Taming Reasoning in Temporal Probabilistic Relational Models

Marcel Gehrke and **Ralf Möller** and **Tanya Braun**[1]

Abstract. Evidence often grounds temporal probabilistic relational models over time, which makes reasoning infeasible. To counteract groundings over time and to keep reasoning polynomial by restoring a lifted representation, we present temporal approximate merging (TAMe), which incorporates (i) clustering for grouping submodels without having to ground or compute marginals as well as (ii) statistical significance checks to test the fitness of the clustering outcome. In exchange for faster runtimes, TAMe introduces a bounded error that becomes negligible over time. Empirical results show that TAMe significantly improves the runtime performance of inference, while keeping errors small.

1 Introduction

Temporal probabilistic relational models express relations between objects, modelling uncertainty as well as temporal aspects. Within one time step, a temporal model is considered static. When time advances, the current model state transitions to a new state. Performing inference on such models requires algorithms to efficiently handle the temporal aspect to be able to efficiently answer queries.

Reasoning in lifted representations has a complexity polynomial in domain sizes. But, models dissolve into ground instances through evidence, which no longer permits reasoning in polynomial time, making query answering infeasible for any reasoning algorithm, exact or approximate. Thus, a key challenge during inference in temporal models is to restore a lifted, i.e., non-grounded, representation. Therefore, we formulate and study the problem of keeping reasoning polynomial (KRP) in temporal models to tame the effect of evidence.

First-order probabilistic inference leverages the relational aspect of a static model, using representatives for groups of indistinguishable, known objects, also known as lifting [16]. Poole [16] presents parametric factor graphs as relational models and proposes lifted variable elimination (LVE) as an exact inference algorithm on relational models. Taghipour et al. [20] extend LVE to its current form. To benefit from the ideas of the junction tree algorithm [11] and LVE, Braun and Möller [4] present the lifted junction tree algorithm (LJT) for exact inference given a set of queries. To answer multiple temporal queries, we [8] present the lifted dynamic junction tree algorithm (LDJT), which combines the advantages of the interface algorithm [13] and LJT. Other approaches for temporal relational models perform approximate inference. Ahmadi et al. [1] propose a colour passing scheme to obtain a lifted representation of a dynamic Markov logic network (DMLN) using exact symmetries and extend lifted belief propagation [18] for temporal approximate inference. Further inference algorithms for DMLNs exist [10, 15]. However, to the best of our knowledge, none of these approaches tackle the KRP problem.

[1] University of Lübeck, Germany, email: <surname>@ifis.uni-luebeck.de

For static relational models, approaches exist to approximate symmetries as evidence may ground even a static model [22]. Van den Broeck and Darwiche [21] approximate lifted binary evidence. Singla, Nath, and Domingos [19] propose approximate lifting techniques, which group together distinguishable objects and treat them identically. Venugopal and Gogate [24] form clusters of objects and project the marginal distribution of one object to all objects of a cluster. Van den Broeck and Niepert [23] present an unbiased approach for approximating symmetries. However, these approaches do not account for temporal aspects.

Thus, we present temporal approximate merging (TAMe) as an approach to solve the KRP problem in temporal models. Specifically, TAMe incorporates (i) clustering to group submodels and (ii) statistical significance checks to test the groups to be merged. Model structure and behaviour are captured in a set of functions that define local distributions for the random variables (randvars) in the model. Clustering forms groups of functions based on the similarity between local distributions. The significance checks allow for determining the fitness of the clustering outcome. If the clustering is deemed fit, each group is merged, yielding an unbiased approximation. In exchange for faster runtime, TAMe introduces an indefinitely bounded error.

Boyen and Koller [3] show that for stationary processes, evidence can lead to conditional dependences in temporal probabilistic propositional models, making inference infeasible. They propose to introduce additional randvars to achieve conditional independences between subprocesses even under evidence. Further, Boyen and Koller show that, for any approximation scheme of belief state representations, the error decreases exponentially as the process evolves, making the introduced error bounded indefinitely [3]. Their approach and TAMe are related as in both cases evidence can make inference infeasible. However, TAMe aims at *automatically* restoring a lifted representation. In summary, the cause, namely evidence, is the same for both problems, but the problems are different and the means to make inference possible again differ highly.

TAMe is applicable to different formalisms and algorithms. However, we discuss TAMe as part of LDJT for two reasons: First, when advancing in time, LDJT computes a minimal message that is the source of the most splits of the next time step. Applying TAMe on this message tackles the KRP problem at its root. Second, using TAMe with an exact algorithm allows for attributing errors to merging rather than imprecisions during reasoning. Additionally, TAMe is deterministic in its approximation, thereby, avoiding problems with sampling rates or ergodicity. Empirical results show that TAMe significantly improves the performance of LDJT in general, while keeping errors small and attributable to merging.

In the following, we recap parameterised probabilistic dynamic models (PDMs) as a formalism for specifying temporal relational

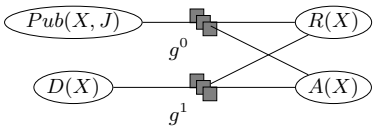

Figure 1. Parfactor graph for G^{ex}

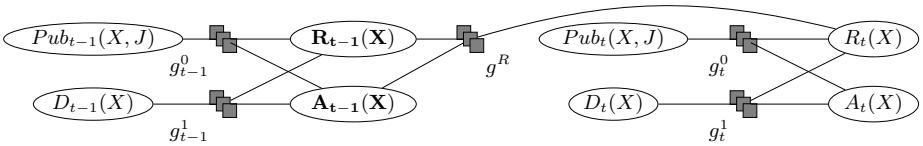

Figure 2. $G^{ex}_{\rightarrow}$ the two-slice temporal parfactor graph for model G^{ex}

models and LDJT for efficient query answering in PDMs. Then, we present TAMe, which includes clustering, significance checks, and merging. Lastly, we evaluate TAMe theoretically and empirically.

2 Preliminaries

We shortly present parameterised probabilistic models (PMs) [5], then extend PMs to the temporal case, resulting in PDMs, and recapitulate LDJT [8, 9], an efficient inference algorithm for PDMs, answering *hindsight*, *filtering*, and *prediction* queries.

2.1 Parameterised Probabilistic Models

PMs combine first-order logic with probabilistic models, using logical variables (logvars) as parameters to represent sets of indistinguishable constructs. As an example, we set up a PM to model the reputation of researchers, inspired by the competing workshop example [12], with a logvar representing researchers. A reputation is influenced by activities such as publishing, doing active research, and attending conferences. A randvar parameterised with logvars forms a parameterised randvars (PRVs).

