24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Lifted Marginal Filtering for Asymmetric Models by Clustering-based Merging

Stefan Lüdtke[1] and **Marcel Gehrke**[2] and **Tanya Braun**[2] and **Ralf Möller**[2] and **Thomas Kirste**[1]

Abstract. Recently, *Lifted Marginal Filtering* (LiMa) has been proposed, an efficient exact Bayesian filtering algorithm for stochastic systems consisting of multiple, (inter-)acting entities where the system dynamics is represented by a *multiset rewriting system* (MRS). The core idea is to represent distributions over multisets more efficiently than by complete enumeration, by exploiting exchangeability that naturally arises due to the MRS dynamics. However, due to system dynamics or observations of individuals, symmetry can break over time, requiring to resort to the original, much larger representation.

In this paper, we propose a method to retain the lifted representation in LiMa. The method identifies groups of lifted multiset states that describe a sufficiently similar distribution of ground multiset states for affording a representation by a single lifted state. Technically, we propose a novel distance measure for lifted states that does not require to completely ground the distribution first, and show how such a single representative for a group of lifted states can be computed. We show empirically that the error induced by this approach is significantly smaller than by limiting the representational complexity conventionally by sampling.

1 Introduction

Many AI tasks like Human Activity Recognition or network analysis involve modeling stochastic systems that consist of multiple, interacting agents or objects. Such systems can be modeled by *dynamic statistic relational models*, like Dynamic Markov Logic Networks [16], Parameterised Probabilistic Dynamic Models [10], CPT-L [27], or Relational Gaussian Models [4]. They specify dynamic probabilistic models on a more abstract level than by complete enumeration of each random variable and dependency, e.g. by combining first-order logic with a probabilistic semantics. For such models, inference on the ground level is typically intractable due to the large number of random variables and factors, and the exponential complexity of ground inference algorithms. Therefore, ground inference methods typically resort to approximation, e.g. by some form of particle filtering [21]. Alternatively, *lifted inference* methods [23, 16] have been devised, which exploit the symmetries in the models by performing computations only once for groups of symmetric random variables.

A major obstacle that prevents the application of lifted inference methods to stochastic systems is that symmetries can break over time such that no symmetries can be exploited anymore and inference algorithms need to operate on the ground distribution.

Lifted Marginal Filtering (LiMa) [18] is such a lifted recursive Bayesian filtering algorithm for *multi-entity systems* (consisting of multiple, (inter-)acting entities, like agents or objects), where the system dynamics is described by a Multiset Rewriting System (MRS). MRSs can directly model complex interactions of entities that cannot be expressed compactly in other formalism, like dynamic Bayesian networks. LiMa avoids the combinatorial explosion by using a suitable factorized representation of distributions over multisets such that some factors can be represented in a parametric way rather than by samples or by complete enumeration (conceptually similar to the Rao-Blackwellized particle filter [5]), as illustrated in the following example:

Example 1 We are modeling a multi-agent activity recognition scenario, where agents can move in an office environment. The goal is to estimate the location of each agent. As long as none of the agents has been observed individually, the random variables that represent the location of agents are *exchangeable*, and a compact representation for that exchangeable distribution can be used. However, when we obtain individual evidence for one of the agents, say *Alice*, the distribution over the random variables (RVs) that describe *Alice* must be handled individually, which increases the representational complexity. If a different agent is observed each time, the representational complexity increases over time, until the representation is completely ground.

Fortunately, the difference between observed and not observed agents typically reduces over time, due to the system dynamics. Thus, intuitively, when the agents become *sufficiently similar* again, we can approximately retain the abstract representation by conflating those similar agents. In this paper, we show how this simple idea is formally realized in LiMa. Specifically, the approach is conceptually based on [11], where sufficiently similar factors are combined into a single *parfactor*: First, groups of lifted states that afford a representation by a single lifted state (that follow certain independence assumptions, Section 3.1) are identified by a clustering-based approach. Afterwards, a single representative state is computed for each cluster.

We provide two non-trivial extensions to this approach to account for the fact that LiMa maintains distributions over *multisets*, instead of conventionally over tuples: We introduce a novel distance measure for lifted multiset states that does not require to compute a ground distribution, based on the idea that the distance of multisets can be defined as the summed pairwise distance of entities under the optimal association of entities from the two multisets (known as Pro-

[1] Institute of Visual & Analytic Computing, University of Rostock, Germany, email: {stefan.luedtke2, thomas.kirste}@uni-rostock.de
[2] Institute of Information Systems, University of Lübeck, Germany, email: {gehrke, braun, moeller}@ifis.uni-luebeck.de

crustes distance in statistical shape analysis [6], Section 3.3). Furthermore, we show how to compute a single representative state for a group of lifted states, by introducing additional independence assumptions that hold only approximately in the original distribution (Section 3.4).

We show empirically that by using this algorithm, a compact representation can be retained (i.e. inference complexity does not grow indefinitely), while the induced error is substantially lower than by limiting the representational complexity conventionally by sampling, as done in particle filtering (Section 5). To the best of our knowledge, this is the first approach that allows lifted filtering in systems with MRS dynamics in situations where no exact symmetries are present.

2 Lifted Marginal Filtering

In this section, we give a brief introduction to LiMa, a lifted recursive Bayesian filtering algorithm based on multiset rewriting. Due to lack of space, we only describe a subset of the algorithm here that is sufficient for discussing the *merging* algorithm later on in Section 3 – for a more complete description of LiMa, we refer to [18].

We start by introducing some background on multiset rewriting systems (MRS) and describe how Bayesian filtering can be realized for MRS. Next, we show how distributions $p(x)$ over multisets x can be factorized, such that some factors are (i) conditionally independent, and (ii) each of the factors represents an *exchangeable* distribution. This allows a more efficient representation of $p(x)$ on the *parametric* level instead of the instance level. Finally, we show how Bayesian filtering can be performed directly on this representation, without computing the original, much larger distribution first.

2.1 Multiset Rewriting Systems

Entities and Multisets Let $\mathcal{E}$ be a set of entities (also called species). A multiset (over $\mathcal{E}$) is a map $x : \mathcal{E} \to \mathbb{N}$ from entities to multiplicities. For entities $e_1, \ldots, e_k \in \mathcal{E}$ and multiplicities $n_1, \ldots, n_k \in \mathbb{N}$ where $n_i > 0$ we write $[\![n_1 e_1, \ldots, n_k e_k]\!]$ to denote the multiset where the multiplicity of e_i is n_i, and the multiplicities of all entities not listed is 0. Multisets are used for describing the *state* of the dynamic system, and thus, in the following, we use the terms *state* and *multiset* interchangeably.

