Lifting Factor Graphs with Some Unknown Factors for New Individuals[☆]

Malte Luttermann[a,*], Ralf Möller[b], Marcel Gehrke[b]

[a] *German Research Center for Artificial Intelligence (DFKI), Ratzeburger Allee 160, 23562, Lübeck, Germany*
[b] *Institute for Humanities-Centered Artificial Intelligence, University of Hamburg, Warburgstraße 28, 20354, Hamburg, Germany*

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 unknown factors, i.e., factors whose underlying function of potential mappings is unknown. We present the *Lifting Factor Graphs with Some Unknown Factors (LIFAGU) algorithm* to identify indistinguishable subgraphs in a factor graph containing unknown factors, thereby enabling the transfer of known potentials to unknown potentials to ensure a well-defined semantics of the model and allow for (lifted) probabilistic inference. We further extend LIFAGU to incorporate additional background knowledge about groups of factors belonging to the same individual object. By incorporating such background knowledge, LIFAGU is able to further reduce the ambiguity of possible transfers of known potentials to unknown potentials.

Keywords: probabilistic graphical models, factor graphs, lifted inference

[☆]This paper is a revised and extended version of a paper (Luttermann et al., 2023) that has been published at the Seventeenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2023).

[*]Corresponding author

Email addresses: malte.luttermann@dfki.de (Malte Luttermann), ralf.moeller@uni-hamburg.de (Ralf Möller), marcel.gehrke@uni-hamburg.de (Marcel Gehrke)

1. Introduction

To perform inference in a probabilistic graphical model, all potential mappings 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 included in 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. More specifically, it is conceivable that in a first examination of the new patient, a measurement of their blood pressure is conducted whereas measurements for other attributes are not immediately performed. 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 underlying function mappings from input to output are unknown. In probabilistic inference, lifting exploits symmetries in a probabilistic graphical model, thereby allowing to carry out query answering more efficiently while maintaining exact answers (Niepert and Van den Broeck, 2014). The main idea behind lifted inference 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 at the same time allow for tractable probabilistic inference (e.g., inference requiring polynomial time) with respect to domain sizes.

Previous work to construct a lifted representation builds on the Weisfeiler-Leman algorithm (Weisfeiler and Leman, 1968) which incorporates a colour passing procedure to detect symmetries in a graph, e.g. to test for graph isomorphism. To construct a lifted representation, denoted as a parameterised factor graph (PFG), for a given FG in which all factors are known, the advanced colour passing (ACP) algorithm (Luttermann et al., 2024a,c,d) is the state of the art. The ACP algorithm builds on the colour passing algorithm (originally named "CompressFactorGraph") (Kersting et al., 2009; Ahmadi et al., 2013), which itself is based on work by Singla and Domingos (2008). ACP detects symmetries in an FG to obtain possible groups of random vari-

ables (randvars) and factors by deploying a colour passing procedure similar to the Weisfeiler-Leman algorithm. Having obtained a lifted representation, algorithms for lifted inference can be applied. A widely used algorithm for lifted inference is the lifted variable elimination algorithm, first introduced by Poole (2003) and afterwards refined by many researchers to reach its current form (De Salvo Braz et al., 2005, 2006; Milch et al., 2008; Kisyński and Poole, 2009; Taghipour et al., 2013; Braun and Möller, 2018). Another prominent algorithm for lifted inference is the lifted junction tree algorithm (Braun and Möller, 2016), which is designed to handle sets of queries instead of single queries. More recently, causal knowledge has also been incorporated into PFGs to allow for lifted causal inference (Luttermann et al., 2024b).

To encounter the problem of constructing a PFG as 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 ACP 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 such that they have an indistinguishable surrounding graph structure, *all* unknown potential mappings in an FG can be replaced by known potential mappings. Thereby, LIFAGU ensures a well-defined semantics of the model and allows for lifted probabilistic inference. We further extend LIFAGU to incorporate background knowledge about multiple factors belonging to the same individual object—that is, if we know that a set of factors belongs to the same individual object, LIFAGU might be able to exploit this knowledge to reduce the ambiguity for possible transfers of known potential mappings.

The remaining part of this paper is structured as follows. Section 2 introduces necessary background information and notations. We first recapitulate FGs, afterwards define PFGs as first-order probabilistic models, and then describe the ACP algorithm as a foundation for LIFAGU. Afterwards, in Section 3, we introduce LIFAGU as a generalisation of ACP allowing us to obtain a lifted representation (a PFG) for an FG that possibly contains unknown factors. In Section 4, we extend LIFAGU to incorporate background knowledge. We then present the results of our empirical evaluation in Section 5 before we conclude in Section 6.

2. Preliminaries

In this section, we begin by defining FGs as propositional representations for a joint probability distribution between randvars and then introduce PFGs, which combine probabilistic models and first-order logic. Thereafter, we describe the ACP 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 compactly encode a full joint probability distribution over a set of randvars by representing the distribution as a product of factors (Frey et al., 1997; Kschischang et al., 2001).

Definition 2.1 (Factor Graph, Kschischang et al., 2001)**.** *An* FG $G = (\boldsymbol{V}, \boldsymbol{E})$ *is an undirected bipartite graph consisting of a node set* $\boldsymbol{V} = \boldsymbol{R} \cup \boldsymbol{F}$*, where* $\boldsymbol{R} = \{R_1, \ldots, R_n\}$ *is a set of randvars (also referred to as variable nodes) and* $\boldsymbol{F} = \{f_1, \ldots, f_m\}$ *is a set of factor nodes, as well as a set of edges* $\boldsymbol{E} \subseteq \boldsymbol{R} \times \boldsymbol{F}$*. Every factor node* $f_j \in \boldsymbol{F}$ *defines a function* $\phi_j(\mathcal{R}_j)$*, where* $\phi_j \colon \times_{R \in \mathcal{R}_j} \mathrm{range}(R) \mapsto \mathbb{R}^+$ *maps a sequence* $\mathcal{R}_j$ *of randvars from* $\boldsymbol{R}$ *to a positive real number (called potential). The term* $\mathrm{range}(R_i)$ *denotes the possible values of a randvar* R_i*. There is an edge between a variable node* R_i *and a factor node* $f_j = \phi_j(\mathcal{R}_j)$ *in* $\boldsymbol{E}$ *if* R_i *appears in the argument list of* ϕ_j*. The semantics of the FG* G *is given by*

$$P_G = \frac{1}{Z} \prod_{j=1}^{m} \phi_j(\mathcal{R}_j) \tag{1}$$

with Z *being the normalisation constant and* $\mathcal{R}_j$ *denoting the randvars appearing in the argument list of* ϕ_j*.*

Example 2.1. *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. 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. Every factor* f_j *defines a function, e.g.,* $f_0 = \phi_0(Epid)$ *defines two potential*

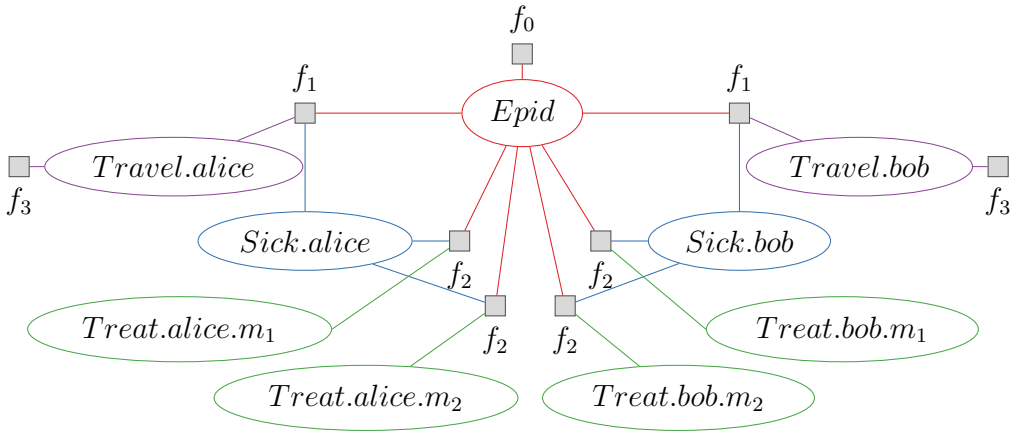


Figure 1: An FG for an epidemic example (Hoffmann et al., 2022) with two individuals *alice* and *bob* as well as two medications m_1 and m_2 for treatment. For simplicity, all randvars are Boolean and the input-output pairs of the factors are omitted.

mappings $\phi_0(Epid = true)$ *and* $\phi_0(Epid = false)$*, which are both mapped to a positive real number, respectively. We omit the exact specification of the potential mappings from arguments to positive real numbers for brevity.*

Note that even though the labelling of the nodes in Fig. 1 may suggest so, there is no explicit representation of individuals in the graph structure of the propositional FG. The labels of the nodes only serve for the reader's understanding and in general, the node labels can be arbitrary strings of characters. For example, consider the label $Treat.alice.m_1$. We deliberately avoid using a notation with parameters such as $Treat(alice, m_1)$ to emphasise that the label does *not* explicitly encode that there is an individual *alice* and a medication m_1. Instead, the label is an arbitrary string of characters (and is only chosen as $Treat.alice.m_1$ for better readability).