Definition 1. Let $\mathbf{R}$ be a set of randvar names, $\mathbf{L}$ a set of logvar names, Φ a set of factor names, and $\mathbf{D}$ a set of constants (universe). All sets are finite. Each logvar L has a domain $\mathcal{D}(L) \subseteq \mathbf{D}$. A *constraint* is a tuple $(\mathcal{X}, C_{\mathbf{X}})$ of a sequence of logvars $\mathcal{X} = (X^1, \ldots, X^n)$ and a set $C_{\mathcal{X}} \subseteq \times_{i=1}^{n} \mathcal{D}(X_i)$. The symbol $\top$ for C marks that no restrictions apply, i.e., $C_{\mathcal{X}} = \times_{i=1}^{n} \mathcal{D}(X_i)$. A *PRV* $R(L^1, \ldots, L^n), n \geq 0$ is a syntactical construct of a randvar $R \in \mathbf{R}$ possibly combined with logvars $L^1, \ldots, L^n \in \mathbf{L}$. If $n = 0$, the PRV is parameterless and forms a propositional randvar. A PRV A or logvar L under constraint C is given by $A_{|C}$ or $L_{|C}$, respectively. We may omit $|\top$ in $A_{|\top}$ or $L_{|\top}$. The term $\mathcal{R}(A)$ denotes the possible values (range) of a PRV A. An *event* $A = a$ denotes the occurrence of PRV A with range value $a \in \mathcal{R}(A)$.

We use the randvar names A, D, R, and Pub for attends conference, does research, reputation, and publishes in journals, respectively, and $\mathbf{L} = \{X, J\}$ with $\mathcal{D}(X) = \{x_1, x_2, x_3\}$ (people) and $\mathcal{D}(J) = \{j_1, j_2\}$ (journals). We build boolean PRVs $A(X)$, $D(X)$, $R(X)$, and $Pub(X, J)$. A parametric factor (parfactor) describes a function, mapping argument values to real values (potentials).

Definition 2. We denote a *parfactor* g by $\phi(\mathcal{A})_{|C}$ with $\mathcal{A} = (A^1, \ldots, A^n)$ a sequence of PRVs, $\phi : \times_{i=1}^{n} \mathcal{R}(A^i) \mapsto \mathbb{R}^+$ a function with name $\phi \in \Phi$, and C a constraint on the logvars of $\mathcal{A}$. We may omit $|\top$ in $\phi(\mathcal{A})_{|\top}$. The term $lv(Y)$ refers to the logvars in some element Y, a PRV, a parfactor or sets thereof. The term $gr(Y_{|C})$ denotes the set of all instances of Y w.r.t. constraint C. A set of parfactors forms a *model* $G := \{g^i\}_{i=1}^{n}$. The semantics of G is given by grounding and building a full joint distribution. With Z as the normalisation constant, G represents $P_G = \frac{1}{Z} \prod_{f \in gr(G)} f$.

Let us build the PM $G_{ex}=\{g^i\}_{i=0}^{1}$, shown in Fig. 1, with $g^0 = \phi^0(R(X), A(X), Pub(X, J))_{|\top}$ and $g^1 = \phi^1(R(X), A(X), D(X))_{|\top}$, each with eight input-output pairs (omitted). Next, we present a temporal extension of a PM.

2.2 Parameterised Probabilistic Dynamic Models

We define PDMs based on the first-order Markov assumption. Further, the underlying process is stationary.

Definition 3. A PDM G is a pair of PMs $(G_0, G_{\rightarrow})$ where G_0 is a PM representing the first time step and $G_{\rightarrow}$ is a two-slice temporal parameterised model representing $\mathbf{A}_{t-1}$ and $\mathbf{A}_t$ where $\mathbf{A}_{\pi}$ a set of PRVs from time slice π. The semantics of G is to instantiate G for a given number of time steps, resulting in a PM as defined above.

Figure 2 shows $G^{ex}_{\rightarrow}$ consisting of G^{ex} for time slice $t-1$ and t with *inter*-slice parfactors for the behaviour over time. The parfactor g^R is the *inter*-slice parfactor. For example, we can observe ECAI attendance, which changes over time as, unfortunately, getting papers accepted at consecutive conferences is difficult. Nonetheless, people with high attendance usually have a good reputation.

In general, a query asks for a probability distribution of a randvar given fixed events as evidence.

Definition 4. Given a PDM G, a query term Q (ground PRV), and events $\mathbf{E}_{0:t} = \{E^i_t = e^i_t\}_{i,t}$, the expression $P(Q_t|\mathbf{E}_{0:t})$ denotes a *query* w.r.t. P_G.

The problem of answering a query $P(A^i_{\pi}|\mathbf{E}_{0:t})$ w.r.t. the model is called hindsight for $\pi < t$, *filtering* for $\pi = t$,and *prediction* for $\pi > t$. In this paper, we focus on *filtering* and *prediction* queries.

2.3 Query Answering Algorithm: LDJT

The important property of LDJT [8] for this paper is that LDJT constructs FO jtrees to efficiently answer multiple queries using LVE. The FO jtrees in LDJT contain a minimal set of PRVs to m-separate time steps, which means that state descriptions about these PRVs renders FO jtrees independent from each other. Let us now define an FO jtree, with parameterised clusters (parclusters) as nodes, and present how LDJT proceeds in time.

Definition 5. 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 $|C$ if $C = \top$. 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) A parcluster $\mathbf{C}^i$ is a set of PRVs from G. (ii) For each parfactor $\phi(\mathcal{A})_{|C}$ in G, $\mathcal{A}$ must appear in some parcluster $\mathbf{C}^i$. (iii) If a PRV from G appears in two parclusters $\mathbf{C}^i$ and $\mathbf{C}^j$, it must also appear in every parcluster $\mathbf{C}^k$ on the path connecting

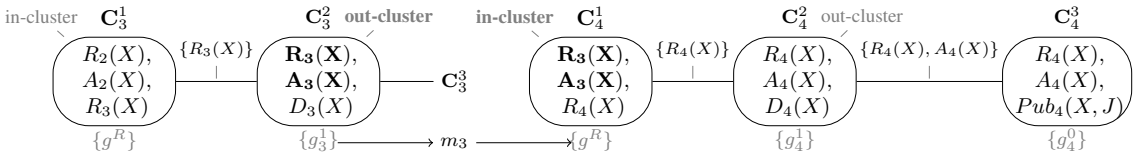

Figure 3. first-order junction tree (FO jtree) J_3 without $\mathbf{C}_3^3$ and FO jtree J_4 connected with m_3

nodes i and j in J. The parameterised set $\mathbf{S}^{ij}$, called *separator* of edge $\{i, j\} \in E$, is defined 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.

