Lifting Factor Graphs with Some Unknown Factors

Malte Luttermann[1,2][0009−0005−8591−6839], Ralf Möller[1,2][0000−0002−1174−3323], and Marcel Gehrke[1][0000−0001−9056−7673]

[1] Institute of Information Systems, University of Lübeck, Germany
[2] German Research Center for Artificial Intelligence (DFKI), Lübeck, Germany
{luttermann,moeller,gehrke}@ifis.uni-luebeck.de

Abstract. Lifting exploits symmetries in probabilistic graphical models by using a representative for indistinguishable objects, allowing to carry out query answering more efficiently while maintaining exact answers. In this paper, we investigate how lifting enables us to perform probabilistic inference for factor graphs containing factors whose potentials are unknown. We introduce the *Lifting Factor Graphs with Some Unknown Factors (LIFAGU) algorithm* to identify symmetric subgraphs in a factor graph containing unknown factors, thereby enabling the transfer of known potentials to unknown potentials to ensure a well-defined semantics and allow for (lifted) probabilistic inference.

1 Introduction

To perform inference in a probabilistic graphical model, all potentials of every factor are required to be known to ensure a well-defined semantics of the model. However, in practice, scenarios arise in which not all factors are known. For example, consider a database of a hospital containing patient data and assume a new patient arrives and we want to include them into an existing probabilistic graphical model such as a factor graph (FG). Clearly, not all attributes of the database are measured for every new patient, i.e., there are some values missing, resulting in an FG with unknown factors and ill-defined semantics when including a new patient in an existing FG. Therefore, we aim to add new patients to an existing group of indistinguishable patients to treat them equally in the FG, thereby allowing for the imputation of missing values under the assumption that there exists such a group for which all values are known. In particular, we study the problem of constructing a lifted representation having well-defined semantics for an FG containing unknown factors—that is, factors whose mappings from input to output are unknown. In probabilistic inference, lifting exploits symmetries in a probabilistic graphical model, allowing to carry out query answering more efficiently while maintaining exact answers [12]. The main idea is to use a representative of indistinguishable individuals for computations. By lifting the probabilistic graphical model, we ensure a well-defined semantics of the model and allow for tractable probabilistic inference with respect to domain sizes.

Previous work on constructing a lifted representation builds on the Weisfeiler-Leman algorithm [15] which incorporates a colour passing procedure to detect symmetries in a graph, e.g. to test for graph isomorphism. To construct a lifted representation for a given FG where all factors are known, the colour passing (CP) algorithm (originally named "CompressFactorGraph") [1,7] is commonly used. Having obtained a lifted representation, algorithms performing lifted inference can be applied. A widely used algorithm for lifted inference is the lifted variable elimination algorithm, first introduced by Poole [13] and afterwards refined by many researchers to reach its current form [3,4,8,11,14]. Another prominent algorithm for lifted inference is the lifted junction tree algorithm [2], which is designed to handle sets of queries instead of single queries.

To encounter the problem of constructing a lifted representation for an FG containing unknown factors, we introduce the Lifting Factor Graphs with Some Unknown Factors (LIFAGU) algorithm, which is a generalisation of the CP algorithm. LIFAGU is able to handle arbitrary FGs, regardless of whether all factors are known or not. By detecting symmetries in an FG containing unknown factors, LIFAGU generates the possibility to transfer the potentials of known factors to unknown factors to eliminate unknown factors from an FG. We show that, under the assumption that for every unknown factor there is at least one known factor having a symmetric surrounding graph structure to it, *all* unknown potentials in an FG can be replaced by known potentials. Thereby, LIFAGU ensures a well-defined semantics of the model and allows for lifted probabilistic inference.

The remaining part of this paper is structured as follows. Section 2 introduces necessary background information and notations. We first briefly recapitulate FGs, afterwards define parameterised factor graphs (PFGs), and then describe the CP algorithm as a foundation for LIFAGU. Afterwards, in Section 3, we introduce LIFAGU as an algorithm to obtain a lifted representation for an FG that possibly contains unknown factors. We present the results of our empirical evaluation in Section 4 before we conclude in Section 5.