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 factors f_1 and f_3 occur 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. In the following, we define PFGs, first introduced by Poole (2003), based on the definitions given by Gehrke et al. (2020). PFGs combine first-order logic with probabilistic models, us-

ing logical variables (logvars) as parameters in randvars to represent sets of indistinguishable randvars, forming parameterised randvars (PRVs).

Definition 2.2 (Parameterised Random Variable)**.** *Let $\boldsymbol{R}$ be a set of randvar names, $\boldsymbol{L}$ a set of logvar names, $\boldsymbol{\Phi}$ a set of factor names, and $\boldsymbol{D}$ a set of constants. All sets are finite. Each logvar L has a domain* $\mathrm{dom}(L) \subseteq \boldsymbol{D}$*. A* constraint *is a tuple $(\mathcal{X}, \boldsymbol{C}_{\mathcal{X}})$ of a sequence of logvars $\mathcal{X} = (X_1, \ldots, X_n)$ and a set $\boldsymbol{C}_{\mathcal{X}} \subseteq \times_{i=1}^{n} \mathrm{dom}(X_i)$. The symbol $\top$ for C marks that no restrictions apply, i.e., $\boldsymbol{C}_{\mathcal{X}} = \times_{i=1}^{n} \mathrm{dom}(X_i)$. A* PRV *$R(L_1, \ldots, L_n)$, $n \geq 0$, is a syntactical construct of a randvar $R \in \boldsymbol{R}$ possibly combined with logvars $L_1, \ldots, L_n \in \boldsymbol{L}$ to represent a set of randvars. If $n = 0$, the PRV is parameterless and forms a propositional randvar. A PRV A (or logvar L) under constraint C is given by $A_{|C}$ ($L_{|C}$, respectively). We may omit $|\top$ in $A_{|\top}$ or $L_{|\top}$. The term* $\mathrm{range}(A)$ *denotes the possible values of a PRV A. An* event *$A = a$ denotes the occurrence of PRV A with range value $a \in \mathrm{range}(A)$ and we call a set of events $\boldsymbol{E} = \{A_1 = a_1, \ldots, A_k = a_k\}$* evidence*.*

Example 2.2. *Consider $\boldsymbol{R} = \{Epid, Travel, Sick, Treat\}$ and $\boldsymbol{L} = \{X, M\}$ with $\mathrm{dom}(X) = \{alice, bob\}$ (people), $\mathrm{dom}(M) = \{m_1, m_2\}$ (medications), combined into Boolean PRVs Epid, $Travel(X)$, $Sick(X)$, and $Treat(X, M)$.*

In the previous example, there are two indistinguishable individuals *alice* and *bob* as well as two indistinguishable medications m_1 and m_2. Note that in general, there might be multiple groups of indistinguishable individuals, e.g., in addition to *alice* and *bob* there might be another group of individuals *dave* and *eve* such that *alice* and *bob* as well as *dave* and *eve* are indistinguishable, respectively. To represent multiple groups of indistinguishable objects, constraints are used. For example, instead of having a single PRV $Sick(X)$, we might have two PRVs $Sick(X)_{|C_1}$ and $Sick(X)_{|C_2}$ with constraints $C_1 = (X, \{alice, bob\})$ and $C_2 = (X, \{dave, eve\})$, respectively, to represent that *alice* and *bob* as well as *dave* and *eve* are indistinguishable with respect to being sick. Analogously, we might have constraints for the PRVs $Travel(X)$ and $Treat(X, M)$, allowing us to represent different combinations of groups of indistinguishable objects.

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

Definition 2.3 (Parfactor)**.** *Let $\boldsymbol{\Phi}$ denote a set of factor names. We denote a* parfactor *g by $\phi(\mathcal{A})_{|C}$ with $\mathcal{A} = (A_1, \ldots, A_n)$ being a sequence of*

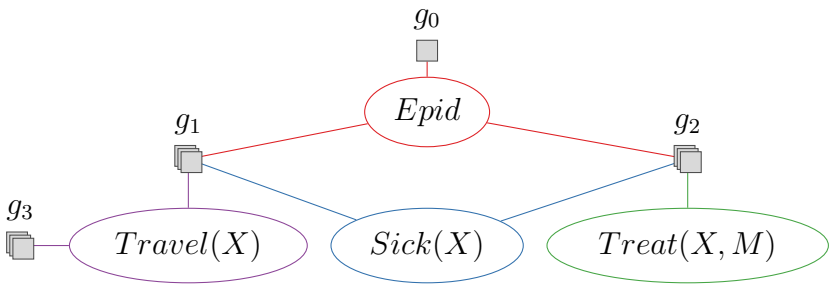


Figure 2: A PFG corresponding to the lifted representation of the FG depicted in Fig. 1. The input-output pairs of the parfactors are again omitted for brevity.

PRVs, $\phi\colon \times_{i=1}^{n} \mathrm{range}(A_i) \mapsto \mathbb{R}^+$ *being a function with name* $\phi \in \mathbf{\Phi}$ *mapping argument values to a positive real number called* potential*, and* C *being 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 (groundings) of* Y *with respect to constraint* C*.*

Example 2.3. *Take a look at* $g_1 = \phi_1(Epid, Travel(X), Sick(X))_{|\top}$ *and let* $\mathrm{dom}(X) = \{alice, bob\}$*. If all PRVs are Boolean,* g_1 *specifies* $2 \cdot 2 \cdot 2 = 8$ *input-output pairs* $\phi_1(true, true, true) = \varphi_1$*,* $\phi_1(true, true, false) = \varphi_2$*, and so on with* $\varphi_i \in \mathbb{R}^+$*. Moreover, it holds that* $lv(g_1) = \{X\}$ *and* $gr(g_1) = \{\phi_1(Epid, Travel(alice), Sick(alice)), \phi_1(Epid, Travel(bob), Sick(bob))\}$*. Thus, in this specific example,* g_1 *represents a set of two ground factors.*

A set of parfactors $\{g_j\}_{j=1}^{m}$ then forms a PFG G.