To obtain the minimal set of PRVs, which combine all necessary state descriptions to m-separate time steps, LDJT uses the interface PRVs $\mathbf{I}_{t-1}$ of $G_{\rightarrow}$.

Definition 6. The forward interface is defined as

$$\mathbf{I}_{t-1} = \{A_t^i \mid \exists \phi(\mathcal{A})_{|C} \in G : A_{t-1}^i \in \mathcal{A} \wedge \exists A_t^j \in \mathcal{A}\}.$$

PRVs $R_{t-1}(X)$ and $A_{t-1}(X)$ from $G_{\rightarrow}^{ex}$, shown in Fig. 2, make up $\mathbf{I}_{t-1}$. While constructing FO jtree structures, LDJT ensures that the structure J_t has a parcluster containing $\mathbf{I}_{t-1}$, which is called *in-cluster*, and a parcluster containing $\mathbf{I}_t$, which is called *out-cluster*. The *in-* and *out-clusters* allow for reusing the FO jtree structures.

To proceed in time, LDJT calculates a forward message m_t over $\mathbf{I}_t$ using the *out-cluster* of J_t. Hence, m_t contains exactly the necessary state descriptions, as a set of parfactors, to be able to answer queries in the next time step. Afterwards, LDJT adds m_t to the local model of the *in-cluster* of J_{t+1}.

Figure 3 depicts how LDJT proceeds in time. To capture the state at $t = 3$, LDJT sums out the non-interface PRV $D_3(X)$ from the local model and received messages of $\mathbf{C}_3^2$ and saves the result in message m_3. Increasing t by one, LDJT adds m_3 to $\mathbf{C}_4^1$'s local model.

3 Temporal Approximate Merging

In a temporal relational model, evidence can slowly ground the model over time by introducing splits. We propose to name the problem of *finding how to undo splits to retain a lifted solution over time, while keeping any error unbiased and acceptable*, as the KRP problem. Retaining a lifted solution over time means that lifted algorithms run in polynomial time w.r.t. the domain size if a lifted solution exists [14]. To solve the KRP problem, an approach is required to identify any number of clusters based on how similar ϕ's of parfactors are and combine them. To keep the error unbiased and acceptable, groundings need to be accounted for and the identified cluster means have to discriminate the clusters. Unfortunately, to combine *similar* ϕ's, we cannot use the colouring algorithm [1] as it uses *exact* symmetries. Before presenting TAMe, we formulate the problem for PDMs.

Even though LDJT instantiates vanilla FO jtrees for each new time step t, i.e., J_t without splits in local models, m_{t-1} carries over splits caused by evidence. Formally, the problem is that in a model $G_t = \{g_t^i\}_{i=1}^n$ at time step t, many parfactors are split. Whenever evidence leads to a split of a parfactor, the split carries over to subsequent time steps. Thus, G_t has the following form:

$$\{g_t^{i,1}, \ldots, g_t^{i,o}\}_{i=1}^n, o \in \mathbb{N}^+. \tag{1}$$

For each i, the different $g_t^{i,j} = \phi_t^{i,j}(\mathcal{A}^i)_{|C^{i,j}}$, $1 \leq j \leq o$, have the same arguments $\mathcal{A}^i$ but different constraints $C^{i,j}$ and varying functions $\phi_t^{i,j}$ as a result of evidence. The assumption is that some $g_t^{i,j}$ have similar ϕ's as differences introduced by evidence are minimal or otherwise are overcome by model behaviour over time, i.e., potentials align again. Then, one can combine similar ϕ's while introducing only a small and bounded error in exchange for faster reasoning. In the theoretical analysis, we show that the assumption holds, by showing that ϕ's converge, allowing them to be merged, and that the error TAMe introduces is bounded.

Assume that we observe x_1 doing research in time step 3. Then, LDJT enters an evidence parfactor encoding $D(X') = true$ for $\mathcal{D}(X') = \{x_1\}$ in J_3. In J_3, the parcluster $\mathbf{C}_3^2$ contains the PRV $D(X)$. Entering the evidence parfactor in $\mathbf{C}_3^2$ leads to splits in the local model of $\mathbf{C}_3^2$. The parfactor g_3^1 is split into a parfactor for x_1 and into another parfactor with a constraint encoding that the parfactor holds for all instances but x_1. During message passing, the splits carry over. Thus, the parfactors g_3^0 and g^R are also split into two parts. One part for x_1 and another part for all other instances. Therefore, all parfactors about the logvar X are split in the same way into the same amount of groups, i.e., in the local modes of J_3 are parfactors $g_3^{0,1}$, $g_3^{1,1}$ $g^{R,1}$ about x_1 and parfactors $g_3^{0,2}$, $g_3^{1,2}$ $g^{R,2}$ about all other instances.

The idea for restoring a lifted representation is to merge those $g_t^{i,j}$ with similar ϕ's into one parfactor

$$g_t^{i,k} = \phi_t^{i,k}(\mathcal{A}^i)_{|C^{i,k}} \tag{2}$$

where $\phi_t^{i,k}$ represents a merged version of the combined $\phi_t^{i,j}$ and $C^{i,k}$ is a union of the combined $C^{i,j}$. Merging all parfactors that behave similarly for each i leads to a G_t' of the following form with parfactors as in Eq. (2) and $l < o$:

$$\{g_t^{i,1}, \ldots, g_t^{i,l}\}_{i=1}^n \tag{3}$$

With TAMe, we present a merging scheme that takes a model G as given in Eq. (1) and computes a model G' as given in Eq. (3). It is reasonable to apply TAMe to G when transitioning from time step t to $t + 1$ as the transition transfers any splits as well. In general, G may be any parfactor model and one may also transfer the idea to a DMLN model [1]. However, models may be very large, e.g., consist of the union of all local models of an FO jtree J_t, such that finding groups for each i is too costly. Therefore, we propose to make TAMe a subroutine of LDJT. Transitioning from t to $t + 1$ requires computing message m_t, which provides a state description of t that is relevant to $t + 1$. Applying TAMe to m_t prepares a message with fewer groups, leading to fewer splits in J_{t+1}. Additionally, m_t normally has considerably fewer parfactors than G_t. Next, we explain in detail how to get from Eq. (1) to Eq. (3) with TAMe.