2 Preliminaries

In this section, we begin by defining FGs as a propositional representation for a joint probability distribution between random variables (randvars) and then introduce PFGs, which combine probabilistic models and first-order logic. Thereafter, we describe the well-known CP algorithm to lift a propositional model, i.e., to transform an FG into a PFG with equivalent semantics.

2.1 Factor Graphs and Parameterised Factor Graphs

An FG is an undirected graphical model to represent a full joint probability distribution between randvars [9]. In particular, an FG is a bipartite graph that consists of two disjoint sets of nodes (variable nodes and factor nodes) with edges between a variable node R and a factor node f if the factor f depends on R. A factor is a function that maps its arguments to a positive real number

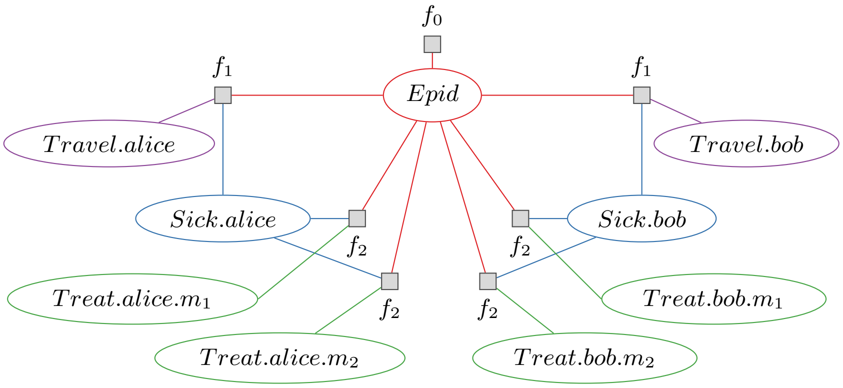

Fig. 1: An FG for an epidemic example [6] with two individuals *alice* and *bob*. The input-output pairs of the factors are omitted for simplification.

(called potential). The semantics of an FG is given by $P(R_1, \dots, R_n) = \frac{1}{Z} \prod_f f$ with Z being the normalisation constant. Figure 1 shows an FG representing an epidemic example with two individuals (*alice* and *bob*) as well as two possible medications (m_1 and m_2) for treatment. For each individual, there are two Boolean randvars $Sick$ and $Travel$, indicating whether the individual is sick and travels, respectively. Moreover, there is another Boolean randvar $Treat$ for each combination of individual and medication, specifying whether the individual is treated with the medication. The Boolean randvar *Epid* states whether an epidemic is present. Although the labelling of the nodes may suggest so, there is no explicit representation of individuals in the graph structure of the propositional FG. The names of the nodes only serve for the reader's understanding.

Clearly, the size of the FG increases with an increasing number of individuals even though it is not necessary to distinguish between individuals because there are symmetries in the model (the factor f_1 occurs two times and the factor f_2 occurs four times). In other words, the probability of an epidemic does not depend on knowing which specific individuals are being sick, but only on how many individuals are being sick. To exploit such symmetries in a model, PFGs can be used. We define PFGs, first introduced by Poole [13], based on the definitions given by Gehrke et al. [5]. PFGs combine first-order logic with probabilistic models, using logical variables (logvars) as parameters in randvars to represent sets of indistinguishable randvars, forming parameterised randvars (PRVs).

Definition 1 (Logvar, PRV, Event). *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. All sets are finite. Each logvar* L *has a domain* $\mathcal{D}(L) \subseteq \mathbf{D}$*. A* constraint *is a tuple* $(\mathcal{X}, C_{\mathcal{X}})$ *of a sequence of logvars* $\mathcal{X} = (X^1, \dots, X^n)$ *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 $R(L_1, \dots, L_n)$*,* $n \geq 0$*, is a syntactical construct of a randvar* $R \in \mathbf{R}$ *possibly combined with logvars* $L_1, \dots, L_n \in \mathbf{L}$ *to represent a set of randvars. If*

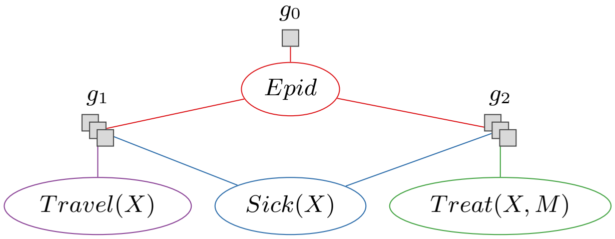

Fig. 2: A PFG corresponding to the lifted representation of the FG depicted in Fig. 1.The input-output pairs of the parfactors are again omitted for brevity.