Definition 2.4 (Parametric Factor Graph)**.** *A* PFG $G = (\boldsymbol{V}, \boldsymbol{E})$ *is a bipartite graph with node set* $\boldsymbol{V} = \boldsymbol{A} \cup \boldsymbol{G}$ *where* $\boldsymbol{A} = \{A_1, \ldots, A_n\}$ *is a set of PRVs and* $\boldsymbol{G} = \{g_1, \ldots, g_m\}$ *is a set of parfactors. A PRV* A_i *and a parfactor* g_j *are connected via an edge in* G *(i.e.,* $\{A_i, g_j\} \in \boldsymbol{E}$*) if* A_i *appears in the argument list of* $g_j = \phi_j(\mathcal{A}_j)$*. The semantics of* G *is given by grounding and building a full joint distribution. With* Z *as the normalisation constant and* $\mathcal{A}_k$ *denoting the PRVs occurring in the argument list of* ϕ_k*,* G *represents*

$$P_G = \frac{1}{Z} \prod_{g_j \in \boldsymbol{G}} \prod_{\phi_k \in gr(g_j)} \phi_k(\mathcal{A}_k). \tag{2}$$

Example 2.4. *Figure 2 shows a PFG G consisting of the PRVs Epid, $Travel(X)$, $Sick(X)$, and $Treat(X, M)$ as well as of the four parfactors $\{g_i\}_{i=0}^{3}$ where $g_0 = \phi_0(Epid)_{|\top}$, $g_1 = \phi_1(Epid, Travel(X), Sick(X))_{|\top}$, $g_2 = \phi_2(Treat(X, M), Sick(X), Epid)_{|\top}$, and $g_3 = \phi_3(Travel(X), Sick(X))_{|\top}$. G is a lifted representation of the FG shown in Fig. 1 and the number of PRVs and parfactors in G remains constant even if the number of individuals and medications in the model increases.*

We remark that the definition of PFGs also includes FGs, as every FG is a PFG containing only parameterless randvars. In Definition 2.1, we assume that all functions encoded by the factors are known. As the semantics of an FG is given by a product over its factors, the input-output mappings of the factors must be known to ensure a well-defined semantics of the model. However, in practice, the underlying function specifications of factors might be unknown, leading to the presence of unknown factors in an FG. The upcoming definition formalises the notion of an unknown factor.

Definition 2.5 (Unknown Factor)**.** *Let $G = (\boldsymbol{V}, \boldsymbol{E})$ denote an FG with node set $\boldsymbol{V} = \boldsymbol{R} \cup \boldsymbol{F}$, where $\boldsymbol{R} = \{R_1, \ldots, R_n\}$ is a set of randvars and $\boldsymbol{F} = \{f_1, \ldots, f_m\}$ is a set of factors. An* unknown factor *f_j is a factor whose function specification $\phi_j(\mathcal{R}_j)$ is unknown, i.e., the arguments $\mathcal{R}_j$ of ϕ_j are known but the potential values to which ϕ_j maps its arguments are unknown.*

Before we deal with unknown factors, we first introduce the ACP algorithm, which constructs a PFG from a given FG where all factors are known. Thereafter, we generalise ACP to handle the presence unknown factors.

2.2. The Advanced Colour Passing Algorithm

The ACP algorithm (Luttermann et al., 2024a) constructs a lifted representation for an FG in which all factors are known. As LIFAGU generalises ACP, we briefly recap how the ACP 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 variable node (randvar) is assigned a colour depending on its range and observed event, meaning that randvars with identical ranges and identical observed events are assigned the same colour, and each factor is assigned a colour depending on its potentials, i.e., factors encoding functions with the same potential mappings receive the same colour. The colours are first passed from every variable node to its

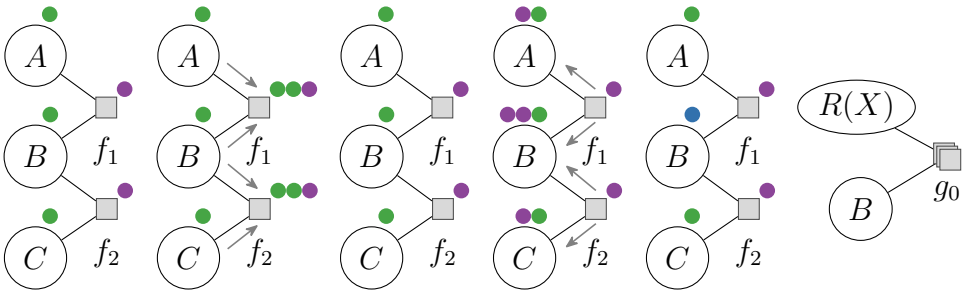


Figure 3: The colour passing procedure of the ACP algorithm on an exemplary input FG (Ahmadi et al., 2013) containing three Boolean randvars without evidence and two factors encoding functions with identical potential mappings.

neighbouring factor nodes and each factor f collects all colours of neighbouring randvars in the order of their appearance in the argument list of f's underlying function. Based on the collected colours and their own colour, factors are grouped together and reassigned a new colour (to reduce communication overhead). Then, colours are passed from factor nodes to variable nodes. Based on the collected colours and their own colour, randvars are grouped together and reassigned a new colour. The colour passing procedure is iterated until groupings do not change anymore and in the end, all randvars and factors are grouped together based on their colour signatures (that is, the messages they received from their neighbours plus their own colour).

Example 2.5. *Figure 3 depicts the procedure of the ACP algorithm on a simple FG. The two factors f_1 and f_2 encode functions with identical potential mappings 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 potential mappings of f_1 and f_2 are identical, so they are assigned the same colour (e.g., purple). The colours are then passed from randvars to factors: f_1 receives two times the colour green from A and B and f_2 receives two times the colour green from B and C. Afterwards, f_1 and f_2 are recoloured according to the colours they received from their neighbours. Since both f_1 and f_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. Here, A receives the colour purple from f_1, B receives the colour purple from f_1 and the colour purple from f_2, and C receives the colour purple from f_2. Building on these new colour signatures, the randvars are recoloured such that A and C receive the same*

colour whereas B is assigned a different colour. In this particular example, further iterating the colour passing procedure does not change these groupings. Finally, ACP introduces logvars to obtain PRVs that represent groups of randvars with identical colour signatures. Further, ACP replaces groups of factors having identical colours by parfactors. Here, A and C are represented by a PRV $R(X)$ having a single logvar X with $\mathrm{dom}(X) = \{A, C\}$ (as shown on the right in Fig. 3). Note that the name R is chosen arbitrarily and in general, it is also possible to have multiple logvars within a PRV.

For more technical details on ACP, we refer the reader to Luttermann et al. (2024a). The authors also provide detailed instructions on how the logvars are introduced to construct the resulting PFG from the colourings.

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

3. The LIFAGU Algorithm

The semantics of an FG (or a PFG) relies on a multiplication of all factors in the model and thus, all factors must be known to ensure a well-defined semantics of the model. 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.

We illustrate the idea of transferring potentials using the exemplary FG depicted in Fig. 4. In this example, another individual *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 do not always know the exact potential mappings 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 the example from Fig. 4, we have three unknown factors, denoted as $f_?$. We can transfer the potentials of the known factors f_1, f_2, and f_3 to the newly introduced unknown factors $f_?$, 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

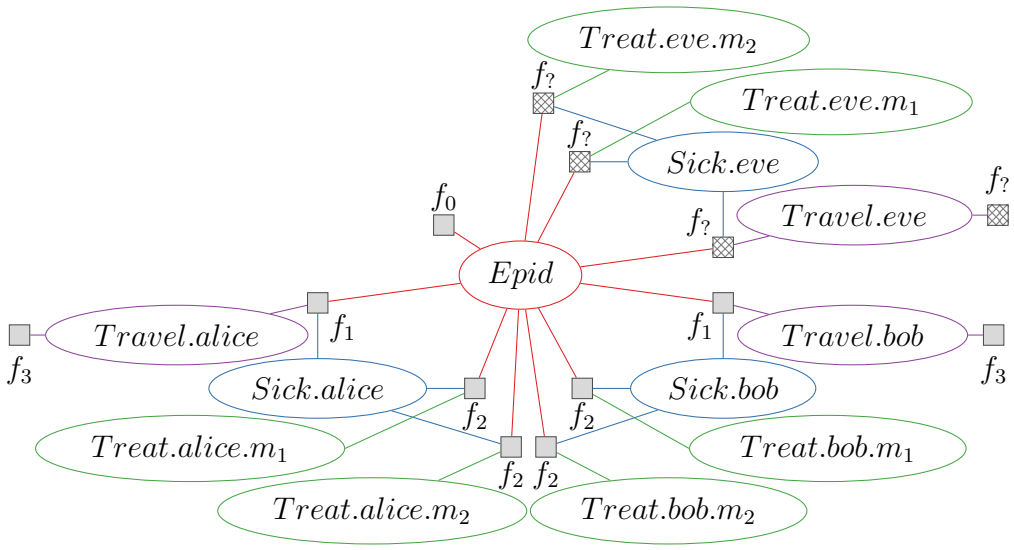


Figure 4: An extension of the epidemic example depicted in Fig. 1. The factors $f_?$ are unknown. The input-output pairs of the remaining factors are again omitted for brevity.

factors. For the upcoming definitions, let $\mathrm{Ne}_G(v)$ denote the set of neighbours of a node v (where v can be a variable node or a 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 f, 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 considering a single iteration of the colour passing procedure in the ACP algorithm.