3.1 Keeping Reasoning Polynomial with TAMe

Algorithm 1 outlines TAMe to solve the KRP problem. Inputs are a model G, possibly m_t, as well as two additional parameters, radius ϵ

Algorithm 1 Temporal Approximate Merging

```
procedure TAME(Model G, Radius ϵ, Significance τ)
    P ← partitioning of G based on logvars                ▷ Eq. (4)
    for each partition P ∈ P do
        P ← multiply overlapping parfactors               ▷ Eq. (5)
        K ← DBSCAN(P, ϵ, 2, rsim)
        if ANOVA(K, rsim′, τ) rejects H_0 then
            G ← G \ P
            for each cluster K ∈ K do
                G ← G ∪ {K merged}                        ▷ Eq. (7)
```

and significance level τ, which become important later on. The first step is to preprocess G for easier handling in subsequent steps. The main loop describes how a clustering algorithm identifies groups for merging and how groups are merged if TAMe deems the clusters to fit. The upcoming paragraphs discuss the individual steps of Alg. 1.

3.1.1 Model Partitioning

Preprocessing of G is a consequence of the following considerations. A challenge that arises from a model as given in Eq. (1) is that merging parfactors for each i independently of each other may lead to different groups that cause splits again, undoing any merging efforts. Basically, i stands for one parfactor that is split into multiple parfactors with different constraints and ϕ's by evidence. Using an i at random and transferring the grouping of the i parfactors to all other parfactors may lead to unreasonable groups for the other i's. A safe option is to multiply parfactors with overlapping constraints into one parfactor which in a worst case leads to a single cluster and very large parfactors that no longer explicitly represent independencies, which may complicate calculations for messages and queries. Within LDJT, one could trace back if a set of parfactors in m_t originates from the message that has come from the direction of the *in-cluster* to the *out-cluster* as this message contains information about the past and is the origin of the most splits in m_t. Therefore, it may be possible to identify a unique i in m_t as a reasonable source for merging. However, there are no guarantees to find such an i. Instead, we opt to partition the parfactors in G based on the logvars appearing in G into a set $\mathbf{P}$ of sets of parfactors. Each partition $P \in \mathbf{P}$ has a set of logvars $\mathbf{X}_p$ that has been affected in the same way by splitting due to evidence. Formally, P has the form

$$P = \{g_t^{i,1}, \ldots, g_t^{i,o}\}_{i=1}^{n_p} \tag{4}$$

with $lv(g_t^{i,j}) \subseteq \mathbf{X}_p$.

In our example, m_t contains the state descriptions of the PRVs $R_t(X)$ and $A_t(X)$. Both PRVs are parameterised with logvar X. Observing evidence, e.g., for $D_t(X)$ influences X and thereby also $R_t(X)$ and $A_t(X)$ in the same way, i.e., the same splits occur. Hence, for m_t we only have one partition. Even by looking at the complete model instead of only the forward messages, we only have one partition. Observing evidence for X causes splits in all parfactors in the same way. Further, observing evidence for $Pub_t(X, J)$, causes splits in X and J. Thus, here the logvars X and J would be effected in the same way by splitting due to evidence, leading again to one partition as all parfactors are effected by same way by splits due to evidence. In case we would have another logvar Y, which does not occur with X or J in a parfactor, then we would have a second partition. However, in case Y would occur with either X or J in a parfactor, we again would only have one partition.

The next step is to identify groups of parfactors in each partition that behave similarly.

3.1.2 Parfactor Clustering

After partitioning G, each partition $P \in \mathbf{P}$ of the form in Eq. (4) has parfactors whose constraints overlap between all i for each j. Therefore, TAMe multiplies all parfactors with overlapping constraints into one parfactor before starting with identifying groups. If there exists a $g_t^{i,j})$ s.t. $lv(g_t^{i,j}) = \mathbf{X}_p$, then each i refers to m parfactors with the same constraint over all i's for each j, i.e, the constraints are the same at position j for all i's. Then, multiplication in P to combine PRVs with the same constraints boils down to

$$P = \left\{ \prod_{i=1}^{n_p} g_t^{i,1}, \ldots, \prod_{i=1}^{n_p} g_t^{i,o} \right\} = \{g_t^{p,1}, \ldots, g_t^{p,o}\} \tag{5}$$

where multiplying parfactors corresponds to the LVE operation of *multiply*, c.f. [20].

To identify groups of parfactors with similar behaviour, one needs to specify (i) what "similar behaviour" means and (ii) how to find such groups automatically. We first consider the second item, which influences specifying the first item.

TAMe needs to identify an unknown number of groups based on how similar ϕ's are. Density-based clustering groups similar points into an unknown number of groups. Therefore, TAMe uses density-based clustering. For the evaluation, we instantiate TAMe with density-based spatial clustering of applications with noise (DBSCAN) [6, 17] as the clustering approach. In the following, we illustrate how density-based clustering fits into the overall scheme of TAMe using DBSCAN. DBSCAN identifies data points as core points if in their neighbourhoods, determined by a radius ϵ around a point, lie a certain number $minPts$ of other data points. A core data point makes up a cluster along with all the data points in its neighbourhood, which recursively proceeds with the next core data point in the neighbourhood. To determine data points in a neighbourhood, DBSCAN requires a distance function as an input. DBSCAN is able to detect outliers, i.e., points which do not occur in any neighbourhood. For the purpose of clustering parfactors, we set $minPts$ to 2 to be able to cluster even two parfactors. The distance measure should assess how similarly parfactors behave, with 0 meaning identical behaviour and larger values meaning less similar behaviour.

To determine the similarity of the behaviour of two parfactors, one could calculate marginal distributions for a PRV that occurs with split constraints and compare if the marginals are in a certain Δ area. However, marginal distributions could result from different potentials and be similar by chance. The potentials of a parfactor on the other hand specify the current weight for each possible assignment. Thus, in case the ratio of the potentials of two parfactors are similar, they also have similar marginal distributions and behave similarly. Overall, by comparing potentials, TAMe does not need to ground parfactors or compute marginals, which can be costly.

For example, a parfactor mapping to 4 and 2 and another parfactor mapping to 8.1 and 3.9 behave similarly. Both parfactors weight the first assignment about twice as much as the second. Assuming both parfactors are independent from the rest and only have one grounding each, the marginals for $true$ would be 0.667 and 0.675 respectively, i.e., less than 0.01 apart from each other. The same case arises for two parfactors mapping to $\langle 4, 2\rangle$ and $\langle 4.1, 1.9\rangle$ respectively.