Here, it is sufficient to only consider *flat* entities (i.e. that are atoms with no internal structure), although all following definitions can be extended in a straightforward way to allow for entities that are key-value maps, which allows to model situations where entities have a rich internal structure or even continuous properties.

Actions The system dynamics is described by *actions*. An action a is a triple (c, f, w), where c is a sequence of preconditions, f is an effect function and w is a weight.

It is useful to formulate the preconditions as *constraints*, i.e. boolean functions on entities. The idea of applying an action to a state is to *bind* entities to the preconditions. Specifically, one entity is bound to each element in the sequence of preconditions. A sequence of entities $e = \langle e_1, \ldots, e_n \rangle$ is *compatible* to a constraint sequence $c = \langle c_1, \ldots, c_n \rangle$ when e and c have the same length, and for each i, e_i satisfies c_i. We call a pair (a, e) of action a and sequence of entities e (that is compatible to the constraints of a) an *action instance*. An action instance can be applied to a multiset state x when all entities are contained in the state, i.e. $\text{items}(e) \sqsubseteq x$.

The effect function f manipulates the lifted state based on the bound entities. Specifically, the posterior state x' of applying an action instance to a state x is obtained by applying the effect function f of a to x and the bound entities: $x' = \text{apply}((a, e), x) = f(e, x)$.

We will only consider *simple* effects here that replace each of the bound entities e in x by a different entity e'. For example, the effect of the action goLeft is that e' is one location further to the left than e.

Compound Actions In the scenarios we are concerned with, all entities can act simultaneously between observations. Formally, this is expressed by a *maximally parallel* MRS [1]. In such a system, each state transition is described by a parallel execution of a multiset of action instances, called *compound action*. A compound action k is *compatible* to a state x, if each action instance in k is compatible to x, and the multiset of all bound entities is contained in x.

Probabilistic MRS For Bayesian filtering, we are interested in *probabilistic* MRSs, where a *distribution* of compound actions is defined for each state.

The probability $p(k|x_t)$ of a compound action, given a state x depends on the weights of the individual actions, and the *multiplicity* of the compound action (the number of ways the entities from x can be assigned to the action instances). For more details, see [19].

Probabilistic MRS are typically used for *simulation* studies, where sample trajectories are drawn as follows: Given a state x, calculate all compatible compound actions and their probabilities, sample one of them, apply it to x and iterate this process for the resulting state x'.

2.2 Bayesian Filtering and Problem Statement

We are concerned with *Bayesian filtering* (also called *recursive Bayesian estimation*) for MRSs. That is, the goal is to estimate the posterior distribution $p(X_t|y_{1:t})$ of the *hidden* system state x_t at time t from the previous posterior $p(X_{t-1}|y_{1:t-1})$ and an observation y_t. We assume that the system dynamics is a first-order Markov chain, i.e. it can be described by a *transition model* $p(X_{t+1}|x_t)$. Specifically, the compound action distribution $p(k|s_t)$ described above directly induces a transition model via $p(x_{t+1}|x_t) = \sum_k \mathbb{1}(\text{apply}(k, x_t) = x_{t+1})\, p(k|x_t)$.

Furthermore, we assume that the observation y_t only depends on the state x_t. The *observation model* $p(y_{t+1}|X_{t+1})$ can be, for example, formulated in terms of *features* of the state (like existence of an entity with a certain property value). That is, the observation model is a set of tuples (c, y, p) where c is a constraint that needs to be satisfied in the state x_t, y is the observation, and p is the corresponding observation likelihood.

Estimation of the posterior can be decomposed into two steps: The *prediction* step calculates the distribution after applying the transition model, i.e.

$$p(X_{t+1}|y_{1:t}) = \sum_{x_t} p(X_{t+1}|X_t{=}x_t)\, p(X_t{=}x_t|y_{1:t}). \quad (1)$$

Afterwards, the *correction* step computes the posterior distribution by employing the observation model $p(y_{t+1}|X_{t+1})$:

$$p(X_{t+1}|y_{1:t+1}) = \frac{p(y_{t+1}|X_{t+1})\, p(X_{t+1}|y_{1:t})}{p(y_{t+1}|y_{1:t})} \quad (2)$$

The question that arises here is how to efficiently represent a categorical distribution $p(X_t|y_{1:t})$ of multiset states. This is not straightforward, as the number of multisets can easily grow very large: When k is the overall number of entities in the multiset, and n is the number of possible different entities, then there are $\binom{n+k-1}{k} = \frac{(n+k-1)!}{k!\,(n-1)!}$ different multisets, such that representing $p(X_t|y_{1:t})$ by a set of tuples (x_i, p_i) is infeasible.

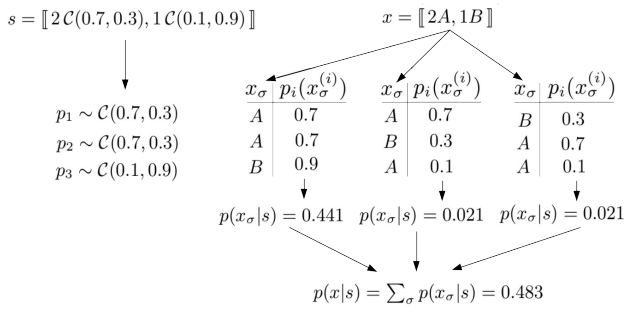

Figure 1: Example illustrating the semantics of lifted states. The probability of a ground state x is obtained by marginalizing over all serializations of x.

2.3 Lifted Representation

A more efficient representation of a distribution $p(x_t|y_{1:t})$ over multisets x_t can be devised by making use of (conditional) independence and exchangeability. Note that, as opposed to conventional distributions over tuples, exploiting independence and exchangeability is not straightforward for distributions over multisets: Naively, the multisets are treated as atoms, so the resulting univariate distribution does not allow any factorization. Conceptually, we define a distribution over tuples, and then explicitly marginalize over all orders of a tuple to define the probability of the corresponding multiset (similar to [9]). Interestingly, this distribution can be *represented* in such a way that the representation is still a multiset (which allows to perform multiset rewriting directly on that representation).