$n = 0$, *the PRV is parameterless and forms a propositional randvar. A PRV* A *(or logvar* L*) under constraint* C *is given by* $A_{|C}$ $(L_{|C})$*. 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* $a \in \mathcal{R}(A)$ *and we call a set of events* $\mathbf{E} = \{A_1 = a_1, \ldots, A_k = a_k\}$ evidence*.*

As an example, consider $\mathbf{R} = \{Epid, Travel, Sick, Treat\}$ and $\mathbf{L} = \{X, M\}$ with $\mathcal{D}(X) = \{alice, bob\}$ (people), $\mathcal{D}(M) = \{m_1, m_2\}$ (medications), combined into Boolean PRVs $Epid$, $Travel(X)$, $Sick(X)$, and $Treat(X, M)$.

A parametric factor (parfactor) describes a function, mapping argument values to positive real numbers (potentials), of which at least one is non-zero.

Definition 2 (Parfactor, Model, Semantics). *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* $\{g_i\}_{i=1}^{n}$ *forms a* PFG G*. The semantics of* G *is given by grounding and building a full joint distribution. With* Z *as the normalisation constant,* G *represents* $P_G = \frac{1}{Z} \prod_{f \in gr(G)} f$*.*

For example, Fig. 2 shows a PFG $G = \{g_i\}_{i=0}^{2}$ with $g_0 = \phi_0(Epid)_{|\top}$, $g_1 = \phi_1(Travel(X), Sick(X), Epid)_{|\top}$, and $g_2 = \phi_2(Treat(X, M), Sick(X), Epid)_{|\top}$. The PFG illustrated in Fig. 2 is a lifted representation of the FG shown in Fig. 1. Note that the definition of PFGs also includes FGs, as every FG is a PFG containing only parameterless randvars.

2.2 The Colour Passing Algorithm

The CP algorithm [1,7] constructs a lifted representation for an FG where all factors are known. As LIFAGU generalises CP, we briefly recap how the CP algorithm works. The idea is to find symmetries in an FG based on potentials of factors, ranges and evidence of randvars, as well as on the graph structure. Each randvar is assigned a colour depending on its range and evidence, meaning that randvars with identical ranges and identical evidence are assigned the

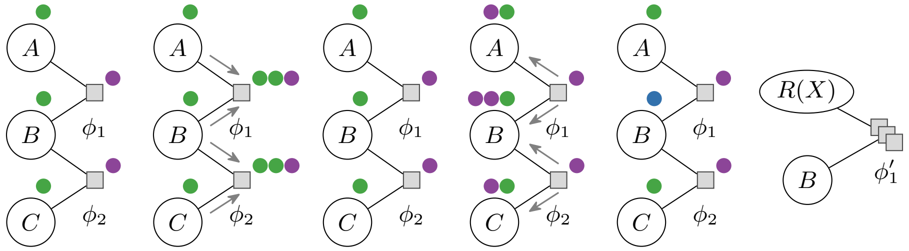

Fig. 3: The colour passing procedure of the CP algorithm on an exemplary input FG containing three Boolean randvars without evidence and two factors with identical potentials. The example has been introduced by Ahmadi et al. [1].

same colour, and each factor is assigned a colour depending on its potentials, i.e., factors with the same potentials get the same colour. The colours are then passed from every randvar to its neighbouring factors and vice versa. Passing colours around is repeated until the groupings of identical colours do not change anymore. In the end, randvars and factors, respectively, are grouped together based on their colour signatures.