Such potentials, when thought of as vectors, have a small angle between them, i.e., a high cosine similarity, which we use to specify "similar behaviour". For the setup of the similarity of two parfactors $g_t^{i,j_1} = \phi_t^{i,j_1}(\mathcal{A}^i)_{|C^{i,j_1}}$ and $g_t^{i,j_2} = \phi_t^{i,j_2}(\mathcal{A}^i)_{|C^{i,j_2}}$, we use a function $rsim : (\times_{i=1}^{n} range(A^i) \mapsto \mathbb{R}^+, \times_{i=1}^{n} range(A^i) \mapsto \mathbb{R}^+) \mapsto \mathbb{R}^+$ that is defined as follows:

$$rsim(\phi_t^{i,j_1}, \phi_t^{i,j_2}) = 1 - \frac{\sum_{\mathbf{a} \in range(\mathcal{A}^i)} \phi_t^{i,j_1}(\mathbf{a}) \cdot \phi_t^{i,j_2}(\mathbf{a})}{\sqrt{\sum_{\mathbf{a} \in range(\mathcal{A}^i)} \phi_t^{i,j_1}(\mathbf{a})^2} \cdot \sqrt{\sum_{\mathbf{a} \in range(\mathcal{A}^i)} \phi_t^{i,j_2}(\mathbf{a})^2}} \quad (6)$$

The result of Eq. (6) lies in the interval $[0, 1]$. The fraction is the definition of the cosine similarity. We calculate 1 minus the fraction to get a "distance" measure, in which a lower value means closer.

As a consequence of $rsim$ with its codomain $[0, 1]$ as the distance function for DBSCAN, ϵ needs to be ≤ 1. Overall, the inputs of DBSCAN are a partition P of parfactors, ϵ, $minPts = 2$, and $rsim$. The radius ϵ trades off cluster sizes with accuracy. With a density based clustering, TAMe does not have to ground parfactors as the grounded factors would have identical ϕ's leading to the same result. The output is a clustering of P, i.e., a set $\mathbf{K}$ of sets in which each $K \in \mathbf{K}$ is a set of parfactors that are assumed to behave similarly.

3.1.3 Fitness of Clustering

The question that remains after clustering is: How good is the clustering? The clustering is highly influenced by the choice of the radius ϵ, which leads to large clusters if set to a high value but may also blur the potentials in the merged parfactor to a higher degree.

One could calculate the error introduced by the clustering w.r.t. a given PRV B by comparing marginal distributions of B before and after merging. However, if a model already is highly shattered, the computational effort can be very high to compute marginal distributions before merging.

DBSCAN clusters together parfactors with a small angle between them. So a clustering fits if the variance of angles within clusters is low and the variance of angles between clusters is high. Analysis of variance (ANOVA) [7] is a statistical method to test for significance of a clustering. In our setup, ANOVA computes the variance of each parfactor in a cluster $K \in \mathbf{K}$ w.r.t. the mean parfactor of K as well as the variance of the mean parfactor of K w.r.t. the mean parfactor of all points in $\mathbf{K}$. Hence, it provides an indication of how good the clustering separates parfactors.

ANOVA is used to accept or reject hypotheses. The default hypothesis is that the means of all clusters are equal. For our problem, the default hypothesis H_0 is that the mean parfactors of the clusters are equal, i.e., are not statistically significant to discriminate clusters. The goal is to be able to reject H_0, that is to say there is more difference between than within clusters. In case TAMe can reject H_0, at least one cluster is significantly different from the others.

To compute a mean parfactor of a cluster K, TAMe calculates the average of all potentials while accounting for groundings. Formally, given a set of parfactors $\{\phi_t^{i,j}(\mathcal{A}^i)_{|C^{i,j}}\}_{j=1}^o$, a mean parfactor $g_t^{i,k} = \phi_t^{i,k}(\mathcal{A}^i)_{|C^{i,k}}$ is determined by

$$\phi_t^{i,k}(\mathbf{a}) = \frac{\sum_{j=1}^o |gr(\phi_t^{i,j}(\mathbf{a})_{|C^{i,j}})| \phi_t^{i,j}(\mathbf{a})_{|C^{i,j}}}{|gr(\phi_t^{i,k}(\mathbf{a})_{|C^{i,k}})|} \quad (7)$$

for each $\mathbf{a} \in range(\mathcal{A}^i)$ and $C^{i,k}$ is a union of the different $C^{i,j}$. TAMe goes through all potentials and for each assignment, adds the current potential, which is multiplied by the number of groundings of the current parfactor. After all potentials for one assignment are added up, TAMe divides the potential by the number of overall groundings to obtain a mean potential.

To illustrate Eq. (7), consider a cluster with 3 parfactors. The first parfactor maps to the potentials 2 and 1 with 2 groundings, the second maps to 3.9 and 1.9 with 5 groundings, and the third maps to 8.1 and 4 with 1 grounding. To calculate the mean potential, TAMe calculates for the first mapping $(2 \cdot 2 + 5 \cdot 3.9 + 1 \cdot 8.1)/8 = 3.95$ and for the second mapping $(2 \cdot 1 + 5 \cdot 1.9 + 1 \cdot 4)/8 = 1.9375$. Thus, the mean parfactor maps to 3.95 and 1.9375 with 8 groundings.

To calculate variances of parfactors, TAMe uses $rsim$ as the clusters have been built based on $rsim$. The intuition behind the choice is that if two parfactors have a very small angle between their potentials, then the variance of the potentials would be close to 0. The variance increases with the angle between potentials. As the number of groundings influences the new potentials, we also include the number of groundings while calculating variances. A parfactor that represents more groundings has a greater weight than one parfactor with one grounding. As we have a grounding semantics, grounding a parfactor with more instances leads to more factors contributing to the full joint distribution.

After computing a mean parfactor $g_t^{i,k}$ for each cluster $K \in \mathbf{K}$ and an overall mean parfactor $g_t^{i,m}$ based on all parfactors in $\mathbf{K}$, ANOVA proceeds to compute the variation between groups, i.e., MSG, and the variation within groups, i.e., MSE, using Eq. (6) and the groundings of parfactors:

$$MSG_{\mathbf{K}} = \frac{1}{l-1} \sum_{K \in \mathbf{K}} |gr(g_t^{i,k})| \cdot (rsim(g_t^{i,k}, g_t^{i,m}))^2$$

$$MSE_{\mathbf{K}} = \frac{1}{m-l} \sum_{K \in \mathbf{K}} \sum_{g_t^{i,j} \in K} |gr(g_t^{i,j})| \cdot (rsim(g_t^{i,j}, g_t^{i,k}))^2$$

where $l = |\mathbf{K}|$, i.e., number of clusters, and $m = |gr(\mathbf{K})|$, i.e., number of overall groundings. Computing $F = \frac{MSG}{MSE}$, ANOVA compares F against a critical value F_{crit}, which depends on τ, $l-1$, and $m-l$ and can be looked up in a pre-computed table. If $F \leq F_{crit}$, TAMe accepts H_0 and discards the clustering. In case TAMe rejects H_0, i.e., $F > F_{crit}$, there is more difference between clusters than within clusters and TAMe proceeds to merging parfactors in clusters.

3.1.4 Merging Parfactors

The new parfactor for each cluster $K \in \mathbf{K}$ is the mean parfactor $g_t^{i,k}$, which is already computed by ANOVA. TAMe replaces P in G with the merged parfactors. Then, TAMe proceeds with the next partition, identifying and checking a clustering for the new partition, until all partitions are processed. The result is a model whose parfactors are merged versions of the input model, partially restoring a lifted representation. Given a forward message m_t, the output is a message that possibly contains fewer groups within logvars and thus, prevents ongoing splitting over time.

3.1.5 Application Cycle

As ANOVA may reject a clustering, TAMe may incur overhead if TAMe cannot merge groups. Therefore, TAMe does not need to be applied at every time step. Normally, the model is slowly grounded over time with evidence, but if the groups behave similarly, which is the case due to the impact of the model, the reoccurring application of the model behaviour results in the potentials being similar enough for TAMe to merge them. Thus, based on how much evidence splits up the model, the interval of how often TAMe should be used as a subroutine needs to be determined.

Next, we look at theoretical implications of TAMe.

4 Theoretical Analysis

We show that TAMe introduces an acceptable, unbiased, and bounded error as well as that TAMe keeps reasoning polynomial.

Proposition 1. *TAMe errors are acceptable and unbiased.*

Due to a density-based clustering, TAMe clusters parfactors with similar ϕ's. ANOVA determines the fitness of clusterings to prevent unacceptable errors. By accounting for groundings during merging, the error is unbiased.

Knowing that TAMe produces acceptable and unbiased errors, let us have a look at theoretical bounds of the approximation error TAMe introduces as well as whether groups with only slightly different evidence do converge, allowing TAMe to keep reasoning polynomial.

Theorem 1. *TAMe introduces a bounded error.*

Proof sketch. A PDM is a Markov process and $G_{\rightarrow}$ describes a temporal transitions model. Given the semantics of a PM, $G_{\rightarrow}$ forms a stochastic transition model Q, which has a so-called minimal mixing rate $\gamma_Q \in]0, 1]$ [3]. The mixing rate γ_Q is the minimal extent to which the model behaviour causes an approximation to converge to the true belief state while transitioning from one time step to the next. TAMe approximates the belief state of the interface $\mathbf{I}_t$ and LDJT computes the transition from t to $t+1$. Thus, the approximation error δ is reduced by the factor $(1-\gamma_Q)$ with each transition. Assuming, that TAMe introduces an error of at most δ for each time step, the expected error up to time step t accumulates to $\delta + (1-\gamma_Q)\cdot\delta + ... + (1-\gamma_Q)^{t-1}\cdot\delta = \sum_{i=0}^{t}\delta\cdot(1-\gamma_Q)^i \leq \sum_{i=0}^{\infty}\delta\cdot(1-\gamma_Q)^i = \delta/\gamma_Q$. For the last step, we apply the geometric series, i.e., $\sum_{i=0}^{\infty}\delta\cdot(1-\gamma_Q)^i = \delta/1-(1-\gamma_Q) = \delta/\gamma_Q$ [3]. Thus, the error is indefinitely bounded by δ/γ_Q. □

For TAMe also the significance check influences the approximation error δ. Before TAMe merges parfactors and thereby, approximates a belief state, TAMe uses a significance check to determine the fitness of a proposed clustering. Therefore, one can use the significance check to obtain a small δ. Now, we prove that TAMe keeps reasoning polynomial.

Theorem 2. *TAMe keeps reasoning polynomial.*

Proof. Without loss of generality, assume we observe evidence for one unary and boolean PRV $B(Y)$. Observing events for multiple instances of logvar Y can split Y into at most three parts, i.e. the true, the false, and the unknown part. Hence, for each time step, LDJT can only assign true, false, or unknown to each grounded PRV of $B(Y)$. For a huge n, where $|\mathcal{D}(Y)| = n$, there always is a large number of ground PRVs with the same subsequence of events. In case these ground PRVs have been split by evidence, the minimal mixing rate γ_Q ensures that the distributions of the ground PRVs converge again. With a subsequence of length l say, the distance between the split distributions of these ground PRVs is reduced by $(1-\gamma_Q)^l$. Therefore, the split distributions converge again and TAMe merges the split distributions at some point in time. Merging parfactors ensures that LDJT calculates a solution in polynomial time w.r.t. domains. □

Now, we use Thm. 2 and the minimal mixing rate to restore the original fully lifted representation.

Corollary 1. *Without new evidence, TAMe obtains a fully lifted representation with the true belief state.*

Proof. During each transition from t to $t+1$, γ_Q ensures that approximated distributions converge to the true distribution as the distributions converge at least by the factor $(1-\gamma_Q)$. Thus, the approximated distributions converge to the true belief state without new evidence provided. Further, all groups have the same origin. Therefore, all groups converge to the same true belief state. Hence, TAMe can merge all groups and thereby, again obtain a fully lifted representation at some point in time. □

Overall we have shown that, TAMe solves the KRP problem with an indefinitely bounded error. Since the underlying distributions of ϕ's converge, TAMe is able to merge ϕ's, allowing TAMe to keep reasoning polynomial. Further, we have shown that TAMe introduces an indefinitely bounded error that is unbiased and acceptable.

5 Evaluation

For the evaluation, we compare runtimes of LDJT with and without TAMe and have a look at the introduced error. We use the model G^{ex} with $|\mathcal{D}(X)| = 100$ and divide these 100 persons equally into symmetry groups, where members of each group behave identically over time. For one time step, each symmetry group has the same evidence, but the evidence can change from one time step to the next. To break symmetries within a group, evidence may be missing with a probability of 0.1 for each person. We split $D(X)$ into 2 to 10 symmetry groups and generate evidence for 20 time steps. For each symmetry group i, LDJT answers $A_{t+\pi}(x_i)$ for $\pi = \{0, 5, 10\}$, i.e., a *filtering* and two *prediction* queries, in each time step t for all 20 time steps. A technical note: The implementations of LVE and thereby, LDJT do not exploit disconnected ground groups to speed up inference, but eliminate all other PRVs to answer queries. Hence, from a runtime perspective we can use G^{ex} for this evaluation.

We vary ϵ and the interval I of how often LDJT applies TAMe. The parameter τ is fixed to 0.005. Based on the problem at hand, an appropriate τ needs to be determined in advance [2]. The three options we evaluate are, from conservative to aggressive: 1) $I = 5$, $\epsilon = 5\cdot10^{-14}$, 2) $I = 5$, $\epsilon = 5\cdot10^{-2}$, and 3) $I = 2$, $\epsilon = 5\cdot10^{-2}$. TAMe with Option 1 mostly merges parfactors that only differ in a scaling factor. TAMe with Options 2 and 3 also merges parfactors that slightly differ in their ratio. With $I = 2$, LDJT calls TAMe every other time step, and with $I = 5$ every fifth time step.

Figure 4 shows runtimes of LDJT without TAMe and with TAMe for the three options. The number of symmetry groups is plotted on the x-axis. With more symmetry groups, evidence can ground the model faster over time. Thus, the runtimes correlate to the number of groups. For 5 symmetry groups, LDJT without TAMe takes about

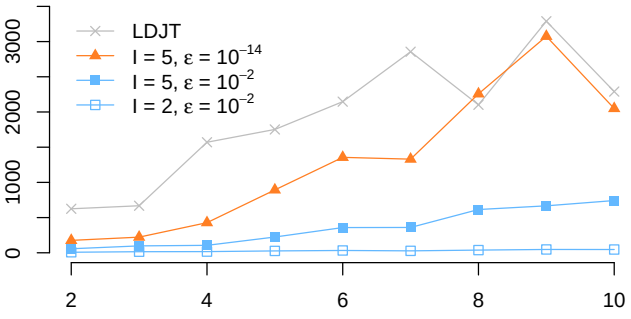

Figure 4. Runtimes [seconds], x-axis: #symmetry groups

twice as long as LDJT with TAMe using the conservative option (1), answering 300 queries for the 20 time steps. However, for 8 symmetry groups, LDJT without TAMe is slightly faster. As merging depends on evidence, which here is randomly generated, TAMe may not always be able to trade off its overhead. TAMe with Option 2 merges more parfactors. Hence, every fifth time step, LDJT answers queries on fewer groups, which are then again split up by evidence. With the most aggressive option (3), LDJT applies TAMe every other time step and thus, answers queries on highly lifted models.

Sampling evidence once can produce an overhead as we have shown above. However, by sampling evidence often as an input for LDJT and then averaging the corresponding runtimes for our different options, we expect smoothed lines w.r.t. symmetry groups and significantly reduced runtimes with TAMe.

In summary, even by only merging parfactors that hardly differ, TAMe merges enough parfactors to improve runtimes of LDJT. TAMe with Options 2 and 3 improves runtimes of LDJT significantly. Overall, TAMe is able to save runtime of LDJT of up to 2 orders of magnitude. Knowing that TAMe can significantly improve the performance of LDJT, we look at the costs of the speed up, namely the introduced absolute error. We calculate the absolute error, by querying all ground instances of the PRV $A_t(X)$ once solely with LDJT and once with LDJT and TAMe. We then report the difference of the results of the marginal queries.

We compare the two options that merge parfactors with a higher angle between them and thereby, do introduce an error. Table 1 shows the absolute error in the marginals for 10 symmetry groups for the most aggressive option and Option 2, when performing filtering, 2 time step prediction, and 4 time step prediction for each instance and each time step. In both cases, for filtering queries, the error is already negligible and decreases for prediction queries. Merging less often (Option 2) has as to be expected a lower average error compared to merging faster (Option 3). For 5 symmetry groups as another representative, Table 2 shows the absolute error. Here, we see the very same behaviour as for 10 symmetry groups. Thus, the empirical evaluation underscores that TAMe keeps reasoning polynomial, introducing only a negligible error. Further, the error converges to the true belief state without new evidence as the prediction queries show. Overall, we observed such a behaviour for all symmetry groups. Next, we investigate the influence of the significance check.

π	Max	Min	Average
0	0.0001512367808	0.0000000000003	0.0000097980420
2	0.0000000837146	0.0000000000000	0.0000000054739
4	0.0000000000470	0.0000000000000	0.0000000000036
0	0.0001537746121	0.0000000001720	0.0000191206488
2	0.0000000851654	0.0000000000001	0.0000000111949
4	0.0000000000478	0.0000000000000	0.0000000000068

Table 1. Error; $\epsilon = 5 \cdot 10^{-2}$, 10 groups, top: $I = 5$ and bottom: $I = 2$

To empirically evaluate the significance check, we run LDJT with TAMe on a model once with and once without the significance check. Table 3 shows the introduced errors for these runs. The maximum error hardly differs between the two runs, which is is due to the error being bounded. Further, the minimum error is lower with the significance check as the significance check does not accept all proposed clusters. Discarding a clustering and thus, not following through with another approximation, the current approximation and the true belief state continue to converge based on the mixing rate. Lastly, the average error without the significance check is around 32% higher. Even though in this case both average errors are negligible on an absolute scale, the average error on a relative scale without the significance check does increase significantly.

π	Max	Min	Average
0	0.0001417171395	0.0000000000001	0.0000124653173
2	0.0000000820954	0.0000000000000	0.0000000073763
4	0.0000000000461	0.0000000000000	0.0000000000048
0	0.0001481776653	0.0000000059350	0.0000209354270
2	0.0000000873810	0.0000000000552	0.0000000129184
4	0.0000000000467	0.0000000000000	0.0000000000071

Table 2. Error; $\epsilon = 5 \cdot 10^{-2}$, 5 groups, top: $I = 5$ and bottom: $I = 2$

	Max	Min	Average
w	0.0002259927071	0.0000000000000	0.0000104567643
w/o	0.0002260554389	0.0000000000168	0.0000137870835

Table 3. Introduced error; with and without significance test

Overall, we show empirically that TAMe does not introduce any unacceptable error due to the significance check and that TAMe keeps reasoning polynomial for LDJT.