Definition 3.1 (2-Step Neighbourhood)**.** *The* 2-step neighbourhood *of a factor f in an FG G is defined as*

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

If the context is clear, we write $2\text{-}step(f)$ instead of $2\text{-}step_G(f)$.

Example 3.1. *The 2-step neighbourhood of f_1 in the FG depicted in Fig. 3 is given by $2\text{-}step(f_1) = \{A, B\} \cup \{f_1, f_2\}$. 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 this example, $G[2\text{-}step(f_1)]$ then consists of the nodes A, B, f_1, and f_2, and contains the edges $A - f_1$, $B - f_1$, and $B - f_2$.

As it is currently unknown whether a general graph isomorphism test is solvable in polynomial time, we make use of the weaker notion of indistinguishable 2-step neighbourhoods instead of relying on isomorphic 2-step neighbourhoods to ensure that LIFAGU is implementable in polynomial time.

Definition 3.2 (Indistinguishable 2-Step Neighbourhoods)**.** *Let G denote an FG and let f_i as well as f_j denote two factors in G. Then, $G[2\text{-}step_G(f_i)]$ and $G[2\text{-}step_G(f_j)]$ are* indistinguishable *if*

1. $|\mathrm{Ne}_G(f_i)| = |\mathrm{Ne}_G(f_j)|$ *and*
2. *there exists a bijection $\tau\colon \mathrm{Ne}_G(f_i) \mapsto \mathrm{Ne}_G(f_j)$ 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 observed event (evidence) for R_k and R_ℓ is identical,* $\mathrm{range}(R_k) = \mathrm{range}(R_\ell)$*, and* $|\mathrm{Ne}_G(R_k)| = |\mathrm{Ne}_G(R_\ell)|$.

Example 3.2. *Take a look again at the FG shown in Fig. 3 and assume that there is no evidence. We can check whether f_1 and f_2 have indistinguishable 2-step neighbourhoods: Both f_1 and f_2 are connected to two randvars as* $\mathrm{Ne}(f_1) = \{A, B\}$ *and* $\mathrm{Ne}(f_2) = \{B, C\}$*, thereby satisfying Item 1. Further, A can be mapped to C with* $\mathrm{range}(A) = \mathrm{range}(C)$ *(Boolean) and* $|\mathrm{Ne}(A)| = |\mathrm{Ne}(C)| = 1$ *and B can be mapped to itself. Thus, Item 2 is satisfied and it holds that $G[2\text{-}step(f_1)]$ and $G[2\text{-}step(f_2)]$ are indistinguishable.*

Recall that our goal is to transfer potentials from known factors to unknown factors. To do so, we need a measure of similarity between factors, even if the underlying functions encoded by the factors are unknown. We thus introduce the notion of possibly identical factors, that is, factors that are indistinguishable based on their 2-step neighbourhoods. In particular, two factors are considered possibly identical if the subgraphs induced by their 2-step neighbourhoods are indistinguishable and the underlying functions (if known) do not differ from each other, as formalised in the next definition.

Definition 3.3 (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. *$G[2\text{-}step_G(f_i)]$ and $G[2\text{-}step_G(f_j)]$ are indistinguishable, and*

2. *at least one of f_i, f_j is unknown, or the underlying functions of f_i and f_j encode identical potential mappings.*

Item 2 serves to ensure consistency when comparing factors with known underlying functions as two factors encoding different potential mappings can obviously not be identical.

Example 3.3. *Applying the definition of possibly identical factors to f_1 and f_2 from Fig. 3, we can verify that f_1 and f_2 are indeed possibly identical because they have indistinguishable 2-step neighbourhoods and their underlying functions encode identical potential mappings.*

We are now ready to introduce the LIFAGU algorithm, which makes use of the notion of possibly identical factors to find known factors that are similar to unknown factors. Algorithm 1 outlines the entire LIFAGU algorithm and we provide a detailed explanation of each step in the following.

LIFAGU assigns colours to unknown factors based on indistinguishable 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. At the same time, the potentials of the factors in $C^{\ell}_{f_i}$ are transferred to f_i. Finally, ACP is called on G, which now includes the previously set colours for the unknown factors in G, to group both known and unknown factors.

Before we take a closer look at the threshold θ, we illustrate the steps undertaken by the LIFAGU algorithm on the exemplary FG from Fig. 4.

Example 3.4. *Consider again the FG G shown in Fig. 4 and assume there is no evidence available, i.e., $\boldsymbol{E} = \emptyset$. First, LIFAGU assigns colours to all known factors in G according to the potential mappings encoded by their underlying functions. In particular, all factors f_1 receive the same colour (e.g.,*

Algorithm 1: LIFAGU

Input : An FG G with randvars $\boldsymbol{R} = \{R_1, \ldots, R_n\}$, known factors $\boldsymbol{F} = \{f_1, \ldots, f_m\}$, unknown factors $\boldsymbol{F'} = \{f'_1, \ldots, f'_z\}$, and evidence $\boldsymbol{E} = \{R_1 = r_1, \ldots, R_k = r_k\}$, as well as a real-valued threshold $\theta \in [0, 1]$.

Output: A lifted representation G' of G.

1 Assign each $f_i \in \boldsymbol{F}$ a colour based on its potentials;
2 Assign each $f'_i \in \boldsymbol{F'}$ a unique colour;
3 **foreach** *unknown factor* $f_i \in \boldsymbol{F'}$ **do**
4 $C_{f_i} \leftarrow \{\}$;
5 **foreach** *factor* $f_j \in \boldsymbol{F} \cup \boldsymbol{F'}$ *with* $f_i \neq f_j$ **do**
6 **if** $f_i \approx f_j$ **then**
7 **if** f_j *is unknown* **then**
8 Assign f_j the same colour as f_i;
9 **else**
10 $C_{f_i} \leftarrow C_{f_i} \cup \{f_j\}$;
11 **foreach** *set of candidates* C_{f_i} **do**
12 $C^{\ell}_{f_i} \leftarrow$ Maximal subset of C_{f_i} s.t. $f_j \approx f_k$ holds for all $f_j, f_k \in C^{\ell}_{f_i}$;
13 **if** $|C^{\ell}_{f_i}| \,/\, |C_{f_i}| \geq \theta$ **then**
14 Assign all $f_j \in C^{\ell}_{f_i}$ the same colour as f_i;
15 Assign f_i the same potentials as the factors $f_j \in C^{\ell}_{f_i}$;
16 $G \leftarrow$ Result from calling ACP on the modified graph G and $\boldsymbol{E}$;

blue), all factors f_2 receive the same colour (e.g., green), all factors f_3 receive the same colour (e.g., purple), and f_0 receives a different colour (e.g., red). The colourings of the known factors are shown in Fig. 5a. In the next step, all unknown factors $f_?$ receive a unique colour and the resulting colourings are given in Fig. 5b. Afterwards, for every unknown factor $f_?$, LIFAGU searches for factors that are possibly identical to $f_?$. In this specific example, $f_?(Epid, Travel.eve, Sick.eve)$ is possibly identical to the factors f_1, $f_?(Treat.eve.m_1, Sick.eve, Epid)$ is possibly identical to $f_?(Treat.eve.m_2, Sick.eve, Epid)$ as well as to all factors f_2, $f_?(Treat.eve.m_2, Sick.eve, Epid)$ is possibly identical to $f_?(Treat.eve.m_1, Sick.eve, Epid)$ and to all f_2, and $f_?(Travel.eve)$ is possibly identical to the factors f_3. Consequently, in Line 8, LIFAGU assigns $f_?(Treat.eve.m_1, Sick.eve, Epid)$ and $f_?(Treat.eve.m_2, Sick.eve, Epid)$ the same colour (because they are possibly identical and both unknown). The