Figure 3 depicts the procedure of the CP algorithm on a simple FG. The two factors ϕ_1 and ϕ_2 share identical potentials in this example. As all three randvars are Boolean and there is no evidence available, A, B, and C are assigned the same colour (e.g., green). Furthermore, the potentials of ϕ_1 and ϕ_2 are identical, so they are assigned the same colour (e.g., purple). The colours are then passed from randvars to factors: ϕ_1 receives two times the colour green from A and B and ϕ_2 receives two times the colour green from B and C. Afterwards, ϕ_1 and ϕ_2 are recoloured according to the colours they received from their neighbours. Since both ϕ_1 and ϕ_2 received the same colours, they are assigned the same colour during recolouring (e.g., purple). The colours are then passed from factors to randvars. During this step, not only the colours are shared but also the position of the randvars in the argument list of the corresponding factor. Thus, A receives a tuple $(\text{purple}, 1)$ from ϕ_1, B receives $(\text{purple}, 2)$ from ϕ_1 and $(\text{purple}, 2)$ from ϕ_2, and C receives $(\text{purple}, 1)$ from ϕ_2. Building on these new colour signatures, the randvars are recoloured such that A and C receive the same colour while B is assigned a different colour. Iterating the colour passing procedure does not change these groupings and thus we obtain the PFG shown on the right in Fig. 3.

When facing a situation with unknown factors being present in an FG, the CP algorithm cannot be applied to construct a lifted representation for the FG. In the upcoming section, we introduce the LIFAGU algorithm which generalises the CP algorithm and is able to handle the presence unknown factors.

3 The LIFAGU Algorithm

As our goal is to perform lifted inference, we have to obtain a PFG where all potentials are known. To transform an FG containing unknown factors into a

PFG without unknown factors, we transfer potentials from known factors to unknown factors. For example, consider again the FG depicted in Fig. 1 and assume that another individual, say *eve*, is added to the model. Like *alice* and *bob*, *eve* can travel, be sick, and be treated and hence, four new randvars with three new corresponding factors are attached to the model. However, as we might have limited data, we might not always know the exact potentials for the newly introduced factors when a new individual is added to the model and thus, we end up with a model containing unknown factors. In this example, we can transfer the potentials of the known factors f_1 and f_2 to the newly introduced unknown factors, as it is reasonable to assume that *eve* behaves the same as *alice* and *bob* as long as no evidence suggesting the contrary is available.

In an FG containing unknown factors, the only information available to measure the similarity of factors is the neighbouring graph structure of the factors. For the upcoming definitions, let $\mathrm{Ne}_G(v)$ denote the set of neighbours of a node v (variable node or factor node) in G, i.e., $\mathrm{Ne}_G(f)$ contains all randvars connected to a factor f in G and $\mathrm{Ne}_G(R)$ contains all factors connected to a randvar R in G. If the context is clear, we omit the subscript from $\mathrm{Ne}_G(v)$ and write $\mathrm{Ne}(v)$ for simplification. We start by defining the 2-step neighbourhood of a factor f as the set containing all randvars that are connected to f as well as all factors connected to a randvar that is connected to f. The concept of taking into account all nodes with a maximal distance of two is based on the idea of a single iteration of the colour passing procedure.

Definition 3 (2-Step Neighbourhood). *The* 2-step neighbourhood *of a factor f in an FG G is defined as*

$$2\text{-}step_G(f) = \{R \mid R \in \mathrm{Ne}_G(f)\} \cup \{f' \mid \exists R : R \in \mathrm{Ne}_G(f) \wedge f' \in \mathrm{Ne}_G(R)\}.$$

If the context is clear, we write $2\text{-}step(f)$ instead of $2\text{-}step_G(f)$. For example, the 2-step neighbourhood of ϕ_1 in the FG depicted in Fig. 3 is given by $2\text{-}step(\phi_1) = \{A, B\} \cup \{\phi_1, \phi_2\}$. By $G[V']$ we denote the subgraph of a graph G induced by a subset of nodes V', that is, $G[V']$ contains only the nodes in V' as well as all edges from G that connect two nodes in V'. In our example, $G[2\text{-}step(\phi_1)]$ then consists of the nodes A, B, ϕ_1, and ϕ_2, and contains the edges $A - \phi_1$, $B - \phi_1$, and $B - \phi_2$. As it is currently unknown whether a general graph isomorphism test is solvable in polynomial time, we make use of the notion of symmetric 2-step neighbourhoods instead of relying on isomorphic 2-step neighbourhoods to ensure that LIFAGU is implementable in polynomial time.

Definition 4 (Symmetric 2-Step Neighbourhoods). *Given an FG G and factors f_i, f_j in G, $G[2\text{-}step_G(f_i)]$ is* symmetric *to $G[2\text{-}step_G(f_j)]$ if*

1. $|\mathrm{Ne}_G(f_i)| = |\mathrm{Ne}_G(f_j)|$ *and*
2. *there exists a bijection $\phi : \mathrm{Ne}_G(f_i) \to \mathrm{Ne}_G(f_j)$ that maps every randvar $R_k \in \mathrm{Ne}_G(f_i)$ to a randvar $R_\ell \in \mathrm{Ne}_G(f_j)$ such that the evidence for R_k and R_ℓ is identical, $\mathcal{R}(R_k) = \mathcal{R}(R_\ell)$, and $|\mathrm{Ne}_G(R_k)| = |\mathrm{Ne}_G(R_\ell)|$.*

Algorithm 1: LIFAGU

```
Input  : An FG $G$ with randvars $\mathbf{R} = \{R_1, \dots, R_n\}$, known factors
         $\mathbf{F} = \{f_1, \dots, f_m\}$, unknown factors $\mathbf{F}' = \{f'_1, \dots, f'_z\}$, and evidence
         $\mathbf{E} = \{R_1 = r_1, \dots, R_k = r_k\}$, and a real-valued threshold $\theta \in [0, 1]$.
Output: A lifted representation $G'$ of $G$.
Assign each $f_i \in \mathbf{F}$ a colour based on its potentials;
Assign each $f'_i \in \mathbf{F}'$ a unique colour;
foreach unknown factor $f_i \in \mathbf{F}'$ do
    $C_{f_i} \leftarrow \{\}$;
    foreach factor $f_j \in \mathbf{F} \cup \mathbf{F}'$ with $f_i \neq f_j$ do
        if $f_i \approx f_j$ then
            if $f_j$ is unknown then
                Assign $f_j$ the same colour as $f_i$;
            else
                $C_{f_i} \leftarrow C_{f_i} \cup \{f_j\}$;
foreach set of candidates $C_{f_i}$ do
    $C^{\ell}_{f_i} \leftarrow$ Maximal subset of $C_{f_i}$ such that $f_j \approx f_k$ holds for all $f_j, f_k \in C^{\ell}_{f_i}$;
    if $|C^{\ell}_{f_i}| \,/\, |C_{f_i}| \geq \theta$ then
        Assign all $f_j \in C^{\ell}_{f_i}$ the same colour as $f_i$;
$G \leftarrow$ Result from calling the CP algorithm on the modified graph $G$ and $\mathbf{E}$;
```

For example, take a look again at the FG shown in Fig. 3 and assume that there is no evidence. We can check whether ϕ_1 and ϕ_2 have symmetric 2-step neighbourhoods: Both ϕ_1 and ϕ_2 are connected to two randvars as $\mathrm{Ne}(\phi_1) = \{A, B\}$ and $\mathrm{Ne}(\phi_2) = \{B, C\}$, thereby satisfying the first condition. Further, A can be mapped to C with $\mathcal{R}(A) = \mathcal{R}(C)$ (Boolean) and $|\mathrm{Ne}(A)| = |\mathrm{Ne}(C)| = 1$ and B can be mapped to itself. Thus, condition two is satisfied and it holds that $G[2\text{-}step(\phi_1)]$ is symmetric to $G[2\text{-}step(\phi_2)]$. Having defined the notion of symmetric 2-step neighbourhoods, we are able to specify a condition for two factors to be possibly identical. Two factors are considered possibly identical if the subgraphs induced by their 2-step neighbourhoods are symmetric.

Definition 5 (Possibly Identical Factors). *Given two factors f_i and f_j in an FG G, we call f_i and f_j* possibly identical*, denoted as $f_i \approx f_j$, if*

1. *$G[2\text{-}step_G(f_i)]$ is symmetric to $G[2\text{-}step_G(f_j)]$ and*
2. *at least one of f_i and f_j is unknown, or f_i and f_j have the same potentials.*

The second condition serves to ensure consistency as two factors with different potentials can obviously not be identical. Applying the definition of possibly identical factors to ϕ_1 and ϕ_2 from Fig. 3, we can verify that ϕ_1 and ϕ_2 are indeed possibly identical because they have symmetric 2-step neighbourhoods and identical potentials. Next, we describe the entire LIFAGU algorithm, which is illustrated in Algorithm 1.

LIFAGU assigns colours to unknown factors based on symmetric subgraphs induced by their 2-step neighbourhoods, proceeding as follows for an input G. As an initialisation step, LIFAGU assigns each known factor a colour based on its potentials and each unknown factor a unique colour. Then, LIFAGU searches for possibly identical factors in two phases. In the first phase, all unknown factors that are possibly identical are assigned the same colour, as there is no way to distinguish them. Furthermore, LIFAGU collects for every unknown factor f_i a set C_{f_i} of known factors possibly identical to f_i. The second phase then continues to group the unknown factors with known factors, including the transfer of the potentials from the known factors to the unknown factors. For every unknown factor f_i, LIFAGU computes a maximal subset $C^{\ell}_{f_i} \subseteq C_{f_i}$ for which all elements are pairwise possibly identical. Afterwards, f_i and all $f_j \in C^{\ell}_{f_i}$ are assigned the same colour if a user-defined threshold is reached. Finally, CP is called on G, which now includes the previously set colours for the unknown factors in G, to group both known and unknown factors in G.

The purpose of the threshold θ is to control the required agreement of known factors before grouping unknown factors with known factors as it is possible for an unknown factor to be possibly identical to multiple known factors having different potentials. A larger θ requires a higher agreement, e.g., $\theta = 1$ requires all candidates to have identical potentials. Note that all known factors in $C^{\ell}_{f_i}$ are guaranteed to have identical potentials (otherwise they would not be pairwise possibly identical) and thus, their potentials can be transferred to f_i. Consequently, the output of LIFAGU is guaranteed to contain only known factors and hence ensures a well-defined semantics if $C^{\ell}_{f_i}$ is non-empty for each unknown factor f_i and the threshold is sufficiently small (e.g., zero) to group each unknown factor with at least one known factor.

Corollary 1. *Given that for every unknown factor f_i there is at least one known factor that is possibly identical to f_i in an FG G, LIFAGU is able to replace all unknown potentials in G by known potentials.*

It is easy to see that LIFAGU is a generalisation of CP, meaning that both algorithms compute the same result for input FGs containing only known factors (if an input FG G contains no unknown factors, only the first line and the last line of Algorithm 1 are executed—which is equivalent to calling CP on G).

Corollary 2. *Given an FG that contains only known factors, CP and LIFAGU output identical groupings of randvars and factors, respectively.*

Next, we investigate the practical performance of LIFAGU in our evaluation.

4 Empirical Evaluation

In this section, we present the results of the empirical evaluation for LIFAGU. To evaluate the performance of LIFAGU, we start with a non-parameterised FG G where all factors are known, serving as our ground truth. Afterwards, we remove the potential mappings for five to ten percent of the factors in G, yielding an incomplete FG G' on which LIFAGU is run to obtain a PFG G_{LIFAGU}. Each factor f' whose potentials are removed is chosen randomly under the constraint