6 Conclusion

Evidence often grounds a temporal model over time. Consequently, inference runtimes suffer. Thus, the idea is to use approximate symmetries to restore a lifted representation and thus, keep reasoning polynomial by taming evidence. To the best of our knowledge, we present the first approach solving the KRP problem for temporal relational probabilistic models. The algorithm can be used within any (exact or approximate) temporal inference algorithm. The main idea is that instances of parfactors with similar ratios between potentials behave similarly. To merge parfactors, TAMe uses a message LDJT sends between time steps as this message is smaller than the model and causes splits in the next time step. To identify similar instances, TAMe uses density-based clustering with the cosine similarity as a distance measure, which captures similarity of potentials, allowing to cluster on potentials without having to ground or compute marginals. TAMe applies ANOVA to the clustering result to check if the cluster means significantly discriminate the clusters. We show that TAMe can merge parfactors as their distributions converge and that TAMe introduces an indefinitely bounded error. Additionally, the approximated distributions converge to the true distributions and without new evidence TAMe obtains a fully lifted representation again since the influence of the model over time outweighs a slight difference in evidence. Empirical results show that LDJT with TAMe significantly outperforms LDJT without TAMe. The results support our analysis that TAMe retains a lifted solution, while keeping the introduced error negligible. Hence, TAMe tames the effect of evidence over time and LDJT with TAMe produces fast and precise results.

Future work includes how to approximate evidence [21] to cause fewer splits in temporal models as well as learning temporal mod-

els. Additionally, we investigate approximate symmetries for count-converted PRVs. For counted PRVs with the same number of groundings, TAMe is directly applicable. Otherwise, one could ground the counted PRVs and see if TAMe can merge some of these factors. However, lifting aims at avoiding groundings, leaving room for improvement for TAMe with counted PRVs.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable suggestions. This research originated from the Big Data project being part of Joint Lab 1, funded by Cisco Systems, at the centre COPICOH, University of Lübeck

REFERENCES

[1] Babak Ahmadi, Kristian Kersting, Martin Mladenov, and Sriraam Natarajan, 'Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training', *Machine learning*, **92**(1), 91–132, (2013).

[2] Daniel Benjamin, James Berger, Magnus Johannesson, Brian Nosek, E.-J Wagenmakers, Richard Berk, Kenneth Bollen, Björn Brembs, Lawrence Brown, Colin Camerer, David Cesarini, Christopher Chambers, Merlise Clyde, Thomas Cook, Paul De Boeck, Zoltan Dienes, Anna Dreber, Kenny Easwaran, Charles Efferson, and Valen Johnson, 'Redefine Statistical Significance', *Nature Human Behaviour*, **2**(1), 6, (2018).

[3] Xavier Boyen and Daphne Koller, 'Tractable Inference for Complex Stochastic Processes', in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 33–42. Morgan Kaufmann Publishers Inc., (1998).

[4] Tanya Braun and Ralf Möller, 'Lifted Junction Tree Algorithm', in *Proceedings of KI 2016: Advances in Artificial Intelligence*, pp. 30–42. Springer, (2016).

[5] Tanya Braun and Ralf Möller, 'Parameterised Queries and Lifted Query Answering', in *IJCAI-18 Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4980–4986. International Joint Conferences on Artificial Intelligence Organization, (2018).

[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, 'A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise', in *KDD'96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231. AAAI Press, (1996).

[7] R.A. Fisher, *Statistical Methods for Research Workers*, Edinburgh Oliver & Boyd, 1925.

[8] Marcel Gehrke, Tanya Braun, and Ralf Möller, 'Lifted Dynamic Junction Tree Algorithm', in *Proceedings of the 23rd International Conference on Conceptual Structures*, pp. 55–69. Springer, (2018).

[9] Marcel Gehrke, Tanya Braun, and Ralf Möller, 'Relational Forward Backward Algorithm for Multiple Queries', in *Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference (FLAIRS-32)*, pp. 464–469. AAAI Press, (2019).

[10] Thomas Geier and Susanne Biundo, 'Approximate Online Inference for Dynamic Markov Logic Networks', in *Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence*, pp. 764–768. IEEE, (2011).

[11] Steffen L. Lauritzen and David J Spiegelhalter, 'Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems', *Journal of the Royal Statistical Society. Series B (Methodological)*, **50**(2), 157–224, (1988).

[12] Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling, 'Lifted Probabilistic Inference with Counting Formulas', in *AAAI08 Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, pp. 1062–1068. AAAI Press, (2008).

[13] Kevin Patrick Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning*, Ph.D. dissertation, University of California, Berkeley, 2002.

[14] Mathias Niepert and Guy Van den Broeck, 'Tractability through Exchangeability: A New Perspective on Efficient Probabilistic Inference', in *AAAI14 Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 2467–2475. AAAI Press, (2014).

[15] Tivadar Papai, Henry Kautz, and Daniel Stefankovic, 'Slice Normalized Dynamic Markov Logic Networks', in *NIPS12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, pp. 1907–1915. Curran Associates Inc., (2012).

[16] David Poole, 'First-order probabilistic inference', in *IJCAI03 Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pp. 985–991. Morgan Kaufmann Publishers Inc., (2003).

[17] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu, 'DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN', *ACM Transactions on Database Systems (TODS)*, **42**(3), 19, (2017).

[18] Parag Singla and Pedro M Domingos, 'Lifted First-Order Belief Propagation', in *AAAI08 Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, pp. 1094–1099. AAAI Press, (2008).

[19] Parag Singla, Aniruddh Nath, and Pedro M Domingos, 'Approximate Lifting Techniques for Belief Propagation', in *AAAI14 Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 2497–2504. AAAI Press, (2014).

[20] Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel, 'Lifted Variable Elimination: Decoupling the Operators from the Constraint Language', *Journal of Artificial Intelligence Research*, **47(1)**, 393–439, (2013).

[21] Guy Van den Broeck and Adnan Darwiche, 'On the Complexity and Approximation of Binary Evidence in Lifted Inference', in *NIPS13 Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pp. 2868–2876. Curran Associates Inc., (2013).

[22] Guy Van den Broeck and Jesse Davis, 'Conditioning in First-order Knowledge Compilation and Lifted Probabilistic Inference', in *AAAI12 Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pp. 1961–1967. AAAI Press, (2012).

[23] Guy Van den Broeck and Mathias Niepert, 'Lifted Probabilistic Inference for Asymmetric Graphical Models', in *AAAI15 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 3599–3605. AAAI Press, (2015).

[24] Deepak Venugopal and Vibhav Gogate, 'Evidence-Based Clustering for Scalable Inference in Markov Logic', in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 258–273. Springer, (2014).