Uncertain Evidence for Probabilistic Relational Models*

Marcel Gehrke[0000−0001−9056−7673], Tanya Braun, and Ralf Möller

Institute of Information Systems, University of Lübeck, Lübeck, Germany
{gehrke, braun, moeller}@ifis.uni-luebeck.de

Abstract. Standard approaches for inference in probabilistic relational models include lifted variable elimination (LVE) for single queries. To efficiently handle multiple queries, the lifted junction tree algorithm (LJT) uses a first-order cluster representation of a model, employing LVE as a subroutine in its steps. LVE and LJT can only handle certain evidence. However, most events are not certain. The purpose of this paper is twofold, (i) to adapt LVE, presenting LVE^{evi}, to handle uncertain evidence and (ii) to incorporate uncertain evidence for multiple queries in LJT, presenting LJT^{evi}. With LVE^{evi} and LJT^{evi}, we can handle uncertain evidence for probabilistic relational models, while benefiting from the lifting idea. Further, we show that uncertain evidence does not have a detrimental effect on completeness results and leads to similar runtimes as certain evidence.

1 Introduction

Areas such as health care or logistics involve probabilistic data with relational aspects where many objects are in relation to each other with uncertainties about object existence, attribute value assignments, or relations between objects. E.g., health care systems involve electronic health records (EHRs) (the relational part) for many patients (the objects) and uncertainties [22] due to, e.g., missing information. Additionally, evidence or events are not always certain due to different sensors or where the tests are performed. Automatically analysing EHRs can improve the care of patients and save time for medical professionals to spend on other important tasks.

Answering queries in probabilistic, relational environments, like predicting outcomes of treatments, needs efficient exact inference algorithms, which is particularly true for health care as approximations might not be good enough [24]. Probabilistic databases (PDBs) can answer queries for relational models with uncertainties [8,19]. However, each query can contain redundant information, resulting in huge queries. In contrast to PDBs, we build more expressive and compact models (offline) enabling efficient answering of more compact queries (online). Currently, these compact models cannot handle uncertain evidence. Therefore, in this paper, we study the problem of exact inference in relational temporal probabilistic models with uncertain evidence.

Research in the field of lifted inference has lead to efficient algorithms for relational models. lifted variable elimination (LVE), first introduced in [16] and expanded in [17,12,21], saves computations by reusing intermediate results for isomorphic subproblems when answering a query. The lifted junction tree algorithm (LJT) sets up a

* This research originated from the Big Data project being part of Joint Lab 1, funded by Cisco Systems Germany, at the centre COPICOH, University of Lübeck

first-order junction tree (FO jtree) to handle multiple queries efficiently [2] using LVE as a subroutine. Van den Broeck et al. apply lifting to weighted model counting and knowledge compilation [6]. To scale lifting, Das et al. use graph databases storing compiled models to count faster [9]. Lifted belief propagation (BP) provides approximate solutions to queries, often using lifted representations, e.g. [1]. But, to the best of our knowledge, none of the approaches handle uncertain evidence.

We focus on *exact* inference for multiple queries and present an efficient algorithm for based on LJT, called LJT^{evi}, handling uncertain evidence. For Bayesian networks, work on uncertain evidence exists, sometimes called soft evidence in contrast to certain, i.e., hard evidence [7,14,13,10,15]. In this paper, we interpret uncertain evidence in the sense of a priori distributions, which is closely related to Pearl's method of virtual evidence [13]. This paper includes two main contributions, (i) an algorithm, LVE^{evi}, handling uncertain evidence for probabilistic relational models and (ii) LJT^{evi}, handling uncertain evidence for multiple queries. Additionally, we show soundness and completeness results for LVE^{evi} and LJT^{evi} and a brief empirical case study.

The remainder of this paper is structured as follows: First, we introduce basic notations and recap LVE and LJT. Then, we show how to handle uncertain evidence and present LJT^{evi}, followed by a discussion. We conclude with upcoming work.

2 Preliminaries

This section specifies notations and recaps LJT. Based on [17], a running example models the interplay of natural or man-made disasters, an epidemic, and people being sick, travelling, and being treated. Parameters represent disasters, people, and treatments.

2.1 Parameterised Probabilistic Models

Parameterised models compactly represent models using logical variables (logvars) to parameterise random variables (randvars), so called parameterised randvars (PRVs).

Definition 1. *Let* $\mathbf{L}$*,* Φ*, and* $\mathbf{R}$ *be sets of logvar, factor, and randvar names respectively. A* PRV $R(L_1, \ldots, L_n)$*,* $n \geq 0$*, is a syntactical construct with* $R \in \mathbf{R}$ *and* $L_1, \ldots, L_n \in \mathbf{L}$ *to represent a set of randvars. For PRV* A*, the term* $range(A)$ *denotes possible values. A logvar* L *has a domain* $\mathcal{D}(L)$*. A* constraint $(\mathbf{X}, C_{\mathbf{X}})$ *is a tuple with a sequence of logvars* $\mathbf{X} = (X_1, \ldots, X_n)$ *and a set* $C_{\mathbf{X}} \subseteq \times_{i=1}^{n} \mathcal{D}(X_i)$ *restricting logvars to values. The symbol* $\top$ *marks that no restrictions apply and may be omitted. For some construct* P*, the term* $lv(P)$ *refers to its logvars, the term* $rv(P)$ *to its PRVs with constraints, and the term* $gr(P)$ *to all instances of* P*, i.e.* P *grounded w.r.t. constraints.*

For the example, we build the boolean PRVs $Epid$, $Sick(X)$, and $Travel(X)$ from $\mathbf{R} = \{Epid, Sick, Travel\}$ and $\mathbf{L} = \{X\}$, $\mathcal{D}(X) = \{alice, eve, bob\}$. $Epid$ holds if an epidemic occurs. $Sick(X)$ holds if a person X is sick, $Travel(X)$ holds if X travels. With a constraint $C = (X, \{eve, bob\})$, $gr(Sick(X)_{|C}) = \{Sick(eve), Sick(bob)\}$. $gr(Sick(X)_{|\top})$ also contains $Sick(alice)$. Parametric factors (parfactors) link PRVs. A parfactor describes a function, identical for all argument groundings, mapping argument values to real values (potentials), of which at least one is non-zero.