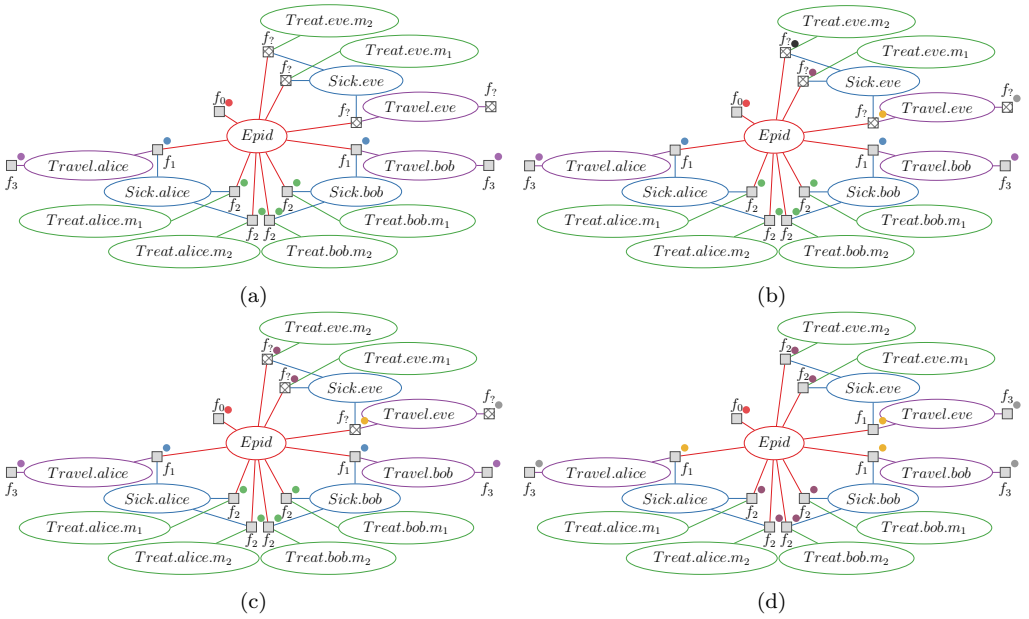


Figure 5: An illustration of the steps undertaken by Alg. 1 (LIFAGU) on the input FG depicted in Fig. 4: (a) Colouring of known factors (Line 1), (b) initial colouring of unknown factors (Line 2), (c) refined colouring for unknown factors (Line 8), and (d) grouping of unknown factors with known factors (Lines 14 and 15).

new colouring is shown in Fig. 5c. Simultaneously, LIFAGU collects for every unknown factor $f_?$ *a set* $C_{f_?}$ *of possibly identical known factors in Line 10:*

- $C_{f_?(Epid,Travel.eve,Sick.eve)} = \{f_1(Epid, Travel.alice, Sick.alice), f_1(Epid, Travel.bob, Sick.bob)\}$,
- $C_{f_?(Treat.eve.m_1,Sick.eve,Epid)} = \{f_2(Treat.alice.m_1, Sick.alice, Epid), f_2(Treat.alice.m_2, Sick.alice, Epid), f_2(Treat.bob.m_1, Sick.bob, Epid), f_2(Treat.bob.m_2, Sick.bob, Epid)\}$,
- $C_{f_?(Treat.eve.m_2,Sick.eve,Epid)} = \{f_2(Treat.alice.m_1, Sick.alice, Epid), f_2(Treat.alice.m_2, Sick.alice, Epid), f_2(Treat.bob.m_1, Sick.bob, Epid), f_2(Treat.bob.m_2, Sick.bob, Epid)\}$,
- $C_{f_?(Travel.eve)} = \{f_3(Travel.alice), f_3(Travel.bob)\}$.

Thereafter, LIFAGU computes for every candidate set $C_{f_?}$ *a maximal subset* $C^{\ell}_{f_?}$ *of pairwise possibly identical factors. Here, it holds that within each*

$C_{f_?}$, all factors are pairwise possibly identical and hence, we have $C_{f_?} = C^{\ell}_{f_?}$ for all unknown factors $f_?$. Due to $C_{f_?} = C^{\ell}_{f_?}$, we have $|C^{\ell}_{f_?}| \,/\, |C_{f_?}| = 1$ for all unknown factors $f_?$ and thus, the if-condition in Line 13 is satisfied independent of the choice of θ in this example. In consequence, the factors f_1 receive the same colour as $f_?(Epid, Travel.eve, Sick.eve)$ and the potentials of the factors f_1 are transferred to $f_?(Epid, Travel.eve, Sick.eve)$, the factors f_2 receive the same colour as $f_?(Treat.eve.m_1, Sick.eve, Epid)$ (and $f_?(Treat.eve.m_1, Sick.eve, Epid)$) and the potentials of f_2 are transferred to $f_?(Treat.eve.m_1, Sick.eve, Epid)$ and $f_?(Treat.eve.m_1, Sick.eve, Epid)$, and the factors f_3 receive the same colour as $f_?(Travel.eve)$ and the potentials of f_3 are transferred to $f_?(Travel.eve)$. The resulting colourings are shown in Fig. 5d, where all unknown factors $f_?$ have been replaced by known factors. Finally, LIFAGU calls ACP on the resulting graph G from Fig. 5d to obtain the lifted representation of G illustrated in Fig. 2.

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 value for 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 (because otherwise they would violate Item 2 of Definition 3.3 and hence 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, LIFAGU 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.[1]

Theorem 3.1. *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.*

Proof. Let G be an FG with known factors $\boldsymbol{F} = \{f_1, \ldots, f_m\}$ and unknown factors $\boldsymbol{F'} = \{f'_1, \ldots, f'_z\}$ such that for each unknown factor $f_i \in \boldsymbol{F'}$ there exists at least one known factor $f_j \in \boldsymbol{F}$ such that $f_i \approx f_j$. Then, it is guaranteed for each unknown factor f_i that Line 10 in Alg. 1 is executed at

[1] As the semantics of an FG is given by a product over its factors, the semantics of the FG is only well-defined if all input-output mappings of the factors are known.

least once and thus C_{f_i} is non-empty for all unknown factors f_i. Afterwards, as $C_{f_i} \neq \emptyset$ holds for all unknown factors f_i, it holds that in Line 12 there is at least one element in $C^{\ell}_{f_i}$ for every unknown factor f_i. Therefore, if we set $\theta = 0$, the if-condition in Line 13 passes successfully for each unknown factor f_i, resulting both in f_i being assigned the same colour as at least one known factor $f_j \in C^{\ell}_{f_i}$ as well as in the transfer of f_j's potentials to f_i. □

The threshold θ defines the trade-off between transferring as much potentials from known factors to unknown factors as possible and avoiding incorrect groupings of unknown factors. While a small threshold θ might be able to provide guarantees about a well-defined semantics of the output of LIFAGU, a larger threshold θ might be able to reduce incorrectly grouped unknown factors. In particular, for an unknown factor f_i, it is generally possible that $C^{\ell}_{f_i}$ is not unique, i.e., there are multiple maximal subsets of candidates of known factors f_i might be grouped with. The threshold θ can be used to avoid such scenarios by setting $\theta > 0.5$.

Theorem 3.2. *Let C_{f_i} be the set of known factors possibly identical to an unknown factor f_i and $C^{\ell}_{f_i}$ a maximal subset of C_{f_i} with $f_j \approx f_k$ for all f_j, $f_k \in C^{\ell}_{f_i}$. Then, $C^{\ell}_{f_i}$ is guaranteed to be unique if $|C^{\ell}_{f_i}| \,/\, |C_{f_i}| > 0.5$.*

Proof. Let C_{f_i} be a set of known factors possibly identical to an unknown factor f_i, $C^{\ell}_{f_i} \subseteq C_{f_i}$ a maximal subset such that $f_j \approx f_k$ holds for all $f_j, f_k \in C^{\ell}_{f_i}$, and $|C^{\ell}_{f_i}|/|C_{f_i}| > 0.5$. For the sake of contradiction, assume that there is another maximal subset $C^{\ell'}_{f_i} \subseteq C_{f_i}$ containing only pairwise possibly identical factors with $C^{\ell'}_{f_i} \neq C^{\ell}_{f_i}$ and $|C^{\ell'}_{f_i}| = |C^{\ell}_{f_i}|$. As $|C^{\ell}_{f_i}| > 0.5 \cdot |C_{f_i}|$ holds, there must be a factor, say f_j, with $f_j \in C^{\ell'}_{f_i} \cap C^{\ell}_{f_i}$. Consequently, f_j is pairwise possibly identical to all $f_k \in C^{\ell'}_{f_i}$ and to all $f_l \in C^{\ell}_{f_i}$, and as all f_k and f_l are known, this implies that both all f_k as well as all f_l have the same potentials as f_j, meaning the f_k and f_l must be pairwise possibly identical as well. This implies that $C^{\ell'}_{f_i} = C^{\ell}_{f_i}$ because if there were a factor f_r with $f_r \in C^{\ell'}_{f_i}$ and $f_r \notin C^{\ell}_{f_i}$, then $C^{\ell}_{f_i}$ can not be maximal as f_r is pairwise possibly identical to all $f_l \in C^{\ell}_{f_i}$. A contradiction to our assumption that $C^{\ell'}_{f_i} \neq C^{\ell}_{f_i}$. □

To close this section, we prove that LIFAGU is a generalisation of ACP, i.e., both algorithms compute the same result for input FGs containing only known factors (independent of θ because θ only affects unknown factors).

Theorem 3.3. *Given an FG that contains only known factors, ACP and LIFAGU output identical groupings of randvars and factors, respectively.*

Proof. Let G be an FG containing only known factors. Then, only Lines 1 and 16 of Alg. 1 are executed—which is equivalent to calling ACP on G. □

Before we evaluate the practical performance of LIFAGU empirically, we extend LIFAGU to incorporate background knowledge about factors belonging to the same individual object in the upcoming section.

4. Incorporating Background Knowledge in LIFAGU

We start this section by first defining the concept of background knowledge and afterwards elaborate on how given background knowledge can be incorporated into LIFAGU. Informally speaking, in our setting, background knowledge specifies which factors belong to the same individual object.

Definition 4.1 (Background Knowledge)**.** *Let G denote an FG with known factors $\boldsymbol{F} = \{f_1, \ldots, f_m\}$ and unknown factors $\boldsymbol{F'} = \{f'_1, \ldots, f'_z\}$. Then,* background knowledge $\mathcal{K} = \langle \boldsymbol{K}_1, \ldots, \boldsymbol{K}_d \rangle$ *is a collection of sets of factors $\boldsymbol{K}_1, \ldots, \boldsymbol{K}_d$ such that $\boldsymbol{K}_i \subseteq \boldsymbol{F} \cup \boldsymbol{F'}$, $i \in \{1, \ldots, d\}$, specifies a set of factors belonging to the same individual i. We say that $\mathcal{K}$ is valid if $\boldsymbol{K}_i \cap \boldsymbol{K}_j = \emptyset$ for all pairs of $\boldsymbol{K}_i \in \mathcal{K}$ and $\boldsymbol{K}_j \in \mathcal{K}$ with $i \neq j$.*

As background knowledge tells us which factors belong to the same individual, we are only interested in valid background knowledge, i.e., background knowledge in which each factor belongs to at most one individual. From now on, we therefore use the term *background knowledge* to refer to valid background knowledge only. Note that background knowledge might not be available for every individual and it is also possible that there is no background knowledge available at all. In general, there are (at least) two possible approaches to make use of background knowledge in LIFAGU:

1. Group unknown factors with known factors solely based on the available background knowledge instead of searching for a maximal subset of known factor candidates which is then used to group unknown factors with known factors if a given threshold is reached.
2. Make the decision of whether an unknown factor should be grouped with a set of known factors based on a combination of the threshold and background knowledge.

Using only the available background knowledge to group unknown factors with known factors is mostly not desirable because background knowledge

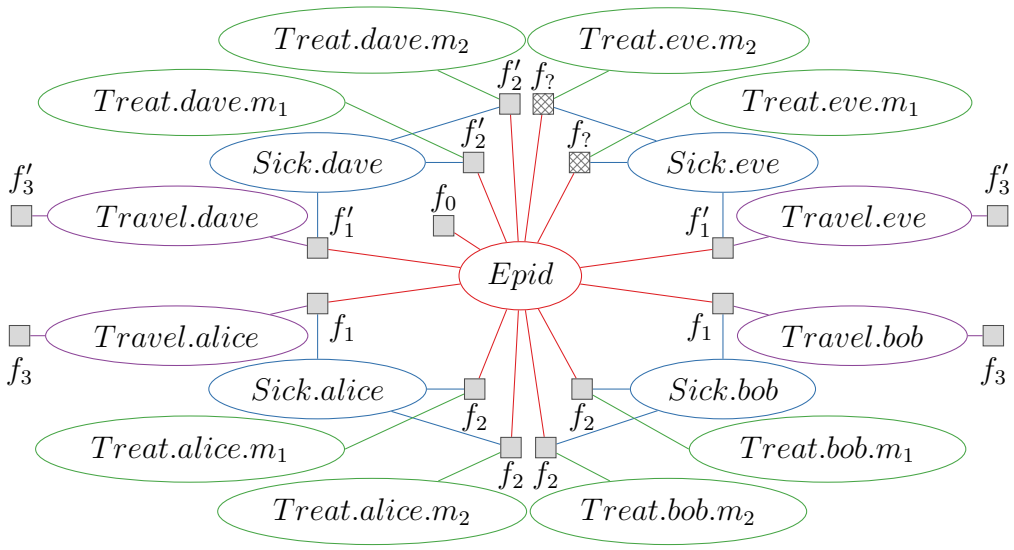


Figure 6: A slightly modified and extended version of the epidemic example depicted in Fig. 4. The factors $f_?$ are unknown and the input-output pairs of the remaining factors are again omitted for brevity. Note that the factors f_1 encode a different underlying function than the factors f'_1 and the factors f_2 encode a different underlying function than f'_2.

often is limited. Combining the threshold with background knowledge, however, possibly reduces the ambiguity for possible transfers of potentials from known factors to unknown factors. We next explain how the threshold can be combined with background knowledge in LIFAGU to determine which set of known factors an unknown factor should be grouped with.

Let us take a look at the FG G depicted in Fig. 6, where four individuals *alice*, *bob*, *dave*, and *eve* are part of the model. In this example, *alice* and *bob* belong to the same group of identically behaving individuals as they share the same potentials for the factors f_1, f_2, and f_3, i.e., it holds that $f_1(Travel.alice, Sick.alice, Epid) = f_1(Travel.bob, Sick.bob, Epid)$ for all possible assignments of the arguments of the factors f_1, and analogously for all factors f_2 and f_3. Further, let us assume that $f'_1 \neq f_1$, $f'_2 \neq f_2$, and $f'_3 \neq f_3$ hold—that is, *dave* belongs to a different group than *alice* and *bob* (formally, this can be encoded in a PFG by using constraints). Additionally, G contains another individual *eve*, for which we have only limited data available. In particular, we do not know the exact potentials for the factors $f_?(Treat.eve.m_1, Sick.eve, Epid)$ and $f_?(Treat.eve.m_2, Sick.eve, Epid)$. However, we do know the potentials for $f'_1(Epid, Travel.eve, Sick.eve)$ and

for $f'_3(Travel.eve)$, and for the sake of the example, let us assume that we are also given the background knowledge $\mathcal{K} = \langle \boldsymbol{K}_1, \boldsymbol{K}_2 \rangle$ with

- $\boldsymbol{K}_1 = \{f'_1(Epid, Travel.eve, Sick.eve), f_?(Treat.eve.m_1, Sick.eve, Epid), f_?(Treat.eve.m_2, Sick.eve, Epid), f'_3(Travel.eve)\}$ and
- $\boldsymbol{K}_2 = \{f'_1(Epid, Travel.dave, Sick.dave), f'_2(Treat.dave.m_1, Sick.dave, Epid), f'_2(Treat.dave.m_2, Sick.dave, Epid), f'_3(Travel.dave)\}$.

In other words, we know which factors belong to the individuals *dave* and *eve* but we do not have any information about the remaining factors in G.

Since it holds that *dave* and *eve* share the same potentials for the factors f'_1, i.e., $f'_1(Travel.dave, Sick.dave, Epid) = f'_1(Travel.eve, Sick.eve, Epid)$ for all possible assignments of the arguments of the factors f'_1, we know that with respect to f'_1, *dave* and *eve* belong to a group of identically behaving individuals (analogously for f'_3). However, we do not know whether the factors $f_?$ should be grouped with the factors f_2, f'_2, or none of them. At this point, we can apply our background knowledge $\mathcal{K}$: As *dave* and *eve* share the same potentials for the factors f'_1 as well as for f'_3 and we know that the factors $f_?$ belong to *eve* as well as that the factors f'_2 belong to *dave*, we might want to decide to group the factors $f_?$ with the factors f'_2 to achieve that *dave* and *eve* are grouped together.

Generally, we thus aim to prefer grouping an unknown factor with a group of known factors that is supported by the available background knowledge. The next definition formalises the idea of supporting background knowledge.

Definition 4.2 (Supporting Background Knowledge)**.** *Let $\mathcal{K} = \langle \boldsymbol{K}_1, \ldots, \boldsymbol{K}_d \rangle$ be given background knowledge, C_{f_i} a set of known factors possibly identical to an unknown factor f_i, and $C^s_{f_i}$ a subset of C_{f_i} such that $f_j \approx f_k$ holds for all $f_j, f_k \in C^s_{f_i}$. We say that $C^s_{f_i}$ is* supported *by $\mathcal{K}$ if*

1. *there exists no set $\boldsymbol{K}_i \in \mathcal{K}$ such that $f_i \in \boldsymbol{K}_i$, or*
2. *there exists a set $\boldsymbol{K}_i \in \mathcal{K}$ such that $f_i \in \boldsymbol{K}_i$, and*
   i. *for all known factors $f_\ell \in \boldsymbol{K}_i$ it holds that there exists at most one set $\boldsymbol{K}_o \in \mathcal{K}$, $\boldsymbol{K}_o \neq \boldsymbol{K}_i$, which contains a factor having the same colour as f_ℓ and all factors in $C^s_{f_i}$ have the same colour as f_ℓ, and*
   ii. *the set $\boldsymbol{K}_o$ is the same set for all known factors $f_\ell \in \boldsymbol{K}_i$.*

The notion of supporting background knowledge can be integrated into LIFAGU by searching for subsets of candidates of known factors that are

pairwise possibly identical and that are supported by the given background knowledge instead of searching for a maximal subset of candidates. More specifically, in Line 12 in Alg. 1, LIFAGU now computes all subsets $C^s_{f_i} \subseteq C_{f_i}$ such that $f_j \approx f_k$ holds for all $f_j, f_k \in C^s_{f_i}$ and then checks for all subsets $C^s_{f_i}$ whether they are supported by the given background knowledge. If no subset is supported by the given background knowledge, LIFAGU proceeds as before and takes the maximal subset $C^\ell_{f_i} \subseteq C_{f_i}$ for the transfer of potentials from known factors to unknown factors. Otherwise (i.e., in case at least one of the subsets $C^s_{f_i}$ is supported by the given background knowledge), LIFAGU takes the maximal subset of all subsets that are supported by the given background knowledge for the transfer of potentials from known factors to unknown factors. The idea behind this approach is that LIFAGU proceeds as usual if background knowledge is either missing or does not uniquely hint at a specific individual whose known factors should be used for grouping. In cases where the known factors of the individual to which f_i belongs might be grouped with known factors from various other individuals, we do not know which of these individuals should be chosen for grouping. Thus, we require that the known factors of the individual to which f_i belongs might be grouped with the known factors of a unique other individual to make use of given background knowledge.

Note that a situation like the one we considered in our toy example from Fig. 6 is abundant in many real-world applications. For example, when a new patient arrives at a hospital, there is limited data available as not all measurements are taken immediately, i.e., it is conceivable that a first examination determines the current blood pressure of the patient while measurements for other attributes are not conducted yet. Background knowledge in combination with partial measurements can help assigning the new patient to a group of indistinguishable patients, thereby allowing to draw tentative conclusions about which measurement to take next or which treatment to apply.

So far, we have introduced LIFAGU and analysed its theoretical properties. We have also extended LIFAGU to incorporate background knowledge. Next, we investigate the practical performance of LIFAGU empirically.

5. 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

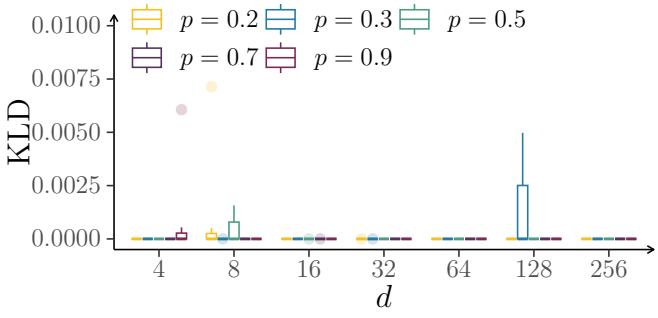
truth. Afterwards, we remove the potential mappings for 5 to 20 percent of the factors in G, yielding an incomplete FG G' on which LIFAGU is run to obtain a PFG G_{LIFAGU}. Each factor f' whose potentials are removed is chosen randomly under the constraint that there exists at least one other factor 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. There is no background knowledge available in our experiments. 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 graph structures for 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 behave identically and thus should be grouped together. We evaluate different choices for the sizes of the cohorts: There is one cohort which contains a proportion of $p \in \{0.2, 0.3, 0.5, 0.7, 0.9\}$ of all randvars in G whereas the other cohorts share the remaining proportion of $1 - p$ 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 between three and four different queries to the ground truth G and to G_{LIFAGU}, respectively. For each query, we compute the Kullback-Leibler divergence (KLD) (Kullback and Leibler, 1951) between the resulting probability distributions for the ground truth G and G_{LIFAGU} to measure the similarity of the query results. The KLD measures the difference between two distributions P and Q and is defined as

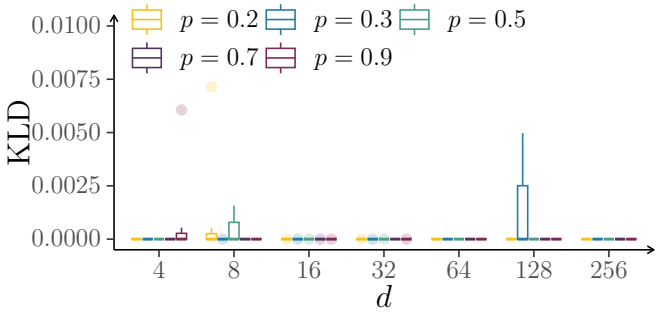
$$\text{KLD}(P \parallel Q) = \sum_x P(x) \cdot \log\left(\frac{P(x)}{Q(x)}\right). \tag{3}$$

If the distributions P and Q are identical, the KLD is zero and the larger the KLD, the more P and Q differ from each other.

In Figs. 7 and 8, we present boxplots showing the distributions of the measured KLDs for the different choices of d and p. We can observe that in



(a)

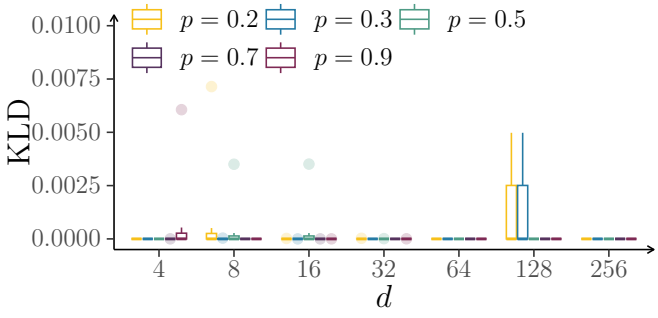


(b)

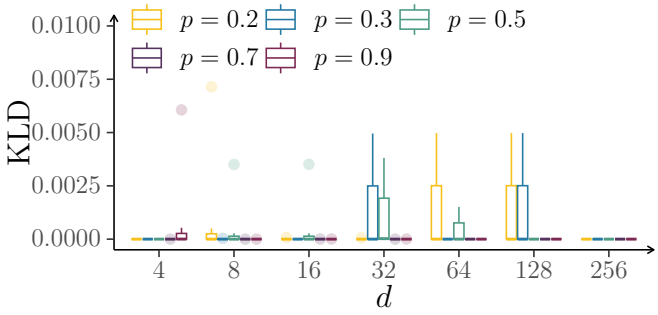
Figure 7: (a) A boxplot showing the measured KLDs for input FGs where roughly 5 percent of the factors are unknown, and (b) a boxplot showing the measured KLD for input FGs where roughly 10 percent of the factors are unknown.

every scenario, the KLD is close to zero, indicating that the query results for G_{LIFAGU} are close to the query results for the ground truth G in practice. Interestingly, there are no major differences between different choices of the parameters d and p. Even though there are some choices of d and p having slightly larger KLDs than other choices for d and p, there are especially no systematic differences between the distribution of the cohort sizes. Note that even the largest values for the KLD are still below 0.01 here.