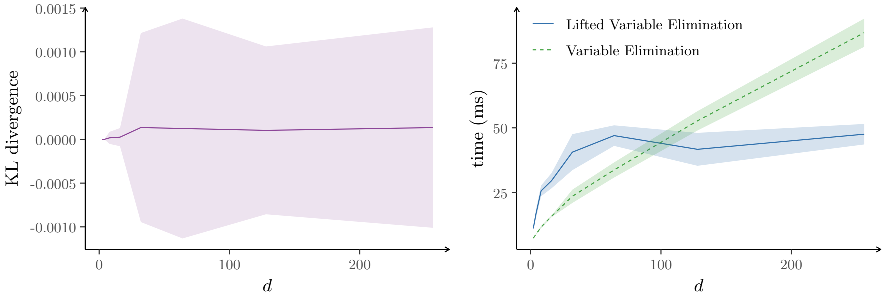

Fig. 4: Left: The mean KL divergence on the queried probability distributions (thick line) as well as the standard deviation of all measured KL divergences for each choice of d (ribbon around the mean). Right: The mean run time of variable elimination and lifted variable elimination for each choice of d.

that there exists at least one other factor with known potentials that is possibly identical to f'. This constraint corresponds to the assumption that there exists at least one group to which a new individual can be added and it ensures that after running LIFAGU, probabilistic inference can be performed for evaluation purposes. Clearly, in our evaluation setting, there is not only a single new individual but instead a set of new individuals, given by the set of factors whose potentials are missing. We use a parameter $d = 2, 4, 8, 16, 32, 64, 128, 256$ to control the size of the FG G (and thus, the size of G'). More precisely, for each choice of d, we evaluate multiple input FGs which contain between $2d$ and $3d$ randvars (and factors, respectively). The potentials of the factors are randomly generated such that the ground truth G contains between three and five (randomly chosen) cohorts of randvars which should be grouped together, with one cohort containing roughly 50 percent of all randvars in G while the other cohorts share the remaining 50 percent of the randvars from G uniformly at random.

We set $\theta = 0$ to ensure that each unknown factor is grouped with at least one known factor to be able to perform lifted probabilistic inference on G_{LIFAGU} for evaluation. To assess the error made by LIFAGU for each choice of d, we pose d different queries to the ground truth G and to G_{LIFAGU}, respectively. For each query, we compute the Kullback-Leibler (KL) divergence [10] between the resulting probability distributions for the ground truth G and G_{LIFAGU} to measure the similarity of the query results. The KL divergence measures the difference of two distributions and its value is zero if the distributions are identical.

In the left plot of Fig. 4, we report the mean KL divergence over all queries for each choice of d. The ribbon around the line illustrates the standard deviation of the measured KL divergences. We find that the mean KL divergence is close to zero for all choices of d in practice. Both the mean KL divergence and the standard deviation of the KL divergences do not show any significant differences between the various values for d. Note that the depicted standard deviation is also very small for all choices of d due to the granularity of the y-axis. The maximum KL divergence measured for any choice of d is about 0.01.

Given our assumptions, a new individual actually belongs to a cohort and most cohorts behave not completely different. So normally, we trade off accuracy of query results for the ability to perform inference, which otherwise would not be possible at all. If the semantics cannot be fixed, missing potentials need to be guessed to be able to perform inference at all, probably resulting in worse errors. As we basically perform unsupervised clustering, errors might happen when grouping unknown factors with known factors. The error might be further reduced by increasing the effort when searching for known factors that are possible candidates for grouping with an unknown factor. For example, it is conceivable to increase the size of the neighbourhood during the search for possible identical factors at the expense of a higher run time expenditure.

In addition to the error measured by the KL divergence, we also report the run times of variable elimination on G and lifted variable elimination on the PFG computed by LIFAGU, i.e., G_{LIFAGU}. The run times are shown in the right plot of Fig. 4. As expected, lifted variable elimination is faster than variable elimination for larger graphs and the run time of lifted variable elimination increases more slowly with increasing graph sizes than the run time of variable elimination. Hence, LIFAGU not only allows to perform probabilistic inference at all, but also speeds up inference by allowing for lifting probabilistic inference. Note that there are on average 24 different groups over all settings with the largest domain size being 87 (for the setting of $d = 256$), i.e., there are a lot of small groups (of size one) which diminish the advantage of lifted variable elimination over variable elimination. We could also obtain more compact PFGs by merging groups that are not fully identical but similar to a given extent such that the resulting PFG contains less different groups at the cost of a lower accuracy of query results. Obtaining a more compact PFG would most likely result in a higher speedup of lifted variable elimination compared to variable elimination.