The following example describes the intuition on that representation; a more formal description is given below.

Example 2 Consider a distribution of the multisets $x_1 = [\![2A]\!]$, $x_2 = [\![1A, 1B]\!]$ and $x_3 = [\![2B]\!]$ with $p(x_1) = p(x_3) = 0.25$ and $p(x_2) = 0.5$ (where $A, B \in \mathcal{E}$). The idea is that such a distribution can be decomposed into independent factors *for each entity*[1]. In this example, the (marginal) probability of both entities of being A is 0.5, and their probability of being B is also 0.5. The overall distribution is then represented by a *multiset* of representations of those factors. In the example, both factors are distributed according to the categorical distribution $\mathcal{C}(A{:}0.5, B{:}0.5)$[2]. Thus, overall, the distribution can be represented as the multiset $[\![2\,\mathcal{C}(A{:}0.5, B{:}0.5)]\!]$.

We call such a multiset a *lifted state*. More formally, a lifted state $s \in \mathcal{S}$ is a multiset of *representations* $\mathcal{R}$ of factors, i.e. $S = \mathcal{R} \to \mathbb{N}$. Such a representation $r \in \mathcal{R}$ can be a table or a set of parameters. For example, the string "$\mathcal{N}(0, 1)$" can represent the univariate normal distribution with $\mu = 0$ and $\sigma = 1$. We denote the distribution represented by a representation r as p_r.

A lifted state s defines a distribution of ground states x by the following generative process: Fix an order of the factors in s, and denote the i-th factor in s by $p_i(\cdot|s)$. Then, for each factor p_i, sample a value, and collect all the values in a multiset. In the resulting multiset, there is no information about which value has been sampled from which factor. That is, for the closed-form expression of the distribution induced by a lifted state, we have to consider all possible associations of values and factors.

[1] Note that in the general formulation described in [18], a single factor can describe the distribution of properties of multiple entities, i.e. there can be dependencies between entities.

[2] We denote categorical distributions where value A has probability p_A, value B has probability p_B etc. as $\mathcal{C}(A{:}p_A, B{:}p_B, \dots)$.

Specifically, we *serialize* the ground states x (arrange the elements in a sequence), and define the distribution via marginalizing over all possible serializations. More formally, let $\text{items}(x_\sigma)$ be the multiset obtained by combining all elements in the sequence x_σ into a multiset. We call x_σ a *serialization* of x when $\text{items}(x_\sigma) = x$. For example, the state $x = [\![2A, 1B]\!]$ has three serializations: $x_{\sigma_1} = (A, A, B)$, $x_{\sigma_2} = (A, B, A)$ and $x_{\sigma_3} = (B, A, A)$. We denote the set of all serializations of x as Σ_x, i.e. $\Sigma_x = \{x_\sigma | \text{items}(x_\sigma) = x\}$. The probability of a serialization x_σ, given a lifted state s, is simply the product of the individual probabilities of the i-th value of x_σ (denoted as $x_\sigma^{(i)}$) in distribution p_i:

$$p(x_\sigma|s) = \prod_{i=1}^{n} p_i(x_\sigma^{(i)}|s), \tag{3}$$

where n is the length of serialization x_σ. A *distribution* of lifted states describes a distribution of ground state serializations via

$$p(x_\sigma) = \sum_s p(s)\, p(x_\sigma|s). \tag{4}$$

Thus, this distribution is a *mixture*, with mixture components $p(x_\sigma|s)$ and mixture weights $p(s)$. Finally, the distribution over ground states x is obtained by marginalizing over all serializations of x (each serialization describes an order in which the values in x could have been sampled):

$$p(x) = \sum_{\sigma \in \Sigma_x} p(x_\sigma) \tag{5}$$

It is sometimes convenient to switch the sums in Equations 4 and 5, such that we can also describe the distribution of ground states that is induced by a lifted state s as $p(x|s) = \sum_{\sigma \in \Sigma_x} p(x_\sigma|s)$. We call the elements in the lifted state *lifted entities*, and we call $\text{region}(s) = \{x | p(x|s) > 0\}$ the *region* of s.

An example illustrating the semantics of lifted states is shown in Figure 1: The probability of ground state $x = [\![2A, 1B]\!]$, given the lifted state $s = [\![2\mathcal{C}(0.7, 0.3), 1\mathcal{C}(0.1, 0.9)]\!]$ is obtained by summing the probabilities $p(x_\sigma|s)$ of all serializations x_σ of x.

2.4 Lifted Bayesian Filtering

Prediction Due to the fact that lifted states are still multisets, multiset rewriting can in principle be applied directly to the lifted states. Thus, generating the original, much larger representation of $p(x)$ can be avoided. Specifically, we need to be able to perform two operations: (i) Testing whether a constraint is satisfied for an element of a lifted state s (i.e. for a representation of a distribution of entities), and (ii) applying an effect to a lifted state.

We say that a constraint is satisfied for a representation r when it is satisfied for all entities that have a non-zero probability in p_r. Applying simple effects (as described above) to lifted states is also straightforward: An effect that maps each entity e to an entity e' can be applied to a categorical distribution p_r by performing the same mapping to the *probabilities* of p_r. The following example illustrates how constraints and effects are applied to lifted states.

Example 3 Consider the action `left`, with the precondition $e \neq A$, i.e. the entity e is not already at the leftmost position A, and which has the effect that the entity moves one position to the left. This action is, for example, compatible to both of the entities in the lifted state $s = [\![1\mathcal{C}(A{:}0, B{:}0.5, C{:}0.5), 1\mathcal{C}(A{:}0, B{:}0, C{:}1)]\!]$. Thus, the compound action $[\![2\,\texttt{left}]\!]$ can be applied to s, leading to the posterior state $s' = [\![1\mathcal{C}(A{:}0.5, B{:}0.5, C{:}0), 1\mathcal{C}(A{:}0, B{:}1, C{:}0)]\!]$ (see

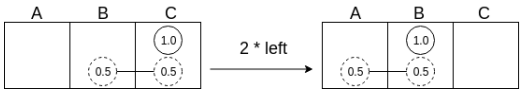

Figure 2: Illustration of the transition semantics: Applying the compound action $[\![\, 2\,\texttt{left}\,]\!]$ to state $[\![\, 1\mathcal{C}(A{:}0, B{:}0, C{:}1), 1\mathcal{C}(A{:}0, B{:}0.5, C{:}0.5)\,]\!]$ leads to the posterior state $[\![\, 1\mathcal{C}(A{:}0, B{:}1, C{:}0), 1\mathcal{C}(A{:}0.5, B{:}0.5, C{:}0)\,]\!]$.

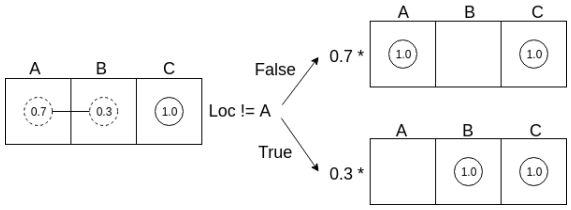

Figure 3: Illustration of a *split* that is necessary due to the precondition of action `left`.

Figure 2). The action is, on the other hand, indeterminate for the entity $\mathcal{C}(A{:}0.7, B{:}0.3, C{:}0)$ – for part of the support of the distribution, the precondition is true, and for a part of the support, the precondition is false. This entity thus requires a split, explained below.

Correction The *correction* step weighs all ground states x by the observation likelihood $p(y|x)$. When $p(y|x)$ is identical for all $x \in \text{region}(s)$, the lifted state s can be weighed by $p(y|x)$ directly, instead of generating all groundings. When a constraint-based observation model, as described in Section 2.2, is used, this property can be checked directly and can be established by splitting. For example, suppose that room A of the office is equipped with a presence sensor that is active when at least one agent is in that room, and has a certain false positive and false negative rate. This observation model can be formalized using the constraint-based formalism with the constraint $e == A$. As for the actions, the constraints can be checked directly on the lifted states s.

Splitting When testing constraints for lifted states, a constraint can be satisfied for just some *part* of the region of a lifted state, and not satisfied for the remaining part. For example, the precondition $c_{\texttt{left}}$ of the action `left` (from Example 3) is indeterminate for the entity $\mathcal{C}(A{:}0.7, B{:}0.3, C{:}0)$ in the state $s = [\![\, 1\,\mathcal{C}(A{:}0, B{:}0, C{:}1)\,1\,\mathcal{C}(A{:}0.7, B{:}0.3, C{:}0)\,]\!]$. More generally, the ground states $x \in \text{region}(s)$ form partitions based on how many entities in x satisfy c.

In such situations, *splitting* operations need to be performed. Splitting a state s on a constraint c results in a set $S = \{(s_i, p_i)\}_{i=1}^{N}$ of weighted lifted states (i.e. a categorical distribution), such that (i) for each $s_i \in S$, all ground states $x \in \text{region}(s_i)$ lie in the same partition regarding c, and (ii) S describes the same distribution of ground states as s, i.e. $\sum_{i=1}^{N} p(x|s_i) = p(x|s)$. Conceptually, this procedure is similar to splitting in Lifted Probabilistic Inference [23].

How splitting is done exactly depends on the kind of distribution that the split is concerned with. In [18], splitting for multivariate hypergeometric distributions is discussed.

For the case considered in this paper (all entities in a lifted state are representations of categorical distributions), splitting can be performed separately for each entity: Assume that we want to split an entity e in s that represents the categorical distribution with $\mathcal{C}(\ldots, V{:}p_v, \ldots)$ on an equality constraint of the form $e == V$. Splitting results in two lifted states s_1 and s_2 with weights p_v and $(1 - p_v)$ that are identical to s, except that e is replaced by e_1 and e_2, respectively, where in e_1, *only* V is possible, and in e_2, V has probability of 0 (and the probabilities of the other values are re-normalized). For example, splitting $s = [\![\, 1\,\mathcal{C}(A{:}0, B{:}0, C{:}1)1\,\mathcal{C}(A{:}0.7, B{:}0.3, C{:}0)\,]\!]$ on the constraint $e == A$ yields the states $s_1 = [\![\, 1\,\mathcal{C}(A{:}1, B{:}0, C{:}0), 1\,\mathcal{C}(A{:}0, B{:}0, C{:}1)\,]\!]$ and $s_2 = [\![\, 1\,\mathcal{C}(A{:}0, B{:}1, C{:}0), 1\,\mathcal{C}(A{:}0, B{:}0, C{:}1)\,]\!]$ with $p(s_1|s) = 0.7$ and $p(s_2|s) = 0.3$ – see Figure 3. To split s completely, this procedure is performed recursively for each entity in s. In the worst case, after splitting, all lifted entities consist only of singleton distributions, i.e. splitting can lead to a complete grounding of the representation.

3 Merging

In this section, we present the main technical contribution of this paper: A systematic way to reduce the representational complexity of a distribution of lifted multiset states, by identifying subsets of lifted states that can be (approximately) represented by a single lifted state. We call this procedure *merging*.

3.1 Problem Statement

As outlined above, splitting increases the representational complexity of the filtering distribution, in the worst case leading to complete a grounding of the distribution over time, thus canceling the benefits of the lifted representation.

However, our intuition is that due to the fact that each lifted state is propagated individually through the transition model, the Kullback-Leibler divergence (KLD) between each pair of lifted states will decrease, and thus there is a chance for finding a set of lifted states that are "sufficiently similar" so that they can be represented by a single lifted state. In the following, we show how to make use of this intuition.

Specifically, the goal is to identify subsets $G \subseteq S$ of lifted states, such that the ground distribution induced by G can be approximated by a single lifted state. Formally, a subset G of the lifted states describes the following ground distribution (by Equations 4 and 5):

$$p(x|G) = \frac{1}{p(G)} \sum_{\sigma \in \Sigma_x} \sum_{s \in G} p(s)\, p(x_\sigma|s) \tag{6}$$

with $p(G) = \sum_{s \in G} p(s)$. Now, we assume that the distribution $p(x_\sigma|G) = 1/p(G) \sum_{s \in G} p(s)\, p(x_\sigma|s)$ factorizes into independent factors (this assumption will allow us to represent the distribution by a single lifted state, shown below). Note that this is an *additional* assumption to the requirement for lifted states made in Equation 3, where we assumed that the distributions $p(x_\sigma|s)$ is *conditionally independent* on s, whereas here, the distribution $p(x_\sigma|G)$ must be fully independent (in the group G). Using this assumption, the sum and product can be switched:

$$\begin{aligned} \tilde{p}(x|G) &= \frac{1}{p(G)} \sum_{\sigma \in \Sigma_x} \sum_{s \in G} p(s) \prod_{i=1}^{n} p_i(x_\sigma^{(i)}|s) \\ &= \sum_{\sigma \in \Sigma_x} \prod_{i=1}^{n} \sum_{s \in G} \frac{p(s)}{p(G)}\, p_i(x_\sigma^{(i)}|s) \end{aligned} \tag{7}$$

The interesting fact that can be observed now is that this distribution can be described by a *single* lifted state s_G, that contains the factors

$p_i(x_\sigma^{(i)}|s_G) = \sum_{s\in G} \frac{p(s)}{p(G)}\, p_i(x_\sigma^{(i)}|s)$: Each of those factors can be directly computed from the states $s \in G$, i.e. instead of maintaining all states s, only the single state s_G needs to be maintained.

Here, the contraction theorem by Boyen and Koller (Theorem 3 in [2]) applies, which states that propagating the distributions p and $\tilde{p}$ through the transition model leads to a constant-factor decrease in their KLD. Thus, the error (in terms of KLD) between the true distribution p and approximate distribution $\tilde{p}$ cannot grow indefinitely. However, the error can still be large, depending on how well $\tilde{p}$ approximates the true distribution p, i.e. how well the independence assumptions hold in p. Thus, our goal is to identify subsets G such that the independence assumptions hold (approximately) in $p(x_\sigma|G)$. Note that this problem is a special case of the problem of simplifying a mixture model, where the lifted states are the mixture components.

3.2 A Special Case: Independence by Identity

A specific case where $p(x_\sigma|G)$ factorizes exactly occurs when for all lifted states $s_j \in G$, the factors $p_i(x_\sigma^{(i)}|s_j)$ are identical (i.e. the lifted states all describe the same ground distribution $p(x_\sigma|s_j)$). In this case, the factors do not depend on the state s_j, but just on the group G, i.e. for each j, $p(x_\sigma^{(i)}|s_j) = p(x_\sigma^{(i)}|G)$. Therefore,

$$\begin{aligned} p(x|G) &= \frac{1}{p(G)} \sum_{\sigma\in\Sigma_x} \sum_{s\in G} p(s) \prod_{i=1}^{n} p_i(x_\sigma^{(i)}|s) \\ &= \frac{1}{p(G)} \sum_{s\in G} p(s) \sum_{\sigma\in\Sigma_x} \prod_{i=1}^{n} p_i(x_\sigma^{(i)}|G) \\ &= \sum_{\sigma\in\Sigma_x} \prod_{i=1}^{n} p_i(x_\sigma^{(i)}|G) = \tilde{p}(x|G). \end{aligned} \tag{8}$$

In this paper, we are concerned exactly with this case: We identify groups G of lifted states that all describe (approximately) the same distribution of ground states, and then represent those groups by a single lifted state by assuming independence between the factors. Later, we show a slightly more general case, where not all i-th factors must be (approximately) identical, but there is a mapping between the factors such that they are identical. Overall, the algorithm works as follows:

(i) Compute pairwise distances of ground distributions induced by lifted states.
(ii) Find groups G of lifted states with similar ground distributions (by clustering).
(iii) Compute a single representative s_G for each group.

In the following, we discuss how steps (i) and (iii) of the algorithm can be computed efficiently, without requiring to compute the ground distribution first.

3.3 Distance Measures for Lifted States

We start by discussing how the distance between ground state distributions induced by lifted states s_1 and s_2 can be computed efficiently. Naively, we can directly compute the ground distributions $p(X|s_1)$ and $p(X|s_2)$ (via Equation 5, or by repeated splitting), and then compute the distance of these distributions, e.g. using the Jensen-Shannon divergence (JSD)

$$JSD(p,q) = \frac{1}{2}\, D(p,m) + \frac{1}{2}\, D(q,m), \tag{9}$$

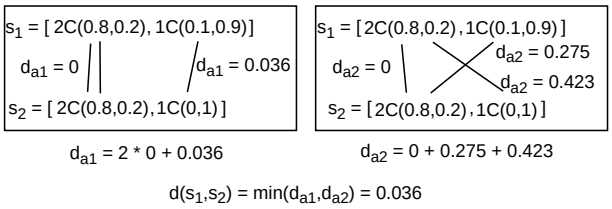

Figure 4: Example of the marginal Procrustes distance. Edges represent the pairing of the entities; the marginal distance of the entities is shown next to the edges. In this example, two pairings a_1 and a_2 are possible, leading to different distance values. The Procrustes distance is the minimal distance value.

where $m = \frac{1}{2}(p+q)$ and D is the Kullback-Leibler divergence (KLD)

$$D(p,q) = -\sum_x p(x) \log \frac{q(x)}{p(x)}. \tag{10}$$

However, computing the distance this way is highly inefficient, as it requires to ground the distribution completely. Thus, our goal is to derive an (approximate) measure of lifted state distance, that does not require to compute the ground distribution.

For distributions that decompose into independent factors, the KLD can be computed on the factors: $D(p,q) = D(p_1,q_1) + D(p_2,q_2)$ for $p(x,y) = p_1(x)\,p_2(y)$ and $q(x,y) = q_1(x)\,q_2(y)$. However, this fact cannot be exploited in our case, as the ground distribution does not factorize directly, but involves summing over serializations (see Equation 5).

Instead, we employ the following intuition: To compare two lifted states s_1 and s_2, we perform a pairwise comparison of the *entities* in s_1 and s_2, and define the overall distance of the states as the summed pairwise distance of entities, given an optimal association of the entities. First, we define a bijective function a, that maps the factors p_i^1 from s_1 to the factors p_j^2 from s_2 (for now, assume that a is given, we will later discuss how to obtain such a function). For example, for the lifted states $s_1 = [\![\, 2\langle \mathcal{C}(0.8, 0.2\rangle), 1\langle \mathcal{C}(0.1, 0.9\rangle \,]\!]$ and $s_2 = [\![\, 2\langle \mathcal{C}(0.8, 0.2\rangle), 1\langle \mathcal{C}(0, 1\rangle \,]\!]$ shown in Figure 4, two distinct ways to map their factors exist: Either $a_1(p_1^1) = p_1^2$, $a_1(p_2^1) = p_2^2$, $a_1(p_3^1) = p_3^2$, or $a_2(p_1^1) = p_3^2$, $a_2(p_2^1) = p_2^2$, $a_2(p_3^1) = p_1^2$.

For each pair of factors $p_1 \in s_1$ and $p_2 \in s_2$, the JSD of their (marginal) distributions can be computed. The overall distance d of s_1 and s_2 is the sum of the individual JSDs, i.e.

$$d_a(s_1,s_2) = \sum_{p_1\in s_1} JSD(p_1, a(p_2)). \tag{11}$$

The intuition here is that there is an *optimal* association of the entities with minimal error that defines the overall distance. In statistical shape analysis, this intuition is formalized by the *Procrustes distance* [6]. It measures the distance between two sets of points and is defined as the *minimal* distance between the sets of points, for all possible Euclidean transformations (reflections, rotations and translations) of one of the sets. A closely related concept is used in the multi-target tracking literature when comparing true and estimated targets, known as optimal sub-pattern assignment distance [25]. For computing the distance between two lifted states, we can employ the same idea, by using the *minimal* distance over all possible entity mappings:

$$d(s_1,s_2) = \min_a d_a(s_1,s_2) \tag{12}$$

We call the distance d the *marginal Procrustes distance* between s_1 and s_2. An example is shown in Figure 4: For the states s_1 and s_2 shown in the figure, two distinct ways to map their entities exist, leading to distances d_{a_1} and d_{a_2}. The Procrustes distance is the minimum of those distance values.

Note that when there are n *distinct* entities (species) in s_1 and s_2, there are $n!$ possible mappings a. Still, an optimal mapping can be obtained in $\mathcal{O}(n^3)$ using the Hungarian algorithm [22]. The distance d can be approximated in *constant* time by drawing a constant number of *samples* from the mappings a, and taking the minimal distance of all samples.

3.4 Finding a Single Representative

After computing all pairwise state distances, sets of states that are "sufficiently similar" (that afford a representation by a single lifted state) are determined by DBSCAN [7], a density-based clustering algorithm. In principle, any clustering algorithm could be used for this step. We choose a density-based clustering approach here, as this way, we do not need to specify the number of clusters in advance, and do not need to make a-priori assumptions about the shape of the clusters. For each cluster G of lifted states, a single representative s_G needs to be computed.

Equation 7 already indicates how to compute s_G: Each factor in s_G is computed as $p_i(x_\sigma^{(i)}|s_G) = \sum_{s \in G} \frac{p(s)}{p(G)} p_i(x_\sigma^{(i)}|s)$. However, this way, it is assumed that the i-th factor (using some canonical ordering) of all states $s \in G$ is used to compute the factor $p_i(x_\sigma^{(i)}|s_G)$. However, as already discussed above, the entities (i.e. the factors) do not have a meaningful order, and thus we can use any association of the entities to compute the new factors. In fact, the *optimal* association of entities (such that the overall distance of the states is minimal) has already been computed for the Procrustes distance above (Equation 12).

More formally, let a_{jk} be the mapping between the factors of s_i and s_k with minimal Procrustes distance. Given a set of set of lifted states $G = \{s_1, \ldots, s_n\}$, the factors $p_i(x_\sigma^{(i)}|s_G)$ are computed as follows:

$$p_i(x_\sigma^{(i)}|s_G) = \sum_{s_j \in G} \frac{p(s)}{p(G)} a_{ij}(p_i)(x_\sigma^{(i)}|s) \tag{13}$$

For the categorical distributions considered here, this sum can again be represented as a categorical distribution. For example, consider the states shown in Figure 4, and suppose that $G = \{s_1, s_2\}$ needs to be merged. Suppose that $p(s_1) = 0.4$ and $p(s_2) = 0.1$. The association a_1 leads to the minimal Procrustes distance. Therefore, we obtain $p_1(x_\sigma^{(1)}|s_G) = p_2(x_\sigma^{(2)}|s_G) \sim \mathcal{C}(0.8, 0.2)$ and $p_3(x_\sigma^{(3)}|s_G) \sim 0.4/0.5\ \mathcal{C}(0.1, 0.9) + 0.1/0.5\ \mathcal{C}(0, 1) = \mathcal{C}(0.08, 0.92)$ and thus $s_G = [\![2\langle \mathcal{C}(0.8, 0.2) \rangle, 1\langle \mathcal{C}(0.08, 0.92) \rangle]\!]$.

4 Experimental Evaluation

The goal of the experimental evaluation is to (a) investigate the error (in terms of JSD) that is induced by merging, in relation to the decrease in representational complexity (i.e. number of maintained lifted states), and (b) the runtime of the merging algorithm.

We use the simple scenario introduced in Example 1 for evaluation, where $n \in \{1, \ldots, 5\}$ entities move in 4 rooms (the number of ground states in this scenario is exponential to the number of entities). Entities can either stay at their current location (action `noop`) or move to the room to the left (action `left`), when they are not already at the leftmost room. We set the lifted state $[\![n\,\mathcal{C}(0.4, 0.3, 0.2, 0.1)]\!]$ as initial state (where n is the number of entities), and performed up to $t = 20$ prediction operations. In this scenario, no observations are used (i.e. no correction step was performed), as this would only add another layer of complexity that is not necessary for investigating the research question considered here.

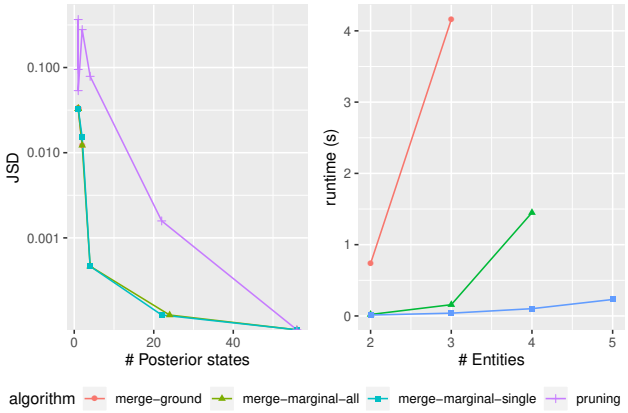

Figure 5: Left: Tradeoff between error (in terms of JSD) and representational complexity of the distribution (i.e. the number of lifted states). All merging algorithms provide the same approximation quality, and pruning leads to a higher error for a given representational complexity. Right: Overall runtime of the different merging algorithms with respect to number of entities in the state (which has an exponential effect on state space size).

For evaluating (a), we compute the JSD between the true and approximate distribution (that is obtained by repeatedly performing merging after each prediction), for different maximum numbers of maintained states. Furthermore, we compare merging with the conventional, sampling-based method to reduce the number of states in particle filtering: *Sampling* a fixed number of states from the distribution $p(x_t|y_{1:t})$ and using these samples as an approximate representation of the distribution. Specifically, we use the optimal (in the sense of least squared error) and unbiased resampling algorithm for categorical distributions presented in [8] that avoids state duplicates (also called *pruning* in this context), for which we also compute the JSD between the true and approximate distributions.

Note that the merging procedure does not require to set the number of posterior states a priori, whereas the pruning algorithm needs to be provided with that number. Thus, to allow a fair comparison between merging and pruning, we proceed as follows: For a given state distribution $p(x_t|y_{1:t})$ and ϵ value of DBSCAN, merging is performed, resulting in a number n of posterior states. Then, pruning is performed, where the number of posterior states is set to n.

To assess (b), we compare the runtime of three merging algorithms, that are different only in the employed distance measure: The (exact) JSD of the ground distribution (`ground`), the marginal Procrustes distance by considering all distinct mappings of entities (`marginal-all`), and the marginal Procrustes distance by considering only a single, random mapping (`marginal-single`). Here, merging is attempted for a set of lifted states that has been obtained by 10 prediction steps from the initial state.

5 Results

Figure 5 (right) shows the runtime of the different merging algorithms. The computationally most expensive operation in all cases

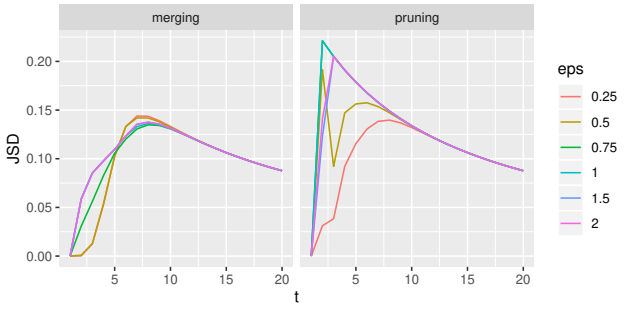

Figure 6: Development of JSD between original distribution and approximated distribution over time, when repeatedly performing approximation operations (merging or pruning) at each time step during inference. Merging achieves a lower error than pruning (in terms of JSD).

is computing all pairwise state distances. We see that using the `ground` and `marginal-all` distances quickly becomes infeasible, as they both have exponential runtime (with respect to the number of entities).

Next, we investigate the size of the error that is induced by merging in comparison to the size of the error induced by pruning. Figure 5 (left) shows the ground JSD between a merged (or pruned) distribution and the original (true) distribution, in relation to the number of allowed posterior states. The true distribution has been obtained by performing 10 prediction steps from the initial state $[\![3\mathcal{C}(0.4, 0.3, 0.2, 0.1)]\!]$, which is represented by 56 lifted states. In general, allowing fewer posterior states naturally leads to a larger error, and allowing 56 states leads to a JSD of 0. For a given number of allowed posterior states, merging achieves a lower JSD than pruning (note the logarithmic scale of the plot). Furthermore, the JSD does not depend on the chosen distance measure. Thus, it is sufficient to compute the pairwise state distances via `marginal-single` (which is fastest), and still achieve a better approximation quality than by pruning.

Figure 6 shows the development of the JSD between the original and the approximated distribution over time, when either performing pruning (right) or merging based on the `marginal-single` distance. The overall error does not grow indefinitely, which is a direct consequence of the contraction theorem [2]. The plot shows that merging achieves a lower JSD than pruning for a given representational complexity, even when it is repeatedly applied during filtering.

6 Related Work

Several approaches for inference in relational dynamic models have been devised. These methods are based on two general principles: Rao-Blackwellization (representing some factors of the distribution on the parametric level, instead of explicitly or by samples), and Lifted Inference (grouping redundant/symmetrical factors of a graphical model into a single representation). Examples of methods that make use of Rao-Blackwellization are the Logical Particle Filter [30], Stochastic Relational Processes [27], and the Relational Particle Filter [12]. Methods for inference in dynamic systems that make use of Lifted Inference include the Lifted Dynamic Junction Tree (LDJT) algorithm [10], and the Relational Kalman Filter [4]. All of these methods potentially suffer from the increase in representational complexity over time due to system dynamics or observations. That is, a parametric distribution must be replaced by an explicit representation or samples, or a lifted representation becomes ground.

For such dynamic inference approaches, only few methods have been devised to approach this problem. For the relational Kalman filter, a method has been proposed that regroups Gaussian potentials that have been split by averaging their covariance matrices [3]. The method devised for the LDJT algorithm [11] is more closely related to our approach: It restores a lifted representation by identifying similar factors by density-based clustering and cosine distance function (as factors that are scaled differently can still be similar). We also use density-based clustering to find mergeable components, but need to account for the fact that LiMa is handling *multisets*, which requires a novel distance measure, and a novel method to merge the multisets.

More generally, in the Lifted Inference literature, methods for retaining compact representation have been devised [29, 26]. They work by identifying (approximate) symmetries (graph automorphisms) in the graphical model, e.g. by using color passing algorithms [17], or low-rank matrix approximations [28]. Unfortunately, such methods are not directly applicable here, as they work on the graphical model representation of the distribution, but the multiset rewriting-based system dynamics considered here cannot be represented compactly by a graphical model that exhibits the symmetrical structure (which is why we devised the lifted multiset representation in the first place).

The merging problem discussed here is a special case of mixture model simplification, where the lifted states are the mixture components. However, in contrast to existing methods, we are concerned with a *discrete* state space, i.e. the mixture components (the lifted states) are discrete distributions of ground states. Thus, methods that rely on the fact that the mixture components are continuous distributions (e.g. that require Gaussian mixtures [14, 13] or work by fitting smooth distributions that best approximate a set of mixture components [31]) cannot be used directly. Furthermore, computing distances between mixture components is typically simple in continuous domains: For example, the KLD between Gaussian mixture components can be computed in closed form [24]. Instead, the challenging aspect in our case is to efficiently compute distances between mixture components.

7 Discussion

In this paper, we proposed a method for keeping the distribution representation of Lifted Marginal Filtering compact (and thus inference efficient). The method identifies groups of lifted states that represent a sufficiently similar ground distribution, as for such a group, the marginal distributions of the entities are (approximately) independent, and thus it can be approximated by a single lifted state. We showed empirically that this approach results in a lower approximation error than pruning, when the same number of states is retained.

In this paper, we only considered the special case where a group of states affords a representation by a single lifted state because of sufficiently similar ground distributions. We believe that this is a common case that will be sufficient for many practically relevant filtering problems. Furthermore, the approach is not limited to categorical distributions as factors in lifted states, as discussed here, but can be applied to all distributions where the group factors shown in Equation 13 can be represented compactly.

However, in general, states that represent completely different distributions can also afford a representation by a single lifted state, e.g. when they represent different "regions" of some parametric distribution. A practically relevant case are distributions over permutations, which arises e.g. when tracking identities of multiple, interacting agents or objects [20]. In such cases, it can be necessary to find

a specific subset of states that afford a unifying representation. However, checking all subsets of lifted states is clearly infeasible. Further classifying the cases where mergeable subsets can be identified efficiently is an interesting direction for future research.

For the case of filtering distributions over *permutations*, efficient approximate methods for limiting the representational complexity already exist, that work by employing methods from noncommutative Fourier analysis to obtain a low-frequency approximation [15]. Combining such methods with the methods proposed here could be one of the next step towards an efficient Bayesian filtering algorithm for large, dynamic relational models with symmetry breaks.

REFERENCES

[1] R. Barbuti, F. Levi, P. Milazzo, and G. Scatena, 'Maximally Parallel Probabilistic Semantics for Multiset Rewriting', *Fundamenta Informaticae*, **112**(1), 1–17, (2011).

[2] 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, (1998).

[3] J. Choi, E. Amir, T. Xu, and A. Valocchi, 'Learning Relational Kalman Filtering.', in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 2539–2546, (2015).

[4] J. Choi, A. Guzman-Rivera, and E. Amir, 'Lifted Relational Kalman Filtering.', in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 2092–2099, (2011).

[5] A. Doucet, N. De Freitas, K. Murphy, and S. Russell, 'Rao-Blackwellised particle filtering for dynamic Bayesian networks', in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 176–183. Morgan Kaufmann Publishers Inc., (2000).

[6] IL Dryden and KV Mardia, *Statistical analysis of shape*, Wiley, 1998.

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al., 'A density-based algorithm for discovering clusters in large spatial databases with noise.', in *Kdd*, volume 96, pp. 226–231, (1996).

[8] Paul Fearnhead and Peter Clifford, 'On-line inference for hidden markov models via particle filters', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **65**(4), 887–899, (2003).

[9] Peter A Flach, Elias Gyftodimos, and Nicolas Lachiche, 'Probabilistic reasoning with terms', *Linkoping Electronic Articles in Computer and Information Science*, **7**(011), (2002).

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

[11] Marcel Gehrke, Tanya Braun, and Ralf Möller, 'Taming Reasoning in Temporal Probabilistic Relational Models', in *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*. IOS Press, (2020).

[12] T. Geier and S. Biundo, 'Approximate online inference for dynamic markov logic networks', in *23rd IEEE International Conference on Tools with Artificial Intelligence*, pp. 764–768. IEEE, (2011).

[13] Jacob Goldberger, Hayit K Greenspan, and Jeremie Dreyfuss, 'Simplifying mixture models using the unscented transform', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**(8), 1496–1502, (2008).

[14] Jacob Goldberger and Sam T Roweis, 'Hierarchical clustering of a mixture model', in *Advances in Neural Information Processing Systems*, pp. 505–512, (2005).

[15] J. Huang, C. Guestrin, and L. Guibas, 'Fourier Theoretic Probabilistic Inference over Permutations', *Journal of Machine Learning Research*, **10**, 997–1070, (June 2009).

[16] K. Kersting, B. Ahmadi, and S. Natarajan, 'Counting belief propagation', in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pp. 277–284, (2009).

[17] Kristian Kersting, Martin Mladenov, Roman Garnett, and Martin Grohe, 'Power iterated color refinement', in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, (2014).

[18] Stefan Lüdtke, Max Schröder, Sebastian Bader, Kristian Kersting, and Thomas Kirste, 'Lifted Filtering via Exchangeable Decomposition', in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, (2018).

[19] Stefan Lüdtke, Max Schröder, and Thomas Kirste, 'Approximate probabilistic parallel multiset rewriting using mcmc', in *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pp. 73–85. Springer, (2018).

[20] Stefan Lüdtke, Kristina Yordanova, and Thomas Kirste, 'Human activity and context recognition using lifted marginal filtering', in *Proceedings of the 15th Workshop on Context Modeling and Recognition (CoMoRea)*, pp. 83 – 88, (2019).

[21] D. Nitti, T. De Laet, and L. De Raedt, 'Probabilistic logic programming for hybrid relational domains', *Machine Learning*, **103**(3), 1–43, (2016).

[22] Christos H. Papadimitriou and Kenneth Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., 1982.

[23] D. Poole, 'First-order probabilistic inference', in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pp. 985–991, (2003).

[24] Andrew R Runnalls, 'Kullback-leibler approach to gaussian mixture reduction', *IEEE Transactions on Aerospace and Electronic Systems*, **43**(3), (2007).

[25] Dominic Schuhmacher, Ba-Tuong Vo, and Ba-Ngu Vo, 'A consistent metric for performance evaluation of multi-object filters', *IEEE transactions on signal processing*, **56**(8), 3447–3457, (2008).

[26] Parag Singla, Aniruddh Nath, and Pedro M Domingos, 'Approximate lifting techniques for belief propagation', in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, (2014).

[27] I. Thon, N. Landwehr, and L. De Raedt, 'Stochastic relational processes: Efficient inference and applications', *Machine Learning*, **82**(2), 239–272, (February 2011).

[28] Guy Van den Broeck and Adnan Darwiche, 'On the complexity and approximation of binary evidence in lifted inference', in *Advances in Neural Information Processing Systems*, pp. 2868–2876, (2013).

[29] 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).

[30] L. Zettlemoyer, H. Pasula, and L. Kaelbling, 'Logical particle filtering', in *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, (2008).

[31] Kai Zhang and James T Kwok, 'Simplifying mixture models through function approximation', *IEEE Transactions on Neural Networks*, **21**(4), 644–658, (2010).