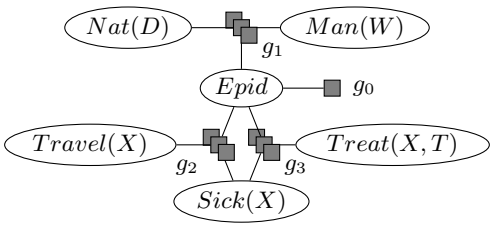

Fig. 1: Parfactor graph for G_{ex}

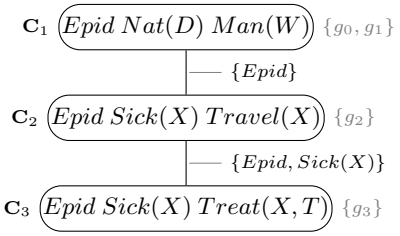

Fig. 2: FO jtree for G_{ex}

Definition 2. *Let* $\mathbf{X}$ *be a set of logvars,* $\mathcal{A} = (A_1, \ldots, A_n)$ *a sequence of PRVs with* $lv(\mathcal{A}) \subseteq \mathbf{X}$*,* C *a constraint on* $\mathbf{X}$*, and* $\phi : \times_{i=1}^{n} range(A_i) \mapsto \mathbb{R}^+$ *a function with name* $\phi \in \Phi$*, identical for* $gr(\mathcal{A}_{|C})$*. We denote a* parfactor *g by* $\forall \mathbf{X} : \phi(\mathcal{A})_{|C}$*. We omit* $(\forall \mathbf{X} :)$ *if* $\mathbf{X} = lv(\mathcal{A})$ *and* $|C$ *if* $C = \top$*. A set of parfactors forms a* model $G := \{g_i\}_{i=1}^{n}$.

We define a model G_{ex} as our running example. Let $\mathbf{L} = \{D, W, M, X\}$, $\Phi = \{\phi_0, \phi_1, \phi_2, \phi_3\}$, and $\mathbf{R} = \{Epid, Nat, Man, Sick, Travel, Treat\}$. We build three more boolean PRVs. $Nat(D)$ holds if a natural disaster D occurs, $Man(W)$ if a man-made disaster W occurs. $Treat(X,T)$ holds if a person X is treated with treatment T. The other domains are $\mathcal{D}(D) = \{earthquake, flood\}$, $\mathcal{D}(W) = \{virus, war\}$, and $\mathcal{D}(T) = \{vaccine, tablet\}$. The model reads $G_{ex} = \{g_i\}_{i=0}^{3}$, $g_0 = \phi_0(Epid)$, $g_1 = \phi_1(Epid, Nat(D), Man(W))_{|\top}$, $g_2 = \phi_2(Epid, Sick(X), Travel(X))_{|\top}$, and $g_3 = \phi_3(Epid, Sick(X), Treat(X,T))_{|\top}$. Parfactors g_1 to g_3 have eight input-output pairs, g_0 has two (omitted here). Figure 1 depicts G_{ex} as a graph with six variable nodes for the PRVs and four factor nodes for the parfactors with edges to arguments.

Evidence displays symmetries if observing the same value for n instances of a PRV [21]. In a parfactor $g_E = \phi_E(R(\mathbf{X}))_{|C_E}$, a potential function ϕ_E and constraint C_E encode the observed values and instances for PRV $R(\mathbf{X})$. Assume we observe the value $true$ for ten randvars of the PRV $Sick(X)$. The corresponding parfactor is $\phi_E(Sick(X))_{|C_E}$. C_E represents the domain of X restricted to the 10 instances and $\phi_E(true) = 1$ and $\phi_E(false) = 0$. A technical remark: To *absorb* evidence, we split all parfactors g_i that cover $R(X)$, called shattering [17], restricting C_i to those tuples that contain $gr(R(X)_{|C_E})$ and a duplicate of g_i to the rest. g_i absorbs g_E (cf. [21]).

The *semantics* of a model G is given by grounding and building a full joint distribution P_G. With Z as the normalisation constant, G represents $P_G = \frac{1}{Z} \prod_{f \in gr(G)} f$. The query answering (QA) problem asks for a marginal distribution of a set of randvars or a conditional distribution given events, which boils down to computing marginals w.r.t. a model's joint distribution, eliminating non-query terms. Formally, $P(\mathbf{Q}|\mathbf{E})$ denotes a query with $\mathbf{Q}$ a set of grounded PRVs and $\mathbf{E} = \{E_i = e_i\}_{i=1}^{n}$ a set of events. An example query for G_{ex} is $P(Epid|Sick(eve) = true)$. Next, we look at LJT, a lifted QA algorithm, which seeks to avoid grounding and building a full joint distribution.

2.2 Query Answering Algorithms

LVE and LJT answer queries for probability distributions. LJT uses an FO jtree with LVE as a subroutine. We briefly recap LVE and LJT.

Algorithm 1 Outline of LVE

```
1: LVE(Model G, Query Q, Evidence E)
2:   Absorb E in G
3:   while G has non-query PRVs do
4:     if PRV A fulfils sum-out prec. then
5:       Eliminate A using sum-out
6:     else
7:       Apply transformator
8:   Multiply and normalise G
```

Algorithm 2 Outline of LJT

```
1: LJT(M. G, Queries {Q_i}_{i=1}^m, Evidence E)
2:   Construct FO jtree J
3:   Enter E into J
4:   Pass messages on J
5:   for each query Q_i do
6:     Find subtree J_i for Q_i
7:     Extract submodel G_i from J_i
8:     LVE(G_i, Q_i, ∅)
```

Lifted Variable Elimination: In essence, LVE computes variable elimination for one case and exponentiates the result for isomorphic instances (lifted summing out), avoiding duplicate calculations. Taghipour et al. implement LVE through an operator suite (cf. [21] for details). Its main operator *sum-out* realises lifted summing out. An operator *absorb* handles evidence in a lifted way. The remaining operators (*count-convert*, *split*, *expand*, *count-normalise*, *multiply*, *ground-logvar*) aim at enabling a lifted summing out, transforming part of a model. All operators have pre- and post-conditions to ensure computing a result equivalent to one computed on $gr(G)$. Algorithm 1 shows an outline. To answer a query, LVE eliminates all non-query randvars from the model.

Lifted Junction Tree Algorithm: Algorithm 2 outlines LJT for a set of queries $\{\mathbf{Q}_i\}_{i=1}^m$ given a model G and evidence $\mathbf{E}$. First, LJT builds an FO jtree, which clusters a model into submodels that LJT uses to answer queries after preprocessing. We define a minimal FO jtree with parameterised clusters (parclusters) as nodes, which are sets of PRVs connected by parfactors, as follows.

Definition 3. *Let* $\mathbf{X}$ *be a set of logvars,* $\mathbf{A}$ *a set of PRVs with* $lv(\mathbf{A}) \subseteq \mathbf{X}$*, and* C *a constraint on* $\mathbf{X}$*. Then,* $\forall\mathbf{X}{:}\mathbf{A}_{|C}$ *denotes a* parcluster*. We omit* $(\forall\mathbf{X}{:})$ *if* $\mathbf{X} = lv(\mathbf{A})$ *and* $|\top$*. An* FO jtree *for a model* G *is a cycle-free graph* $J = (V, E)$*, where* V *is the set of nodes, i.e., parclusters, and* E *the set of edges.* J *must satisfy three properties: (i)* $\forall\mathbf{C}_i \in V$*:* $\mathbf{C}_i \subseteq rv(G)$*. (ii)* $\forall g \in G$*:* $\exists\mathbf{C}_i \in V$ *s.t.* $rv(g) \subseteq \mathbf{C}_i$*. (iii) If* $\exists A \in rv(G)$ *s.t.* $A \in \mathbf{C}_i \wedge A \in \mathbf{C}_j$*, then* $\forall\mathbf{C}_k$ *on the path between* $\mathbf{C}_i$ *and* $\mathbf{C}_j$*:* $A \in \mathbf{C}_k$ *(running intersection property).* J *is* minimal *if by removing a PRV from any parcluster,* J *ceases to be an FO jtree, i.e., no longer fulfils at least one of the three properties. The parameterised set* $\mathbf{S}_{ij}$*, called* separator *of edge* $\{i, j\} \in E$*, is given by* $\mathbf{C}_i \cap \mathbf{C}_j$*. Each* $\mathbf{C}_i \in V$ *has a* local model G_i *and* $\forall g \in G_i$*:* $rv(g) \subseteq \mathbf{C}_i$*. The* G_i*'s partition* G*.*

In a minimal FO jtree, no parcluster is a subset of another parcluster. Figure 2 shows a minimal FO jtree for G_{ex} with parclusters $\mathbf{C}_1 = \{Epid, Nat(D), Man(W)\}$, $\mathbf{C}_2 = \{Epid, Sick(X), Travel(X)\}$, and $\mathbf{C}_3 = \{Epid, Sick(X), Treat(X, T)\}$. $\mathbf{S}_{12} = \{Epid\}$ and $\mathbf{S}_{23} = \{Epid, Sick(X)\}$ are the separators. Parfactor g_0 appears at $\mathbf{C}_1$ but could be in any local model as $rv(g_0) = \{Epid\} \subset \mathbf{C}_i \; \forall \; i \in \{1, 2, 3\}$.

During construction, LJT assigns the parfactors in G to local models (cf. [2]). LJT enters $\mathbf{E}$ into each parcluster $\mathbf{C}_i$ where $rv(\mathbf{E}) \subseteq \mathbf{C}_i$. Local model G_i at $\mathbf{C}_i$ absorbs $\mathbf{E}$ as described above. Message passing distributes local information within the FO jtree.

Two passes from the periphery to the center and back suffice [11]. If a node has received messages from all neighbours but one, it sends a message to the remaining neighbour (*inward* pass). In the *outward* pass, messages flow in the opposite direction. Formally, a *message* m_{ij} from node i to node j is a set of parfactors, with arguments from $\mathbf{S}_{ij}$. LJT computes m_{ij} by eliminating $\mathbf{C}_i \setminus \mathbf{S}_{ij}$ from G_i and the messages of all other neighbours with LVE. A minimal FO jtree enhances the efficiency of message passing. Otherwise, messages unnecessarily copy information between parclusters. To answer a query $\mathbf{Q}_i$, LJT finds a subtree J' covering $\mathbf{Q}_i$, compiles a submodel G' of local models in J' and messages from outside J', and sums out all non-query terms in G' using LVE.

3 LVE for Uncertain Evidence

Currently, evidence in LVE and therefore LJT is always certain. However, often sensors are not completely reliable or some test may be more precisely performed in a hospital compared to a test in a general practice [18]. Before we incorporate uncertain evidence in LVE, we take a closer look at how LVE handles certain evidence.

3.1 Evidence in LVE

In Alg. 1, handling evidence appears as "absorb evidence". Thus, let us now have a look at lifted absorption, which is outlined in Alg. 3, without including counting PRVs for ease of explanation. The operator uses a count function defined as follows.

Definition 4. *Given a constraint $C = (\mathbf{X}, C_{\mathbf{X}})$, for any $\mathbf{Y} \subseteq \mathbf{X}$ and $\mathbf{Z} \subseteq \mathbf{X} \setminus \mathbf{Y}$, the function* $\text{COUNT}_{\mathbf{Y}|\mathbf{Z}} : C_{\mathbf{X}} \to \mathbb{N}$ *is defined by*

$$\text{COUNT}_{\mathbf{Y}|\mathbf{Z}}(t) = |\pi_{\mathbf{Y}}(C_{\mathbf{X}} \bowtie_{\mathbf{Z}} \pi_{\mathbf{Z}}(t))|.$$

i.e., for a tuple $t \in C_{\mathbf{X}}$, it outputs how many constants for $\mathbf{Y}$ co-occur with the value of $\mathbf{Z}$ in t. We define $\text{COUNT}_{\mathbf{Y}|\mathbf{Z}}(t) = 1$ *for $\mathbf{Y} = \emptyset$. $\mathbf{Y}$ is* count-normalised *w.r.t. $\mathbf{Z}$ in C iff*

$$\exists n \in \mathbb{N} : \forall t \in C_{\mathbf{X}} : \text{COUNT}_{\mathbf{Y}|\mathbf{Z}}(t) = n.$$

If n exists, we call it the conditional count of $\mathbf{Y}$ given $\mathbf{Z}$ in C, denoted by $\text{COUNT}_{\mathbf{Y}|\mathbf{Z}}(C)$.

Before we take a closer look at the operator, we illustrate the count function. Consider the constraint $C = ((X, T), \{(eve, tablet), (alice, vaccine), (alice, tablet), (bob, vaccine), (bob, tablet)\})$. With $\mathbf{X} = \{X, T\}$, $\mathbf{Y} = \{T\}$, and $\mathbf{Z} = \{X\}$, the count function calculates the following for tuple $(eve, tablet)$: First, it projects $(eve, tablet)$ onto $\{X\}$, which leaves (eve). Then, it joins eve with the tuples from C, i.e., $(eve, tablet)$, and projects the tuples onto $\{T\}$, which results in a set with one element, $(tablet)$. Last, it outputs the cardinality of the set, here 1. For $(alice, vaccine)$, the first projection yields $(alice)$, with the join resulting in $(alice, vaccine)$ and $(alice, tablet)$ and the second projection resulting in $(vaccine)$ and $(tablet)$, yielding a cardinality of 2. Thus, there does not exist a unique n for all tuples in C, that is, M is not count-normalised. Now, assume the constraint $C' = ((X, T), \{(alice, vaccine), (alice, tablet), (bob, vaccine), (bob, tablet)\})$. Here, each tuple leads to a count of 2

Algorithm 3 Lifted Absorption [21].

Operator ABSORB

Inputs:

(1) $g = \phi(\mathcal{A})_{|C}$: a parfactor in G
(2) $A_i \in \mathcal{A}$ with $A_i = R(\mathbf{X})$
(3) $g_E = \phi_E(R(\mathbf{X}))_{|C_E}$: an evidence parfactor
Let $\mathbf{X}^{excl} = \mathbf{X} \setminus lv(\mathcal{A} \setminus A_i)$;
$L' = lv(\mathcal{A}) \setminus \mathbf{X}^{excl}$;
$o =$ the observed value for $R(\mathbf{X})$ in g_E

Preconditions:

(1) $gr(A_{i|C_i}) \subseteq gr(A_{i|C_E})$
(2) $\mathbf{X}^{excl}$ is count-normalised w.r.t. L' in C.

Output: $\phi'(\mathcal{A}')_{|C'}$, with

(1) $\mathcal{A}' = \mathcal{A} \setminus A_i$
(2) $C' = \pi_{lv(C) \setminus \mathbf{X}^{excl}}(C)$
(3) $\phi'(\dots, a_{i-1}, a_{i+1}, \dots) = \phi(\dots, a_{i-1}, o, a_{i+1}, \dots)^r$ with $r = \text{COUNT}_{\mathbf{X}^{excl}|L'}(C)$

Postcondition: $G \cup \{g_E\} \equiv G \setminus \{g\} \cup \{g_E, \text{ABSORB}(g, A_i, g_E)\}$

given $\mathbf{X} = \{X, T\}$, $\mathbf{Y} = \{T\}$, and $\mathbf{Z} = \{X\}$ and thus, M is count-normalised w.r.t. X in C'. The conditional count of T given X in C' is 2. In case, $alice$ and bob would additionally receive another treatment, the count would be 3. The count in this case is important as absorbing evidence eliminates as many instances as the count function yields, and thus, LVE needs to exponentiate the result with the count.

ABSORB has as inputs an evidence parfactor g_E with evidence for a PRV A_i and a parfactor g, which contains A_i. As a precondition, A_i covers at most the randvars of g_E in g. Thus, LVE often performs a shattering before absorption to split parfactors into parts with evidence and without evidence. The other precondition relates to logvars being eliminated during absorption. For the output parfactor, the operator deletes A_i from g, reducing the dimensions in g. The operator also projects the constraint C of g onto the remaining logvars. Lastly, it collects all potentials that agree with the evidence, i.e., where $A_i = o$, and exponentiates them accordingly. As the operator performs a dimension reduction by deleting A_i from the argument sequence, rather than keeping the argument and setting all potentials where $A_i \neq o$ to 0, LVE has to apply the absorption operator to each parfactor that contains A_i.

To illustrate the ABSORB operator, assume that eve is sick, i.e. $Sick(eve) = true$. LVE builds an evidence parfactor $g_E = \phi_E(Sick(X))_{|C_E}$, with $C_E = (X, \{eve\})$. As g_2 and g_3 also contain $Sick(X)$, both need to absorb g_E. To absorb g_E in g_2, LVE first splits g_2 into g_2' for eve and g_2'' for all other instances, i.e., $alice$ and bob. With g_E and g_2' as inputs, the first precondition holds as both g_E and g_2' have X restricted to eve. Since $\mathbf{X}^{excl} = X \setminus lv(Epid, Travel(X)) = \emptyset$, i.e., no logvars are eliminated, $\mathbf{X}^{excl}$ is count-normalised and $r = 1$. Hence, the operator can proceed. It removes $Sick(X)$ from g_2'. The constraint remains unchanged. Lastly, all potentials that agree with $Sick(eve) = true$ remain and are exponentiated to the power of 1. Similarly g_E gets absorbed in g_3. One could also perform lifted absorption by multiplying g_E into g, which leads to potentials of 0 whenever $A_i \neq o$. Afterwards, one could drop the

mappings with potentials of 0 and then eliminate A_i from the argument sequence as after dropping the mappings, $A_i = o$ in all remaining mappings, holding no further information. However, absorption as in Alg. 3 only works for certain evidence.

3.2 Uncertain Evidence in LVE^{evi}

The main differences to certain evidence and its handling are in specifying evidence, constructing evidence parfactors, and handling evidence parfactors within LVE. Currently, one event has a potential of 1, while all others have a potential of 0 in evidence. With uncertain evidence, we need to be able to specify potentials different from 0 for possible events of a PRV. However, evidence should not incur a scaling factor. Therefore, individual events of a PRV A have assigned a potential p with $p \in [0, 1]$ and the potentials of all possible events of A add up to 1. We allow for two options to specify potentials for events. The first option is to specify the potential for each possible event of a PRV A_i with the sum of the potentials being 1. LVE^{evi} then constructs an evidence parfactor $g_E = \phi_E(A_i)_{|C_E}$ accordingly. The second option is to specify a subset of the events with the sum of the potentials s being at most 1. LVE^{evi} constructs an evidence parfactor $g_E = \phi_E(A_i)_{|C_E}$, distributing the residual potential $1 - s$ on the remaining range values in a max-entropy style [23]. Constructing evidence parfactors in such a way ensures that all range values have a potential and that the potentials add up to 1.

Assume the potential of eve being sick is 0.9. We may specify the evidence using a complete distribution, $Sick(eve) = ((true, 0.9), (false, 0.1))$. The other option is to only specify $Sick(eve) = (true, 0.9)$, a subset of the distribution. Then, LVE^{evi} would distribute the remaining 0.1 max-entropy alike on the remaining range values, while constructing the evidence parfactor. With $Sick(X)$ being boolean, there is only one other range value, namely $false$, which would be assigned a potential of 0.1. In case of another range value, e.g., $immune$, then both would be assigned a potential of 0.05. Assigning a distribution to evidence still allows for specifying certain evidence. Given $Sick(eve) = ((true, 1))$, all other range values would be assigned the potential 0, which is identical to the evidence so far in LVE.

We now present LVE^{evi} to handle uncertain evidence while answering a query. The workflow of LVE^{evi} is identical to LVE as given in Alg. 1 with line 2 changing. Instead of absorbing all evidence $\mathbf{E}$ in affected parfactors, a case distinction occurs, which is specified in Alg. 4, for each evidence parfactor g_E constructed for $\mathbf{E}$. If g_E contains certain evidence, g_E is absorbed in G as before. If g_E is uncertain evidence, g_E is added to G. During query answering, the uncertain evidence is then properly accounted for since g_E is multiplied into the model at one point and therefore, influences a queried distribution accordingly. Next, we consider soundness and completeness of LVE^{evi}.

Algorithm 4 Evidence Handling in LVE^{evi}

```
1: procedure ADDEVIDENCE(G, g_E)
2:     if g_E is uncertain then
3:         Add g_E to G
4:     else
5:         Absorb g_E in G
```

Theorem 1. *LVE^{evi} is sound, i.e., computes a correct result for a query* $\mathbf{Q}$ *given an input model* G *and evidence* $\mathbf{E}$*.*

Proof sketch. Since both LVE^{evi} and LVE handle certain evidence in the same way and LVE is correct [21], LVE^{evi} is correct w.r.t. certain evidence. We interpret uncertain evidence as an a priori distribution for events. LVE^{evi} simply adds evidence parfactors of uncertain evidence to a model. During query answering, LVE^{evi} then handles these parfactors as part of the model, multiplying evidence parfactors into other parfactors accordingly, thus, accounting for evidence as a form of a priori distribution. □

Theorem 2. *LVE^{evi} is complete for unary evidence, i.e., the time complexity is polynomial in the domain sizes of the model logvars.*

Proof sketch. Given certain, unary evidence, i.e., evidence which can be represented in a parfactor with an evidence PRV using one-logvar, LVE is complete [5,20]. Replacing certain, unary evidence with uncertain, unary evidence with a given distribution leads to the same number of splits during shattering and the number of splits is linear per evidence PRV and model parfactor [21]. Thus, LVE^{evi} still has a time complexity polynomial in the domain sizes of the model logvars given uncertain, unary evidence and the completeness results for unary evidence from LVE also hold for LVE^{evi}. □

4 LJT for Uncertain Evidence

LVE^{evi} handles uncertain evidence efficiently for single queries. To handle multiple queries efficiently, we incorporate uncertain evidence into LJT based on the same principles that have guided the adaptation of LVE to handle uncertain evidence. Before we present LJT^{evi}, we first take a closer look at how LJT handles evidence.

4.1 Evidence in LJT

Evidence handling in LJT generally works by performing the following steps: (i) Construct evidence parfactors. (ii) Enter evidence parfactors into FO jtree. (iii) Shatter local models on entered evidence parfactors. (iv) Absorb evidence parfactor in local models. Basically, LJT handles evidence in each local model as LVE does in its input model. In each parcluster that covers an evidence PRV, LJT tests each parfactor for evidence absorption. If a parfactor in a local model covers the evidence PRV, LJT shatters the parfactor on the evidence and lets the affected parfactor absorb the evidence parfactor. Whenever a separator no longer covers an evidence PRV, LJT can omit checking the subtree behind the neighbour associated with the separator based on the running intersection property.

Again, assume that eve is sick as certain evidence with an evidence parfactor $g_E = \phi_E(Sick(X))_{|C_E}$, with $C_E = (X, \{eve\})$. Then, LJT enters g_E in J_{ex}, which is shown in Fig. 1. The PRV $Sick(X)$ occurs in $\mathbf{C}_2$ and $\mathbf{C}_3$. LJT shatters the local models G_2 and G_3, i.e., g_2 and g_3. LJT splits g_2 into g_2' for eve and g_2'' for all other instances, in this case, $alice$ and bob. Analogously, LJT splits g_3 into g_3' and g_3''. Finally, LJT absorbs g_E in g_2' and g_3'. After the absorption, all local models encode information about certain evidence and the overall model.

Algorithm 5 Evidence Handling in LJT^{evi}

```
1: procedure ENTEREVIDENCE(J, g_E)
2:   if g_E is uncertain then
3:     Add g_E to the local model of one parcluster, which contains the PRV of g_E
4:     Shatter local model (optional)
5:     Multiply g_E into local model (optional)
6:   else
7:     Enter g_E in all parclusters, which contain the PRV of g_E
8:     Shatter local models
9:     Absorb g_E
```

4.2 Uncertain Evidence in LJT

LJT^{evi} is based on LJT and is able to handle uncertain evidence as well. Evidence may be specified in the same manner as for LVE^{evi}, which allows for certain evidence as well as uncertain evidence, partially or fully specified with distributions, whose potentials add up to 1. LJT^{evi} has the same workflow as LJT, outlined in Alg. 2. Only line 3 now references steps that incorporate uncertain evidence. Algorithm 5 describes the steps to enter an evidence parfactor g_E in an FO jtree J. Again, a case distinction occurs. If g_E encodes certain evidence, LJT^{evi} works as LJT, absorbing g_E in the affected parfactors of all parclusters that cover the evidence PRV. If g_E encodes uncertain evidence, LJT^{evi} adds g_E to one local model of a parcluster that covers the evidence PRV. During message passing, the information about the evidence is distributed to all other parclusters, which makes it apparent, why uncertain evidence should only be added to one local model. In case LJT^{evi} would add the uncertain evidence parfactor to all parclusters containing the evidence PRV, then the evidence would be distributed during message passing and accounted for multiple times. One could directly shatter a local model of the chosen parcluster and multiply g_E into it. But, the operations are optional: LJT^{evi} uses LVE for its calculations, which is able to handle g_E accordingly and multiply g_E into other parfactors when necessary, resulting in more efficient multiplications.

Let us consider uncertain evidence and LJT^{evi}. Assume that eve is sick with a potential of 0.9. So, LJT^{evi} builds an evidence parfactor $g_E = \phi_E(Sick(X))_{|C_E} = ((true, 0.9), (false, 0.1)$, with $C_E = (X, \{eve\})$, as would LVE^{evi}. Now, LJT^{evi} only needs to find one parcluster containing $Sick(X)$, instead of all parclusters containing $Sick(X)$. Both parclusters $\mathbf{C}_2$ and $\mathbf{C}_3$ contain $Sick(X)$. LJT^{evi} randomly chooses to add g_E to $\mathbf{C}_3$. As the remaining part is optional, we choose against it for efficiency reasons. Evidence entering now is complete.

During message passing, LJT^{evi} sends a message m_{32} from $\mathbf{C}_3$ to $\mathbf{C}_2$. To calculate m_{32}, LJT^{evi} splits g_3 into g'_3 for eve and g''_3 for all other instances. Then, LJT^{evi} eliminates $Treat(X, T)$ from g'_3 and g''_3. Afterwards, LJT^{evi} sends m_{32}, which contains g_E, g'_3, and g''_3, to $\mathbf{C}_2$. In m_{32}, we can easily see that LJT^{evi} propagates evidence to all parclusters containing the PRV of the evidence parfactor as it is an explicit part of the message. Next, we consider soundness and completeness of LJT^{evi}.

Theorem 3. *LJT^{evi} is sound, i.e., computes a correct result for a query* $\mathbf{Q}$ *given an input model* G *and evidence* $\mathbf{E}$*.*

Proof sketch. For certain evidence, LJT^{evi} computes the same result as LJT since they perform the same steps. Given that LJT is correct [3], LJT^{evi} is correct. For uncertain evidence, LJT^{evi} adds evidence parfactors once to a local model of one parcluster. During message passing and query answering, LJT^{evi} then properly accounts for the evidence as an a priori distribution for the given events. □

Theorem 4. *LJT^{evi} is complete for unary evidence, i.e., the time complexity is polynomial in the domain sizes of the model logvars.*

Proof sketch. The completeness results for unary evidence and LVE [5,20] extend also to LJT. Following the same argument as in the proof sketch of completeness for LVE^{evi}, the change from certain to uncertain evidence over one distribution does not lead to groundings, which means the runtime complexity is still polynomial in the domain sizes of the model logvars and the completeness results extend to LJT^{evi}. □

5 Empirical Case Study

We have implemented a prototype version of LJT^{evi} and adapted an LVE implementation by Taghipour (https://dtai.cs.kuleuven.be/software/lve) for uncertain evidence. Given the changes from certain to uncertain events in LVE and LJT and their effects on completeness, we expect implementations of the algorithms to accomplish similar runtimes for certain and uncertain evidence given certain evidence does not cancel out a majority of the model. If certain evidence exists for a majority of the PRVs in a model, the dimension reduction during absorption leaves a very small model, enabling fast query answering. Thus, we use the running example with a domain size of 1000 and add certain evidence $Sick(X) = ((true, 1))$ as well as uncertain evidence $Sick(X) = ((true, 0.8), (false, 0.2))$, covering 0% to 100% of $gr(Sick(X))$ in 10% steps. The query randvar is $Travel(x_{1000})$. We look at two aspects, (i) runtimes for answering a single query with LVE^{evi} and LJT^{evi} and (ii) runtimes of the LJT^{evi} steps.

Figure 3 shows runtimes in milliseconds [ms] for answering a single query with LVE^{evi} (triangles) and LJT^{evi} (circles) with evidence coverage ranging from 0% to 100% on the x-axis. The filled symbols show runtimes for certain evidence. The hollow symbols show runtimes for uncertain evidence. As expected, LJT runtimes are shorter than LVE runtimes since LJT is able to use a smaller submodel compared to the original

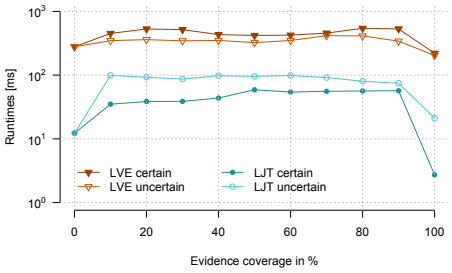

Fig. 3: Runtimes for query answering

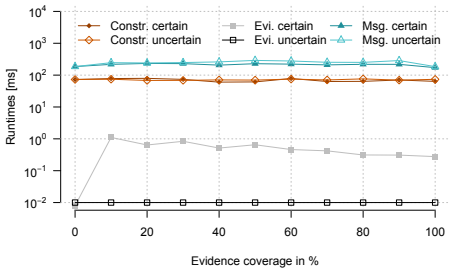

Fig. 4: Runtimes for LJT steps

input model. For LJT^{evi}, certain evidence leads to shorter runtimes than uncertain evidence due to the dimension reduction as well as its preprocessing. Evidence is already handled when LJT^{evi} starts answering the query. And as the submodel for query answering is rather small, the dimension reduction has a comparatively large impact. For LVE, certain evidence leads to larger runtimes as the overall impact of the dimension reduction is not as large and absorption in itself is a rather expensive operation, even though is leads to faster runtimes afterwards. The increase in runtimes from 0% to 10% evidence as well as the decrease in runtimes from 90% to 100%, which occurs for both certain and uncertain evidence, comes from the shattering on the evidence. With 0% and 100% evidence, no splits are necessary, which means smaller models in terms of the number of parfactors to handle.

Figure 4 shows runtimes in milliseconds [ms] of the steps construction (diamond), evidence entering (squares), and message passing (triangles) of LVE^{evi} with evidence coverage ranging from 0% to 100% on the x-axis (filled = certain, hollow = uncertain). Evidence has no influence on construction. Therefore, runtimes are nearly the same for certain and uncertain evidence. Certain evidence leads to larger runtimes as LJT^{evi} absorbs the evidence during this step. Uncertain evidence is simply added to a local model and thus, entering uncertain evidence does not depend on evidence coverage. Message passing with uncertain evidence takes slightly longer than with certain evidence as the dimension reduction also helps during message calculation.

Overall, the case study shows that uncertain evidence leads to similar runtimes for LVE^{evi} and LJT^{evi} compared to certain evidence with a limited scope. Comparing runtimes for domain sizes of 10 to domain sizes of 1000 shows that even though the domain sizes rise by a factor of 100, runtimes only rise by a factor of 2.7 to 8.6 for uncertain evidence and LJT^{evi}. As uncertain evidence basically leads to an additional parfactor and a limited number of splits, we expect further empirical results from [4,3] to also hold for LVE^{evi} and LJT^{evi}, with both algorithms outperforming the ground case.

6 Conclusion

We present LVE^{evi} and LJT^{evi}, versions of LVE andLJT, which incorporate uncertain evidence and allow for similar runtimes as before. We specify how to construct and handle uncertain evidence. LVE^{evi} and LJT^{evi} close the gap to PDBs to also allow for uncertain evidence in probabilistic relational models.

We currently work on learning lifted models. Other interesting algorithm extensions include parallelisation, construction using hypergraph partitioning, and different message passing strategies. Additionally, we look into areas of application to see its performance on real-life scenarios.

References

1. Ahmadi, B., Kersting, K., Mladenov, M., Natarajan, S.: Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training. Machine Learning **92(1)**, 91–132 (2013)
2. Braun, T., Möller, R.: Lifted Junction Tree Algorithm. In: Proc. of KI 2016: Advances in AI. pp. 30–42. Springer (2016)

3. Braun, T., Möller, R.: Counting and Conjunctive Queries in the Lifted Junction Tree Algorithm - Extended Version. In: Postproc. of the 5th Internat. GKR Workshop. Springer (2018)
4. Braun, T., Möller, R.: Parameterised Queries and Lifted Query Answering. In: Proceedings of IJCAI 2018. pp. 4980–4986 (2018)
5. Van den Broeck, G., Davis, J.: Conditioning in first-order knowledge compilation and lifted probabilistic inference. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence. pp. 1–7. AAAI Press (2012)
6. van den Broeck, G., Taghipour, N., Meert, W., Davis, J., Raedt, L.D.: Lifted Probabilistic Inference by First-order Knowledge Compilation. In: IJCAI-11 Proc. of the 22nd International Joint Conference on AI (2011)
7. Chan, H., Darwiche, A.: On the revision of probabilistic beliefs using uncertain evidence. Artificial Intelligence **163**(1), 67 – 90 (2005)
8. Dalvi, N., Suciu, D.: Efficient query evaluation on probabilistic databases. The VLDB Journal—The International Journal on Very Large Data Bases **16**(4), 523–544 (2007)
9. Das, M., Wu, Y., Khot, T., Kersting, K., Natarajan, S.: Scaling Lifted Probabilistic Inference and Learning Via Graph Databases. In: Proc. of the SIAM International Conference on Data Mining. pp. 738–746 (2016)
10. Jeffrey, R.C.: The logic of decision. University of Chicago Press (1990)
11. Lauritzen, S.L., Spiegelhalter, D.J.: Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. Journal of the Royal Statistical Society. Series B: Methodological **50**, 157–224 (1988)
12. Milch, B., Zettelmoyer, L.S., Kersting, K., Haimes, M., Kaelbling, L.P.: Lifted Probabilistic Inference with Counting Formulas. In: AAAI-08 Proc. of the 23rd Conference on AI. pp. 1062–1068 (2008)
13. Pearl, J.: Probabilistic reasoning in intelligent systems: Networks of plausible reasoning (1988)
14. Pearl, J.: On two pseudo-paradoxes in bayesian analysis. Annals of Mathematics and Artificial Intelligence **32**(1-4), 171–177 (2001)
15. Peng, Y., Zhang, S., Pan, R.: Bayesian network reasoning with uncertain evidences. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **18**(05), 539–564 (2010)
16. Poole, D.: First-order Probabilistic Inference. In: IJCAI-03 Proc. of the 18th International Joint Conference on AI (2003)
17. de Salvo Braz, R., Amir, E., Roth, D.: Lifted First-order Probabilistic Inference. In: IJCAI-05 Proc. of the 19th International Joint Conference on AI (2005)
18. Steinhäuser, J., Kühlein, T.: Role of the general practitioner. In: Patient Blood Management, pp. 61–65. Thieme (2015)
19. Suciu, D., Olteanu, D., Ré, C., Koch, C.: Probabilistic databases. Synthesis lectures on data management **3**(2), 1–180 (2011)
20. Taghipour, N., Fierens, D., van den Broeck, G., Davis, J., Blockeel, H.: Completeness Results for Lifted Variable Elimination. In: Proc. of the 16th International Conference on AI and Statistics. pp. 572–580 (2013)
21. Taghipour, N., Fierens, D., Davis, J., Blockeel, H.: Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. Journal of AI Research **47(1)**, 393–439 (2013)
22. Theodorsson, E.: Uncertainty in measurement and total error: tools for coping with diagnostic uncertainty. Clinics in laboratory medicine **37**(1), 15–34 (2017)
23. Thimm, M., Kern-Isberner, G.: On Probabilistic Inference in Relational Conditional Logics. Logic Journal of IGPL **20**(5), 872–908 (2012)
24. Wemmenhove, B., Mooij, J.M., Wiegerinck, W., Leisink, M., Kappen, H.J., Neijt, J.P.: Inference in the Promedas Medical Expert System. In: Conference on Artificial Intelligence in Medicine in Europe. pp. 456–460. Springer (2007)