5 Conclusion

In this paper, we introduce the LIFAGU algorithm to construct a lifted representation for an FG that possibly contains unknown factors. LIFAGU is a generalisation of the widespread CP algorithm and allows to transfer potentials from known factors to unknown factors by identifying symmetric subgraphs. Under the assumption that for every unknown factor there exists at least one known factor having a symmetric surrounding graph structure to it, LIFAGU is able to replace all unknown potentials in an FG by known potentials.

Acknowledgements

This work was partially supported by the BMBF project AnoMed. The research of Malte Luttermann was also partially supported by the Medical Cause and Effects Analysis (MCEA) project. This preprint has not undergone peer review or any post-submission improvements or corrections. The Version of Record of this contribution is published in *Lecture Notes in Computer Science, Volume 14294*, and is available online at https://doi.org/10.1007/978-3-031-45608-4_25.

References

1. Ahmadi, B., Kersting, K., Mladenov, M., Natarajan, S.: Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training. Machine Learning **92**, 91–132 (2013)
2. Braun, T., Möller, R.: Lifted Junction Tree Algorithm. In: Proceedings of KI 2016: Advances in Artificial Intelligence (KI-16). pp. 30–42. Springer (2016)
3. De Salvo Braz, R., Amir, E., Roth, D.: Lifted First-Order Probabilistic Inference. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05). pp. 1319–1325. Morgan Kaufmann Publishers Inc. (2005)
4. De Salvo Braz, R., Amir, E., Roth, D.: MPE and Partial Inversion in Lifted Probabilistic Variable Elimination. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06). pp. 1123–1130. AAAI Press (2006)
5. Gehrke, M., Möller, R., Braun, T.: Taming Reasoning in Temporal Probabilistic Relational Models. In: Proceedings of the Twenty-Fourth European Conference on Artificial Intelligence (ECAI-20). pp. 2592–2599. IOS Press (2020)
6. Hoffmann, M., Braun, T., Möller: Lifted Division for Lifted Hugin Belief Propagation. In: Proceedings of the Twenty-Fifth International Conference on Artificial Intelligence and Statistics (AISTATS-22). pp. 6501–6510. PMLR (2022)
7. Kersting, K., Ahmadi, B., Natarajan, S.: Counting Belief Propagation. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI-09). pp. 277–284. AUAI Press (2009)
8. Kisyński, J., Poole, D.: Constraint Processing in Lifted Probabilistic Inference. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI-09). pp. 293–302. AUAI Press (2009)
9. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor Graphs and the Sum-Product Algorithm. IEEE Transactions on Information Theory **47**, 498–519 (2001)
10. Kullback, S., Leibler, R.A.: On Information and Sufficiency. The Annals of Mathematical Statistics **22**, 79–86 (1951)
11. Milch, B., Zettlemoyer, L.S., Kersting, K., Haimes, M., Kaelbling, L.P.: Lifted Probabilistic Inference with Counting Formulas. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08). pp. 1062–1068. AAAI Press (2008)
12. Niepert, M., Van den Broeck, G.: Tractability through Exchangeability: A New Perspective on Efficient Probabilistic Inference. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14). pp. 2467–2475. AAAI Press (2014)
13. Poole, D.: First-Order Probabilistic Inference. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03). pp. 985–991. Morgan Kaufmann Publishers Inc. (2003)
14. Taghipour, N., Fierens, D., Davis, J., Blockeel, H.: Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. Journal of Artificial Intelligence Research **47**, 393–439 (2013)
15. Weisfeiler, B., Leman, A.A.: The Reduction of a Graph to Canonical Form and the Algebra which Appears Therein. NTI, Series **2**, 12–16 (1968), English

translation by Grigory Ryabov available at https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf