu and R. Dechter. 2008. Mixed deterministic and probabilistic networks. ICS technical report, University of California, Irvine, May.

W. Meert, J. Struyf, and H. Blockeel. 2008. Learning ground CP-logic theories by leveraging Bayesian network learning techniques. *Fundamenta Informaticae*, 89(1):131–160.

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

D. Poole and N. Zhang. 2003. Exploiting contextual independence in probabilistic inference. *Artificial Intelligence Research*, 18:263–313.

J. Ramon, T. Croonenborghs, D. Fierens, H. Blockeel, and M. Bruynooghe. 2008. Generalized ordering-search for learning directed probabilistic logical models. *Machine Learning*, 70(2-3):169–188.

M. Richardson and P. Domingos. 2006. UW-CSE. http://alchemy.cs.washington.edu/data/uw-cse/.

P. Savicky and J. Vomlel. 2007. Exploiting tensor rank-one decomposition in probabilistic inference. *Kybernetika*, 43(5):747–764.

J. Vennekens, M. Denecker, and M. Bruynooghe. 2009. CP-logic: A language of causal probabilistic events and its relation to logic programming. *Theory and Practice of Logic Programming*, 9(3):245–308.

N. Zhang and D. Poole. 1996. Exploiting causal independence in bayesian network inference. *Artificial Intelligence Research*, 5:301–328.