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 of the model 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 whenever unknown factors are grouped with



(a)



(b)

Figure 8: (a) A boxplot showing the measured KLDs for input FGs where roughly 15 percent of the factors are unknown, and (b) a boxplot showing the measured KLD for input FGs where roughly 20 percent of the factors are unknown.

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 for LIFAGU.

In addition to the error measured by the KLD, 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 average run times over all scenarios are shown in Fig. 9. 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 17 different

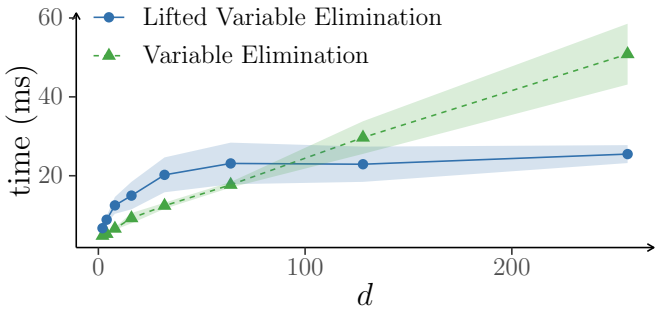


Figure 9: The average run times of variable elimination and lifted variable elimination.

groups of randvars over all settings with the largest group size being 205 (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 a more compact PFG 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 for query results. Obtaining a more compact PFG would most likely result in a higher speedup of lifted variable elimination compared to variable elimination.

Finally, we remark that assuming there exists at least one group to which a new individual can be added is clearly a strong assumption that might not hold in practical settings. We made this assumption for our experiments to guarantee a well-defined semantics of the model, as otherwise query answering would not be possible at all and hence, comparing KLDs and run times could not be performed. Despite this rather strong assumption, the experiments provide a first impression for the order of magnitude of the error induced by LIFAGU. The results on the synthetic data used in this section are promising and suggest that LIFAGU performs well in practice.

6. Conclusion

We introduce the LIFAGU algorithm to construct a lifted representation, denoted as a PFG, for an FG that possibly contains factors whose underlying potential mappings are unknown. LIFAGU is a generalisation of the ACP algorithm and allows to transfer potentials from known factors to unknown factors by identifying indistinguishable subgraph structures. Under the assumption that for every unknown factor there exists at least one known factor such that they have an indistinguishable surrounding graph structure,

LIFAGU is able to replace all unknown potential mappings in an FG by known potential mappings. To reduce ambiguity when grouping unknown factors with known factors, we introduce the concept of supporting background knowledge and show how it can be integrated into LIFAGU.

In future work, we aim to further generalise the ACP algorithm to allow for a small deviation between the potentials of two known factors f_1 and f_2 for f_1 and f_2 to be considered identical while at the same time maintaining a bounded error on probabilistic queries posed to the lifted model.

Acknowledgements

This work is partially funded by the BMBF project AnoMed 16KISA057 and 16KISA050K.

References

Ahmadi, B., Kersting, K., Mladenov, M., Natarajan, S., 2013. Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training. Machine Learning 92, 91–132.

Braun, T., Möller, R., 2016. Lifted Junction Tree Algorithm, in: Proceedings of KI 2016: Advances in Artificial Intelligence (KI-16), Springer. pp. 30–42.

Braun, T., Möller, R., 2018. Parameterised Queries and Lifted Query Answering, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-2018), IJCAI Organization. pp. 4980–4986.

De Salvo Braz, R., Amir, E., Roth, D., 2005. Lifted First-Order Probabilistic Inference, in: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05), Morgan Kaufmann Publishers Inc.. pp. 1319–1325.

De Salvo Braz, R., Amir, E., Roth, D., 2006. MPE and Partial Inversion in Lifted Probabilistic Variable Elimination, in: Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06), AAAI Press. pp. 1123–1130.

Frey, B.J., Kschischang, F.R., Loeliger, H.A., Wiberg, N., 1997. Factor Graphs and Algorithms, in: Proceedings of the Thirty-Fifth Annual Allerton Conference on Communication, Control, and Computing, Allerton House. pp. 666–680.

Gehrke, M., Möller, R., Braun, T., 2020. Taming Reasoning in Temporal Probabilistic Relational Models, in: Proceedings of the Twenty-Fourth European Conference on Artificial Intelligence (ECAI-20), IOS Press. pp. 2592–2599.

Hoffmann, M., Braun, T., Möller, 2022. Lifted Division for Lifted Hugin Belief Propagation, in: Proceedings of the Twenty-Fifth International Conference on Artificial Intelligence and Statistics (AISTATS-22), PMLR. pp. 6501–6510.

Kersting, K., Ahmadi, B., Natarajan, S., 2009. Counting Belief Propagation, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI-09), AUAI Press. pp. 277–284.

Kisyński, J., Poole, D., 2009. Constraint Processing in Lifted Probabilistic Inference, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI-09), AUAI Press. pp. 293–302.

Kschischang, F.R., Frey, B.J., Loeliger, H.A., 2001. Factor Graphs and the Sum-Product Algorithm. IEEE Transactions on Information Theory 47, 498–519.

Kullback, S., Leibler, R.A., 1951. On Information and Sufficiency. The Annals of Mathematical Statistics 22, 79–86.

Luttermann, M., Braun, T., Möller, R., Gehrke, M., 2024a. Colour Passing Revisited: Lifted Model Construction with Commutative Factors, in: Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24), AAAI Press. pp. 20500–20507.

Luttermann, M., Hartwig, M., Braun, T., Möller, R., Gehrke, M., 2024b. Lifted Causal Inference in Relational Domains, in: Proceedings of the Third Conference on Causal Learning and Reasoning (CLeaR-2024), PMLR. pp. 827–842.

Luttermann, M., Machemer, J., Gehrke, M., 2024c. Efficient Detection of Commutative Factors in Factor Graphs, in: Proceedings of the Twelfth International Conference on Probabilistic Graphical Models (PGM-2024), PMLR. pp. 38–56.

Luttermann, M., Machemer, J., Gehrke, M., 2024d. Efficient Detection of Exchangeable Factors in Factor Graphs, in: Proceedings of the 37th International FLAIRS Conference (FLAIRS-24), Florida Online Journals.

Luttermann, M., Möller, R., Gehrke, M., 2023. Lifting Factor Graphs with Some Unknown Factors, in: Proceedings of the Seventeenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-23), Springer. pp. 337–347.

Milch, B., Zettlemoyer, L.S., Kersting, K., Haimes, M., Kaelbling, L.P., 2008. Lifted Probabilistic Inference with Counting Formulas, in: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08), AAAI Press. pp. 1062–1068.

Niepert, M., Van den Broeck, G., 2014. Tractability through Exchangeability: A New Perspective on Efficient Probabilistic Inference, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14), AAAI Press. pp. 2467–2475.

Poole, D., 2003. First-Order Probabilistic Inference, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), Morgan Kaufmann Publishers Inc.. pp. 985–991.

Singla, P., Domingos, P., 2008. Lifted First-Order Belief Propagation, in: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08), AAAI Press. pp. 1094–1099.

Taghipour, N., Fierens, D., Davis, J., Blockeel, H., 2013. Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. Journal of Artificial Intelligence Research 47, 393–439.

Weisfeiler, B., Leman, A.A., 1968. The Reduction of a Graph to Canonical Form and the Algebra which Appears Therein. NTI, Series 2, 12–16. English translation by Grigory Ryabov available at https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf.