Journal of Artificial Intelligence Research 66 (2019) 33-56 Submitted 01/2019; published 09/2019

XEGGORA: Exploiting Immune-to-Evidence Symmetries with Full Aggregation in Statistical Relational Models

Mohammad Mahdi Amirian M.AMIRIAN@AUT.AC.IR
Computer Engineering & Information Technology Department
Amirkabir University of Technology, Tehran, Iran

Saeed Shiry Ghidary SHIRY@AUT.AC.IR
Math & Computer Science Department
Amirkabir University of Technology, Tehran, Iran

Abstract

We present improvements in maximum a-posteriori inference for Markov Logic, a widely used SRL formalism. Inferring the most probable world for Markov Logic is NP-hard in general. Several approaches, including Cutting Plane Aggregation (CPA), perform inference through translation to Integer Linear Programs. Aggregation exploits context-specific symmetries independently of evidence and reduces the size of the program. We illustrate much more symmetries occurring in long ground clauses that are ignored by CPA and can be exploited by higher-order aggregations. We propose Full-Constraint-Aggregation, a superior algorithm to CPA which exploits the ignored symmetries via a lifted translation method and some constraint relaxations. RDBMS and heuristic techniques are involved to improve the overall performance. We introduce XEGGORA as an evolutionary extension of ROCKIT, the query engine that uses CPA. XEGGORA evaluation on real-world benchmarks shows progress in efficiency compared to ROCKIT especially for models with long formulas.

1. Introduction

Recently, there has been a growing interest in learning and inference in Statistical Relational Learning (SRL) models. In these models, especially in Markov Logic Networks (MLN), one of the most widely used SRL formalisms, we are concerned with Maximum A-Posteriori (MAP) inference, i.e., inferring the most likely explanation (world) that maximizes the sum of the weights of the satisfied clauses.

Raised from First-Order Logic by attaching weights to the first-order formulas, MLN offers a compact representation of Markov Networks with repetitive structures. An MLN can be *grounded* to produce the Markov Network to which propositional inference algorithms are applicable. This procedure results in potentially large amounts of repetitive computations. *Lifted inference* is introduced to avoid this undesired blow-up. Both marginal and MAP lifted inference approaches in MLNs can be categorized into two major methodologies: *top-down*, which starts from a lifted model and avoids grounding as much as possible, and *bottom-up* methodology, which starts from a propositional model and detects repetitive structures (Kimmig, Mihalkova, & Getoor, 2015). The former recognizes identical structures and performs computations only once, caching the results and re-using them. Lifted belief propagation (Singla & Domingos, 2008), first-order variable elimination (Poole, 2003) and its extensions (de Salvo Braz, Amir, & Roth, 2006; Kisynski & Poole, 2009; Milch, Zettlemoyer, Kersting, Haimes, & Kaelbling, 2008), domain-lifted MAP inference over non-shared MLNs (Sarkhel, Venugopal, Singla, & Gogate, 2014a), lifted recursive conditioning (Poole, Bacchus, & Kisynski, 2011), and its extension on MLN conversion to other liftable first-order models (Kazemi, Kimmig, Van den Broeck, & Poole, 2016) are some of top-down approaches to name but a few. A key advantage of these algorithms is their

much smaller computational complexity than propositional algorithms. Their major disadvantage however, lays on the fact that the presence of evidence breaks the symmetries in the model. Models coming from real-world problems are often highly connected, containing many evidence axioms and variables. Indeed, several top-down lifted inference approaches (more precisely, those considering symmetry only as exchangeability) are evaluated on artificial datasets that are composed of a simple first-order template with many instances, resulting in a huge ground network with many indistinguishable objects.

For several algorithms, both top-down and bottom-up versions exist (Lüdtke, Schröder, Krüger, Bader, & Kirste, 2018). For example, lifted belief propagation has also a bottom-up variant (Kersting, Ahmadi, & Natarajan, 2009). For real-world MLN models, the latter methodology seems to fit better in obtaining symmetries. ROCKIT (Noessner, Niepert, & Stuckenschmidt, 2013) is one of the most famous MLN inference engines that is categorized in the bottom-up approaches, involved in several applications including ensemble matching (Meilicke, Leopold, Kuss, Stuckenschmidt, & Reijers, 2017), root cause analysis (Schoenfisch, Meilicke, von Stülpnagel, Ortmann, & Stuckenschmidt, 2018), computing LP^{MLN} (Lee, Talsania, & Wang, 2017), and data cleaning (Visengeriyeva, Akbib, & Kaul, 2016). It introduces aggregation on constraints, namely *Cutting Plane Aggregation*[1], that improves compilation from a ground network to an Integer Linear Program (ILP) by decreasing the size of the program and better exposing its symmetries. The ILP is further passed to traditional solvers and the solution is mapped to a MAP state of the original MLN.

Despite exploiting acceptable amounts of symmetries, there still exist further symmetries that CPA is unable to handle. In this work, we first illustrate some types of these symmetries by introducing *order of aggregation* and showing that CPA is able to perform only aggregations of order one, whereas higher-order aggregations are required to exploit the ignored symmetries. Secondly, a superior algorithm, namely *Full-Constraint-Aggregation* (FCA) is introduced, empowered with any possible order of aggregation with respect to the model. FCA performs a novel translation to efficient ILPs by relaxing some constraints that have no impact on the final solution. We then propose complementary techniques, including heuristics and *Relational Database Management Systems* (RDBMS) leverage to choose efficiently among multiple candidates the one in which FCA fits the best. The proposed methods are implemented within an evolutionary extension of ROCKIT, namely XEGGORA. The extension also supports numerical constraints in Markov Logic (Chekol, Huber, Meilicke, & Stuckenschmidt, 2016). Finally, we show that XEGGORA's time-preserving techniques outperform ROCKIT while applied to models with long formulas.

There also exist approaches that affect other phases of lifted inference independently of the FCA scope, such as the grounding phase. For instance, Martínez-Angeles, Dutra, Costa, and Buenabad-Chávez (2016) make use of graphics processing units to speed up grounding, and Mangal, Zhang, Kamath, Nori, and Naik (2016) introduce an eager-lazy grounding algorithm which eagerly exploits proofs and lazily refutes counterexamples. FCA is a meta-algorithm that affects the translation to ILPs and the optimization phase, and is able to be jointed with all these orthogonal algorithms. *Cutting Plane Inference* (CPI) (Riedel, 2008) is another approach that reduces the inference runtime by splitting the grounding, translation, and optimization phases into iterations. FCA is also able to get involved in CPI procedure by moving into the iterations.

1 Its owners' intention of naming the method *cutting plane aggregation* is unclear to us. However, we prefer to leave this name for the aggregation while jointed with CPI and call the single procedure c*onstraint aggregation* or *first-order aggregation.*

The rest of this paper is organized as follows. Section 2 defines the preliminaries, mainly constraint aggregation and related techniques and describes the problem. Section 3 is composed of the FCA algorithm description and its soundness proofs. Section 4 describes add-on techniques that leverage RDBMS and related heuristics to improve the overall performance. Section 5 includes ILP solving analytics and reports the empirical evaluation of FCA on six benchmark MLNs. Recent symmetry exploitation approaches with connections to this work are briefly described in Section 6, and Section 7 concludes.

2. Preliminaries

In this section, we review the concepts required for understanding the rest of the paper and define useful notations. As the evaluation part includes experiment reports in which we use CPI in conjunction with CPA and FCA, we also present a brief explanation of the algorithm, although it is orthogonal to our approach with no close relation to the contents of this paper.

2.1 First-Order Logic

A *first-order knowledge base* (KB) is a set of formulas in *first-order logic* (FOL). Formulas are constructed using four types of symbols: *constants*, *variables*, *functions*, and *predicates*. Constants represent objects in the domain of interest. Variables range over the objects. Functions represent mappings from tuples of objects to objects. Predicates represent relations among objects in the domain or attributes of objects. A *term* is any expression representing an object in the domain; it can be a constant, a variable or a function of terms. An atomic formula or *atom* is a predicate applied to a tuple of terms. Formulas are recursively constructed from atoms using logical connectives ¬, ∧, ∨ and quantifiers ∀, ∃. A positive *literal* is an atom; a negative literal is a negated atom. A KB is in the standard *Conjunctive Normal Form* (CNF) if it is expressed as a conjunction of *clauses* (disjunctive formulas). The *length* of a clause is the number of its constituting literals. The process of (partial) *grounding* corresponds to substituting constants for (some) all of the free variables in a predicate or a formula. An *interpretation* (*possible world*) assigns a truth value to each ground atom. In this paper, we assume function-free KB, and permit only the universal quantification on variables. For specific applications that use existential quantification in MLNs (e.g., Schoenfisch et al., 2018), some pre-processing is involved by Meilicke and Stuckenschmidt (2015) to treat it in terms of a comprehensive grounded representation. A good alternative would be using the concept of cardinality constraints in formulas as implemented within the ROCKIT system.

2.2 Markov Logic

Markov Logic (Richardson & Domingos, 2006) is a combination of first-order logic and probabilistic graphical models. It allows softening a first-order formula *f*, by attaching a real-valued weight *w* to it. A positive (negative) weight makes the formula support (penalize) worlds in which it is satisfied. Hard formulas are regular first-order formulas which are expressed with infinite weights and have to be fulfilled by every possible world. The probability of a possible world *x* is proportional to the exponential sum of the weights of the soft formulas *F* that are satisfied in that world. The formal definition is:

$$P(X = x) = \frac{1}{Z} exp\left(\sum_{i:f_i \in F} w_i n_i(x)\right)$$

where Z is a normalization constant and $n_i(x)$ is the number of true groundings of f_i in x. In the presence of evidence e, the possible world should satisfy e in addition to hard formulas, and its probability is reformulated as:

$$P(X = x|e) = \frac{1}{Z_e} exp\left(\sum_{i:f_i \in F} w_i n_i(x, e)\right)$$

where Z_e is the normalization constant with respect to e, and $n_i(x, e)$ is the number of true groundings of f_i in x that satisfy e.

MAP inference in Markov Logic corresponds to inferring the most likely possible world:

$$argmax_x P(x|e) = argmax_x \frac{1}{Z_e} exp\left(\sum_{i:f_i \in F} w_i n_i(x, e)\right) = argmax_x \sum_{i:f_i \in F} w_i n_i(x, e).$$

Thus, MAP inference reduces to finding the interpretation that maximizes the sum of the weights of the satisfied clauses. The problem is NP-hard in general. Any standard solver such as MaxWalkSAT (Kautz, Selman, & Jiang, 1997) can be used over the ground network to find the MAP solution. As an advanced alternative, this can be left as an ILP formulation (Wolsey, 1998) for efficient optimizers to be solved.

There is a rich literature of compiling computational problems to ILPs in operations research community. Thanks to advances in optimizers development, a wide variety of applications resolve queries by translating them to ILPs (or other types of optimization programs) and having an optimizer find the exact or approximate solution. Huynh and Mooney (2009) proposed a translation of MAP queries to ILPs and Noessner et al. (2013) extended the approach by applying constraint aggregation besides minor enhancements to improve efficiency. Here we describe the ILP formulation and constraint aggregation methods.

2.3 Traditional ILP Formulation

In all parts of the process, formulas are supposed to be conjunction-free, i.e., composed of disjunctive clauses. Formulas with conjunctions are treated similarly in a separate procedure (Noessner, 2014). Given a set of ground clauses $\mathcal{G}$ as input, one binary ILP variable x_ℓ is associated with each ground atom ℓ occurring in the set, assigning it a value 1 provided ℓ is true and 0 otherwise. Each evidence atom (assignment of a truth value to an atom) can be represented as a linear constraint of the form $x_\ell \leq 0$ or $x_\ell \geq 1$, or simply by replacing the corresponding ILP variable with a constant binary value. For each ground clause $g \in \mathcal{G}$ with weight $w_g \in \mathbb{R}$, a binary variable z_g with coefficient w_g is added to the objective. The objective totally becomes:

$$maximize \sum_{g \in \mathcal{G}} w_g z_g.$$

It is straightforward to design constraints considering z_g to be assigned 1 provided its corresponding ground clause is satisfied in the MAP solution and 0 otherwise. For a clause with $w_g > 0$, the following constraint is added to the ILP:

$$\sum_{\ell \in L^+(g)} x_\ell + \sum_{\ell \in L^-(g)} (1 - x_\ell) \geq z_g$$

where $L^+(g)$ and $L^-(g)$ refer to the sets of all ground atoms occurring unnegated and negated in g, respectively. A truth value assignment to any ground atom that fulfills the disjunctive clause g, makes the left-hand side of the constraint positive, thus, z_g can take its maximum possible value 1.

For hard clauses ($w_g = \infty$), no term is added to the objective and z_g in the constraint is simply replaced with 1:

$$\sum_{\ell \in L^+(g)} x_\ell + \sum_{\ell \in L^-(g)} (1 - x_\ell) \geq 1.$$

For a clause with $w_g < 0$, the optimizer tries to lower z_g to its minimum possible value 0. The following constraint is added to permit this only if none of its constituting ground atoms satisfy g:

$$\sum_{\ell \in L^+(g)} x_\ell + \sum_{\ell \in L^-(g)} (1 - x_\ell) \leq (|L^+(g)| + |L^-(g)|)\, z_g.$$

Finally, for clauses with zero weights, no constraint is added to the ILP as they have no effect on the MAP solution.

Example 1 *Consider an MLN with a single formula of weight 1.4 stating that each kid may be happy of having a kind parent:*
1.4 Child(k, p) ∧ Kind(p) ⇒ Happy(k).

The formula is expressed in the standard form as:
1.4 ¬Child(k, p) ∨ ¬Kind(p) ∨ Happy(k).

By grounding and applying evidence:
Child(Mary, Jack),
Child(Mary, Rose),
Child(Bob, Jack),
Child(Kate, Jack),

the essential ground clauses would be:
g_1: 1.4 ¬Kind(Jack) ∨ Happy(Mary)
g_2: 1.4 ¬Kind(Rose) ∨ Happy(Mary)
g_3: 1.4 ¬Kind(Jack) ∨ Happy(Bob)
g_4: 1.4 ¬Kind(Jack) ∨ Happy(Kate)

This MLN is translated to the following ILP (all variables types are binary):

maximize $1.4z_1 + 1.4z_2 + 1.4z_3 + 1.4z_4$
subject to:
c_1: $(1 - Kind_{Jack}) + Happy_{Mary} \geq z_1$,
c_2: $(1 - Kind_{Rose}) + Happy_{Mary} \geq z_2$,
c_3: $(1 - Kind_{Jack}) + Happy_{Bob} \geq z_3$,
c_4: $(1 - Kind_{Jack}) + Happy_{Kate} \geq z_4$.

By the essential ground clauses, we mean having inessential, say, free groundings whose truth values have no effect on the MAP state removed from the entire ground clauses. For instance, generating the ground clause: 1.4 ¬Child(Bob, Rose) ∨ ¬Kind(Rose) ∨ Happy(Bob) in Example 1 is inessential. In other words, we always perform grounding only if we need to. This is the only reason for involving evidence in the given example.

2.4 Cutting Plane Inference

The well-known idea of using cutting planes from operations research (Dantzig, Fulkerson, & Johnson, 1954) was involved by Riedel (2008) in ILP solving of MAP inference for Markov Logic. The algorithm starts with the given evidence and the trivial parts of the KB (e.g. unary clauses) and solves the corresponding ILP which can be actually much smaller than the ILP formulation of the whole MLN. The solution is then checked if it violates any constraints. The violated constraints (if any) are then added to the program and solved again. This incremental process iterates until no violated constraint remains. It is shown by experiments that CPI can solve previously intractable MAP inference queries within short times.

2.5 Constraint Aggregation

Grounding formulas in MLN often results in symmetries in multiple ground clauses, and consequently, in the ILP constraints. Constraint aggregation proposes aggregating ground clauses that include symmetries, resulting in smaller constraint matrices and aiding symmetry detection algorithms of the ILP solver. The candidate sets of ground clauses to which aggregation can be applied are defined as follows:

Definition 1 *Let $G \subseteq \mathcal{G}$ be a set of n weighted ground clauses and let c be a ground clause. We say that G can be aggregated with respect to c if (a) all ground clauses in G have the same weight and (b) for every $g_i \in G, 1 \leq i \leq n$, we have that $g_i = \ell_i \vee c$ where ℓ_i is a positive or a negative literal.*

For such a set, all corresponding ILP constraints can be replaced by fewer constraints with the following rules:

Definition 2 (First-Order Aggregation Rules) *Let $G \subseteq \mathcal{G}$ be a set of n ground clauses with weight w_G that can be aggregated with respect to a ground clause c.*

I. *In case of a finite weight ($w_G \neq \infty$), replace the corresponding terms in the ILP objective with a new integer variable $z_G \in \{0, .., n\}$ with coefficient w_G:*

$$\textit{maximize } w_G z_G \textit{ + rest of the objective.}$$

If $w_G > 0$, replace the corresponding ILP constraints with:

$$\sum_{p|p \vee c \in G} x_p + \sum_{q|\neg q \vee c \in G} (1 - x_q) + \sum_{\ell \in L^+(c)} n x_\ell + \sum_{\ell \in L^-(c)} n(1 - x_\ell) \geq z_G,$$

and if $w_G < 0$[2], with:

$$\sum_{p|p \vee c \in G} x_p + \sum_{q|\neg q \vee c \in G} (1 - x_q) \leq z_G,$$
$$n x_\ell \leq z_G \qquad for\ every\ \ell \in L^+(c),$$
$$n(1 - x_\ell) \leq z_G \quad for\ every\ \ell \in L^-(c).$$

II. *In case of an infinite weight ($w_G = \infty$)[3], only replace the corresponding ILP constraints with:*

$$\sum_{p|p \vee c \in G} x_p + \sum_{q|\neg q \vee c \in G} (1 - x_q) + \sum_{\ell \in L^+(c)} n x_\ell + \sum_{\ell \in L^-(c)} n(1 - x_\ell) \geq n.$$

According to the aggregation rules, z_G is considered to be assigned the number of satisfied ground clauses in G; it gains the maximum value *n* provided a solution satisfies the ground clause *c*, otherwise it is equal to the number of the literals (of the form *p* or ¬*q*) satisfied by the solution. CPA algorithm chooses greedily among sets that can be aggregated to a more compact ILP and applies constraint aggregation.

2,3 Despite the perfect explanation of the process by Noessner et al. (2013) and Noessner (2014), the implementation of ROCKIT (up to the current ver. 0.5.276) lacks the aggregation of hard clauses and also happens to generate irrelevant ILP constraints for the case of negative weights which result in incorrect solutions. These issues are fixed in XEGGORA.

A fundamental distinction should be considered in the resulting ILP between negative and positive weights: More constraints are required for negative weights to bind z_G with the desired value. Intuitively, satisfying each negative weighted ground clause brings one penalty in the objective; hence, the common disjunctive part (*c*) of the aggregated clauses should be checked conjunctively for each of its constituting literals to bring *n* penalties if satisfied. To be more precise, adding each integer variable of the mathematical program with a negative weighted representative in the objective leads to minimizing that variable, which in turn, appears in the right-hand side of multiple inequality constraints to minimize the maximum of all the left-hand side values. This is regarded as *minimax multi-objective optimization* in operations research community (Aissi, Bazgan, & Vanderpooten, 2009).

Example 2 *Among the ground clauses of Example 1,* $\{g_1, g_2\}$ *can be aggregated with respect to Happy(Mary), while* $\{g_1, g_3, g_4\}$ *can be aggregated with respect to ¬Kind(Jack).* $\{g_1, g_3, g_4\}$ *is chosen by greedy for aggregation. The terms:* $1.4z_1 + 1.4z_3 + 1.4z_4$ *in the objective are replaced by* $1.4z_{1,3,4}$, $z_{1,3,4} \in \{0..3\}$ *and the ILP constraints* $\{c_1, c_3, c_4\}$ *are replaced by:*

$$c_{1,3,4}: Happy_{Mary} + Happy_{Bob} + Happy_{Kate} + 3\left(1 - Kind_{Jack}\right) \geq z_{1,3,4}.$$

A distinctive advantage of the aggregation approach is its immunity to evidence in contrast to other lifting methods and relates to the different views of the symmetry concept. This can be explained by justifying the aggregation rules versus other symmetry exploitation techniques. Conventional lifting approaches (e.g. fractional automorphisms for MAP LPs by Bui, Huynh, and Riedel, 2013) define and expose symmetry as *exchangeability* which is justified by encapsulating exchangeable entities into a single representative, hence reducing the problem size. This type of symmetry is broken by evidence. The reader is referred to Van den Broeck and Darwiche (2013) for illustration. Moreover, the current exchangeable variables may not remain exchangeable by any upcoming additional formulas giving information about them. In the aggregation approach in contrast, symmetric variables do not need to be assigned the same value, but to be just capable of being *summed up* in the constraints. This capability won't suffer from evidence. The difference may be precisely explored in Example 3:

Example 3 *Consider the traditional ILP translation of the MLN in Example 1. The binary variables* $Happy_{Mary}$, $Happy_{Bob}$ *and* $Happy_{Kate}$ *are symmetric and may be unified by a conventional lifting technique to a representative, say,* $Happy_{MBK}$. *Consequently, the constraints* $\{c_1, c_3, c_4\}$ *would be grouped as a unique constraint with one binary representative in the lifted ILP objective. Adding the new evidence: "Mary is happy" or the complicated knowledge: "Only one person is not happy" to the MLN breaks the symmetries, bringing back the underlying entities to ground. The restriction is even more intensive for example in fractional automorphism based methods, because the presence of* $Happy_{Mary}$ *in* g_2 *avoids unifying the predicate with* $Happy_{Bob}$ *and* $Happy_{Kate}$ *even with no additional evidence. However, this would not happen to the aggregated constraints in Example* 2 *as the underlying variables have not been unified but summed up.*

As seen in Example 2, constraint aggregation is able to reduce the number of ILP variables and constraints. This is however not the case in Example 4, where the members of every non-singular subset of the ground clauses differ in at least two literals:

Example 4 *Consider an MLN with a single formula of weight 2.3 stating that each kid may be happy of having a kind parent who has fun with him (her):*
2.3 ¬Child(k, p) ∨ ¬Kind(p) ∨ ¬HasFunWith(p, k) ∨ Happy(k).

By grounding and applying the same evidence of Example 1, the essential ground clauses would be:
g_1: 2.3 ¬Kind(Jack) ∨ ¬HasFunWith(Jack, Mary) ∨ Happy(Mary)

g_2: 2.3 ¬Kind(Rose) ∨ ¬HasFunWith(Rose, Mary) ∨ Happy(Mary)
g_3: 2.3 ¬Kind(Jack) ∨ ¬HasFunWith(Jack, Bob) ∨ Happy(Bob)
g_4: 2.3 ¬Kind(Jack) ∨ ¬HasFunWith(Jack, Kate) ∨ Happy(Kate)

Here, there still exist similarities between the members of both $\{g_1, g_2\}$ and $\{g_1, g_3, g_4\}$, but in addition, diversity in more than one literal. These are ignored by the constraint aggregation algorithm; it allows only one literal to be distinct while all others must be identical. This is the justification of naming the method *first-order* aggregation. This failure often happens in models containing long formulas. We introduce Full-Constraint-Aggregation that can exploit this type of symmetry.

3. The FCA Method

We start by illustrating symmetry types that appear in ground clauses as in Example 4, based on a parameter we name *order of aggregation*. Then we describe our superior algorithm which performs higher-order aggregations.

Definition 3 *Let $G \subseteq \mathcal{G}$ be a set of n weighted ground clauses and let c be a ground clause with non-zero length. We say that G can be aggregated of order k with respect to c, if (a) all ground clauses in G have the same weight and (b) for every $g_i \in G$, $1 \leq i \leq n$, we have that $g_i = L_i \vee c$ where L_i is a ground clause of maximum length k. We call k the aggregation order of G, if k is the minimum order of which G can be aggregated (with respect to any ground clause c).*

Due to Definition 3, the constraint aggregation method is only applicable to the sets of ground clauses with aggregation order of 1, performing *first-order aggregation*. For a set of a higher aggregation order, we need to associate disjunctions of literals with binary terms in the optimization program, e.g. $a \vee b \equiv x_a + x_b - x_a x_b$. This results in an *Integer Polynomial Program*. To be more precise, a k[th] order trivial aggregation may be performed to formulate the MAP inference problem as an integer k[th] order mathematical program. The program can be further linearized to an equivalent ILP using the classic linearization method outlined by Watters (1967), as previously done in a different MAP inference method by Sarkhel, Venugopal, Singla, and Gogate (2014b). However, the resulting ILP would carry much more variables and constraints and last longer to be solved than the one from the traditional ILP formulation.

To overcome this issue, we propose a novel relaxed translation of MLNs to ILPs by associating auxiliary variables with disjunctive clauses. This is done through higher-order aggregation rules.

Definition 4 (Higher-Order Aggregation Rules) *Let $G \subseteq \mathcal{G}$ be a set of n ground clauses with weight w_G with the aggregation order of $k > 1$, i.e., can be aggregated of order k with respect* to *a ground clause c. Each ground clause L_i such that $L_i \vee c \in G$ is a disjunction of at most k unnegated or negated ground atoms. For each L_i with length 1, apply the First-Order Aggregation Rules. For the remainder:*

I. *Corresponding to each L_i, define a new binary ILP variable x_{s_i}. Choose one of the following Bound Constraints (BCs) after the sign of w_G and add it to the ILP:*
Upper-BC: $x_{s_i} \leq \sum_{\ell \in L^+(L_i)} x_\ell + \sum_{\ell \in L^-(L_i)} (1 - x_\ell)$ *if* $w_G > 0$ *or* $w_G = \infty$
Lower-BC: $\sum_{\ell \in L^+(L_i)} x_\ell + \sum_{\ell \in L^-(L_i)} (1 - x_\ell) \leq |L_i| . x_{s_i}$ *if* $w_G < 0$

II. *In case of a finite weight ($w_G \neq \infty$), replace the corresponding terms in the ILP objective with a new integer variable $z_G \in \{0, .., n\}$ with coefficient w_G:*
maximize $w_G z_G$ + rest of the objective.

If $w_G > 0$, replace the corresponding ILP constraints with:

$$\sum_{i=1}^{n} x_{s_i} + \sum_{\ell \in L^+(c)} n x_\ell + \sum_{\ell \in L^-(c)} n(1 - x_\ell) \geq z_G,$$

and if $w_G < 0$, *with:*

$$\sum_{i=1}^{n} x_{s_i} \leq z_G,$$
$$n x_\ell \leq z_G \qquad for\ every\ \ell \in L^+(c),$$
$$n(1 - x_\ell) \leq z_G \quad for\ every\ \ell \in L^-(c).$$

III. *In case of an infinite weight* ($w_G = \infty$), *only replace the corresponding ILP constraints with:*

$$\sum_{i=1}^{n} x_{s_i} + \sum_{\ell \in L^+(c)} n x_\ell + \sum_{\ell \in L^-(c)} n(1 - x_\ell) \geq n.$$

The Upper-BC and Lower-BC pair if used altogether, represent the disjunction operator(s) of L_i in the ILP. Intuitively, to correspond a logical OR operator for two Boolean variables: $c \equiv a \vee b$ in the integer space, one may add two inequalities: $x_c \leq x_a + x_b \leq 2x_c$. Similar inequalities would represent the disjunction of *k* Boolean variables. Now if x_c is involved in either maximization or minimization, one can drop the corresponding inessential constraint.

Example 5 *The traditional ILP formulation of Example 4 is straightforward and not mentioned here. Among the ground clauses,* $\{g_1, g_2\}$ *can be aggregated of order 2 with respect to Happy(Mary), and* $\{g_1, g_3, g_4\}$ *can also be aggregated of order 2 with respect to ¬Kind(Jack). By some heuristics,* $\{g_1, g_3, g_4\}$ *is chosen as the target for aggregation. The corresponding terms in the objective are replaced by:* $2.3z_{1,3,4}$, $z_{1,3,4} \in \{0..3\}$ *and the corresponding ILP constraints are replaced by:*

$x_{s_1} \leq \left(1 - HasFunWith_{Jack,Mary}\right) + Happy_{Mary}$,
$x_{s_3} \leq \left(1 - HasFunWith_{Jack,Bob}\right) + Happy_{Bob}$,
$x_{s_4} \leq \left(1 - HasFunWith_{Jack,Kate}\right) + Happy_{Kate}$,
$x_{s_1} + x_{s_3} + x_{s_4} + 3(1 - Kind_{Jack}) \geq z_{1,3,4}$.

3.1 ILP Analysis

Applying higher-order aggregation rules results in relaxed ILPs that are necessary to be analyzed in terms of soundness and efficiency. To this end, we have the following theorem:

Theorem *Let M be a Markov Logic Network and HOA(M) be the ILP formulation having higher-order aggregations applied to all or part of it. Each solution of HOA(M) corresponds one-to-one to a maximum a-posteriori state of M.*

Proof. The aggregation process is similar to the first-order aggregation whose soundness has been proven earlier by Noessner (2014, Theorem 3). We need moreover to show that the auxiliary relaxed variables and the BCs of rule I in Definition 4 do not affect the soundness. We build a new MLN with the same logic as *M*, and show that *HOA(M)* can be produced by applying first-order aggregation to it. Consider a set of *n* ground clauses G in *M* with weight *w* which has been aggregated of order $k > 1$ with respect to some ground clause c. Without loss of generality, suppose that the other parts of *M* are not aggregated. For each $g_i \in G$, $g_i = L_i \vee c$, $|L_i| > 1$ we add to *M*, without affecting its logic, a Boolean variable s_i and new hard formulas $s_i \Rightarrow L_i$, $L_i \Rightarrow s_i$ and substitute $s_i \vee c$ for g_i. Note that any MAP inference algorithm tries to satisfy clauses with positive weights (besides hard clauses), and dissatisfy those with negative weights. Hence for each *i*, if $w > 0$ or $w = \infty$, it never tries to dissatisfy s_i (because it appears in no ground clause except $s_i \vee c$), so $L_i \Rightarrow s_i$ has no impact on the truth values of the literals occurring in L_i, and can be relaxed (removed) . Similarly, $s_i \Rightarrow L_i$ can be relaxed if $w < 0$ because the inference engine never tries to satisfy s_i. Thus, in both cases we can drop either the upper-BC or the lower-BC (as done in rule I). It is remarkable that through relaxation, s_i is no longer equivalent to L_i;

we do not need it to be though. It is straightforward to show that the new MLN can be ILP-formulated as *HOA(M)*: For each $i \in \{1,2,..,n\}$, associate s_i with the x_{s_i} in Definition 4; each $s_i \Rightarrow L_i$ in the standard CNF form is a ground clause of maximum length $k+1$ that can form an upper-BC. Meanwhile, each $L_i \Rightarrow s_i$ is composed of at most k ground clauses in the standard form that can be first-order aggregated with respect to s_i and form a lower-BC. On the other hand, $\{s_i \vee c | 1 \leq i \leq n\}$ can be first-order aggregated with respect to *c* and form the constraints of rule II or III. □

Now we deal with the computational aspect of higher-order aggregation by comparing the number of variables and constraints of the resulting ILP before and after applying the rules. For a set of *n* ground clauses of length *m* and aggregation order of *k*, the identical clause *c* has length $m-k$. Assuming that the clauses don't share any variables except those occurring in *c*, the number of variables would be $n.k+m-k$ (for hard clauses) or $n.k+m-k+n$ (for soft clauses). The number of corresponding constraints is *n*. After applying k^{th} order aggregation, the number of variables increases by *n* (for hard clauses) or by 1 (for soft clauses). The number of constraints also increases by 1 (in case of positive weight) or by $m-k+1$ (in case of negative weight). This would increase by *n* more constraints without relaxing either upper-BCs or lower-BCs. First-order aggregation is an exception, which in case the auxiliary variables and BCs need not be added; hence we have a decrease in the ILP size as outlined by Noessner et al. (2013). For the higher aggregation orders, despite the increase in the ILP size, we empirically observed runtime enhancements of the solver. This is because the increase in the ILP size is of order of the lengths of clauses and can be ignored while compared to the size of symmetries that aggregation makes explicit to the solver.

Higher-order aggregation rules bring us the option to choose the target among wider sets of ground clauses. Consider a first-order formula that is grounded to a set of ground clauses $\mathcal{G}$. For each cluster $G \subseteq \mathcal{G}$, we can partition the constituting literals into *identical* and *distinct* parts. Identical literals are the ones that every clause in G shares exactly the same grounding for. Aggregation would be applicable if at least one identical literal exists. Therefore, no similarity is ignored and a full aggregation can be performed. The order of aggregation would be the number of distinct literals that can range from 1 to the length of the clauses minus 1.

The FCA algorithm is composed of three phases that are executed iteratively: It first seeks the candidate clustering schemes of ground clauses in the MLN for aggregation. Then it chooses a target among them by heuristics and finally applies higher-order aggregation rules to the target. The first two phases should be essentially time-preserving and efficiency-preserving, so that the enhancement earned by the third phase would not be neutralized, i.e., efficient candidates should be chosen to fit the best for aggregation, and this should be done efficiently. To this end, multiple techniques are involved that are described in the next section.

4. Leveraging RDBMS and Heuristics

We showed analytically a major difference between the first-order and higher-order aggregations. First-order aggregation, if applicable, reduces the ILP size on the order of the number of aggregated clauses, which results in much faster program solving. This is also verified through experiments on the artificial data. In order to compare the efficiency between possible clustering schemes for higher-order aggregations, one may obtain desired metrics. However, designing such metrics is not trivial because efficiency is an abstract measure that relates to the cost of the aggregation procedure in terms of size and runtime, along with the cost of solving the ILP. Moreover, involving a metric in the algorithm would acquire

time-consuming computations. Instead, we propose a heuristic approach to choose an appropriate clustering scheme for aggregation as described in the following algorithm. In order to benefit from query optimization technologies, this algorithm leverages RDBMS to compute the compactness of possible aggregations. The idea of RDBMS leverage for MAP inference in Markov Logic was inspired by Riedel (2008) and evolved by Niu, Ré, Doan, and Shavlik (2011) and Noessner et al. (2013). They used RDBMS in the grounding phase and also in finding the violated constraints, but the computation of counting features was not implemented with RDBMS because it requires storing or regeneration of the ground table which is often inefficient for first-order aggregation. In the case of higher-order aggregations however, counting the number of rows is required several times which is done by the `COUNT DISTINCT` query. It is a standard SQL query used for counting the number of distinguishable rows on the specified columns in a table. Extensionally, MySQL permits using it multiple times in a single query.

Algorithm CHOOSEFORAGGREGATION

Input $\mathcal{F}$: a first-order disjunctive clause
Input $\mathcal{G}$: a SQL table, containing all essential ground clauses of $\mathcal{F}$
Output appropriate clustering scheme as the target for aggregation

1: L ← $Literals\_of\ (\mathcal{F})$
2: $\mathcal{V}$ ← $Variables\_of$ (L)
3: **if** |L| = 1
4: *candidate_sets* ← {∅}
5: **else**
6: *candidate_sets* ← {}
7: **foreach** non-empty $V \subseteq \mathcal{V}$
8: *identical_literals* ← $\{\ell_i \in L \mid Variables\_of\ (\ell_i) \subseteq V\}$
9: **if** *identical_literals* ∉ *candidate_sets* and *identical_literal*s ≠ L
10: add *identical_literals* to *candidate_sets*
11: **if** *candidate_sets* contains any sets of literals with the size of |L| - 1
12: **return** best of them greedily to be further first-order aggregated
13: **else**
14: *query* ← `'SELECT '`
15: **foreach** *identical_part* ∈ *candidate_sets*
16: *query* ← *query* + `'COUNT (DISTINCT '` + $Serialize$ (*identical_part*) + `') as '` + $Caption$ (*identical_part*) [+ `','`] *// except for the last loop*
17: *query* ← *query* + `'FROM '` + $\mathcal{G}$
18: execute *query* into *#clusters*
19: *best_candidate_sets* ← {argmin$_{\#clusters}$ *candidate_sets*}
20: **return** argmax$_{\text{cardinality}}$ *best_candidate_sets*

The algorithm is provided with a disjunctive clause and a table containing all of its essential groundings in the rows and its constituting variables in the columns. Inessential groundings (e.g. by the substitution of $\{k \mapsto Bob,\ p \mapsto Rose\}$ in Example 4) should have been previously ignored in order to produce an effective output. The algorithm first acquires the set of all literals that constitute the clause and also all variables that participate in grounding (lines 1, 2). In lines 3-6, a set is initialized for storing all feasible partitioning schemes of the literals. Each *candidate set* is a representative for the set of identical

literals (i.e., clause *c* in Definition 3), so the complementary literals are considered distinct. The cardinality of a candidate set represents the clause length minus its proposing order of aggregation. $\emptyset$ is by default a feasible candidate set. However, it is added only for first-order aggregation as it would not be efficient for higher orders, as considered in Definition 3. Lines 7-10 collect candidate sets for aggregation. One may trivially collect the sets based on ground literals; however, we build the collection method upon non-empty sets of variables, in order to avoid producing infeasible candidate sets. It starts with partitioning the variables. Each partitioning scheme of variables specifies a feasible partitioning scheme of literals which in turn corresponds to a clustering scheme of the essential ground clauses. Line 11 discovers if any candidates are present for first-order aggregation, to be in case the arguments of a call to the procedure (line 12). If there exist more than one candidate, one would be chosen greedily as done in CPA. Lines 14-18 contain the code for generating and executing the SQL query that counts the number of clusters aggregated according to each partitioning scheme. Each single clause that is not aggregated is considered in a singleton cluster. We involve the query output *#clusters* as a lookup table which returns the number of clusters given a candidate set. Finally, the candidate partitioning scheme that offers the minimum number of clusters is selected. If there exist multiple solutions, the one with the lowest aggregation order is returned (lines 19-20). This is motivated from the analysis on the artificial data; keeping other parameters fixed, lower-order aggregations often produce slightly more efficient ILPs.

Example 6 *Consider $\mathcal{F}$ to be the first-order clause:* $\neg\text{Kind(p)} \vee \neg\text{HasFunWith(p, k)} \vee \text{Happy(k)}$. *Also consider $\mathcal{G}$ to contain its essential groundings from Example 4 as stored in the following SQL table:*

k	p
Mary	Jack
Mary	Rose
Bob	Jack
Kate	Jack

Following the algorithm onto line 13, the candidate sets would become $\{\{\text{k}\},\{\text{p}\}\}$. *To elaborate the process, suppose exceptionally that we permit the universal set to be also a candidate. The counting SQL query given the candidate sets:* $\{\{\text{k}\},\{\text{p}\},\{\text{k, p}\}\}$ *would be generated as:*

```
SELECT COUNT (DISTINCT k) as '{Happy(k)}'
     , COUNT (DISTINCT p) as '{¬Kind(p)}'
     , COUNT (DISTINCT k, p) as '{¬Kind(p), ¬HasFunWith(p, k), Happy(k)}'
FROM 𝒢
```

By executing the query, we are provided with a lookup table for #clusters as:

candidate set	#clusters
$\{\text{Happy(k)}\}$	3
$\{\neg\text{Kind(p)}\}$	2
$\{\neg\text{Kind(p)}, \neg\text{HasFunWith(p, k)}, \text{Happy(k)}\}$	4

Finally, the 2nd candidate set: $\{\neg\text{Kind(p)}\}$ *is chosen and returned as the identical part for aggregation.*

An extra point which is not described here in detail is that we collect the candidate sets in an elegant order by which the number of costly `COUNT DISTINCT` commands in the query becomes fewer than as done by brute force, and the last argmaxing step in the algorithm becomes inessential.

5. Empirical Evaluation

This section is composed of two main evaluation reports. In order to design efficient heuristics for FCA as described earlier, the ILP resolution phase was empirically analyzed and evaluated on the artificial data. Furthermore, the whole algorithm was experimented on real benchmark MLNs. In both evaluations, Gurobi[4] version 8 was employed as the ILP solver to find an exact or approximate solution based on a *gap* parameter (bound of relative error). The gap was set to 10^{-6} to reach the exact solution in the experiments it is reachable. For each benchmark with intractable exact inference, we tried various gap ranges to find the best approximation in admissible time. All experiments were performed on a PC with 8 GB RAM and 4 cores with 2.1 GHz.

5.1 Analysis on the Artificial Data

In order to elaborate the enhancement achieved by higher-order aggregations, we first implemented a prototype system to generate artificial ILPs that can be produced from the sets of ground clauses with various lengths and aggregation orders. We generated two batches of ILPs as the translations by the traditional (TRAD) and FCA methods from the same ground clauses of lengths from 1 to 10. The two ILP batches were then passed to the ILP solver and the solving times were measured and compared to test FCA scalability. In addition to the length and the aggregation order features, we needed to classify the operational domain based on two more features in order to analyze the impact of our approach more precisely: the sign of the clauses weights and the presence of evidence. All tests were done upon both positive and negative-weighted sets of clauses, without and in the presence of evidence for 5%, 10% and 20% of all atoms. Figure 1 displays the ILP solving times for a set of 100000 ground clauses of length 10 and various aggregation orders. The times are calculated by the average on 5 runs. Based on the preceding features, these results are observed from the experiments:

A. Observations on the TRAD curves characteristics:
   I. All curves slope downward from right to left, i.e., more symmetries in the model cause faster inference. These symmetries are originally implicit in the ILP and exposed by the solver itself, leading to faster optimization. FCA is supposed to better expose the symmetries by making them explicit to the solver.
   II. Solve time rises with an increase in the portion of evidence. Moreover, a significant gap is observed between the curves of evidence-free models and the rest. This is because evidence (even over few atoms) breaks variables exchangeability which in turn prevents the variables encapsulation.
   III. Among the evidence-free models, no significant difference is observed between inference on positive and negative-weighted clauses, while among clauses including constrained atoms (caused by evidence), inference on those with negative weights is slightly slower. Briefly arguing, the difference may be due to the way TRAD encodes weighted satisfiability constraints as ILPs and solves them in turn. As explained earlier, the translation of a negative weighted clause comes with a coefficient $(|L^+(g)| + |L^-(g)|)$ for one binary variable. By LP relaxation, the variable is temporarily relaxed to accept real values and later discretized somehow (trivially rounded) to permit only binary values. The gap between the optimal solutions of the LP and its underlying IP often causes this relaxation/discretization formalism generate non-optimal (or infeasible) intermediate solutions (Wolsey, 1998). On the variables with coefficients, this gap may become larger and cause

4 https://www.gurobi.com/

longer solving times in comparison to the positive weighted case in which no coefficient exists.

B. Observations on the impact of FCA is fourfold:

I. The best enhancement is achieved for the class of CPA-applicables (the first column), where a discrete sharp point in the curve is observed. The reason, as analyzed earlier, is the order of the ILP size reduction in case is linear to the number of aggregated constraints.

II. Among the rest of the domain, the sets of clauses with lower aggregation orders, starting with aggregation order of 2 (Quadratic Aggregation applicable) receive more enhancements. This is obviously due to more symmetric clauses and is the justification in preferring lower orders of aggregation in our target selection heuristic.

III. In high-symmetric clauses with low aggregation orders, the disorder caused by evidence in efficacy of FCA is ignorable in contrast to the conventional lifting methods. This is resulted from the novel view of symmetry in aggregation which has been discussed earlier.

IV. As the aggregation order reaches near the clauses length (the identical part vanishes), FCA loses its advantage in negative weights, and the meaningful gap between the evidence-free curve and other curves becomes observable (following that of the TRAD curves), yet surprisingly, positive weighted clauses proceed receiving enhancements from aggregation. In other words, aggregation preserves the efficiency in the presence of evidence even for low-symmetric positive-weighted target sets. This difference could be caused by the dissimilar forms of aggregation rules for negative and positive weights. Aggregation on positive weights is straightforward and results in a classical maximization problem, while on negative weights it leads to minimax multi-objective optimization which potentially requires more complicated steps to solve, especially when the gap between multiple objectives rises (Aissi et al., 2009).

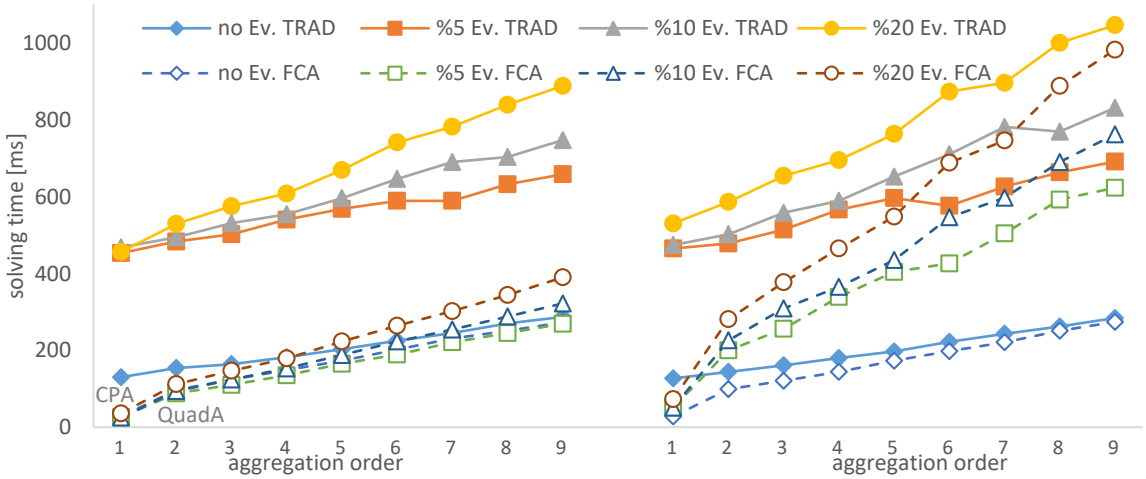

Figure 1: ILP solving times (average on 5 runs) of TRAD and FCA on sets of 100000 ground clauses for positive (left) and negative (right) weights of length 10 with various portion of random evidence and aggregation orders

We also considered the FCA enhancement for a fixed order of aggregation on various clause lengths. Figure 2 shows the average solving times of ILPs produced from Quadratic Aggregation (QuadA) compared to TRAD. In addition to the previous observations, it is exemplified that keeping the aggregation

order fixed, the solving time for FCA remains near constant as the ground clauses get longer, but grows with clauses length in the traditional case. This means that FCA better exposes aggregation advantages while applied to longer clauses with symmetries.

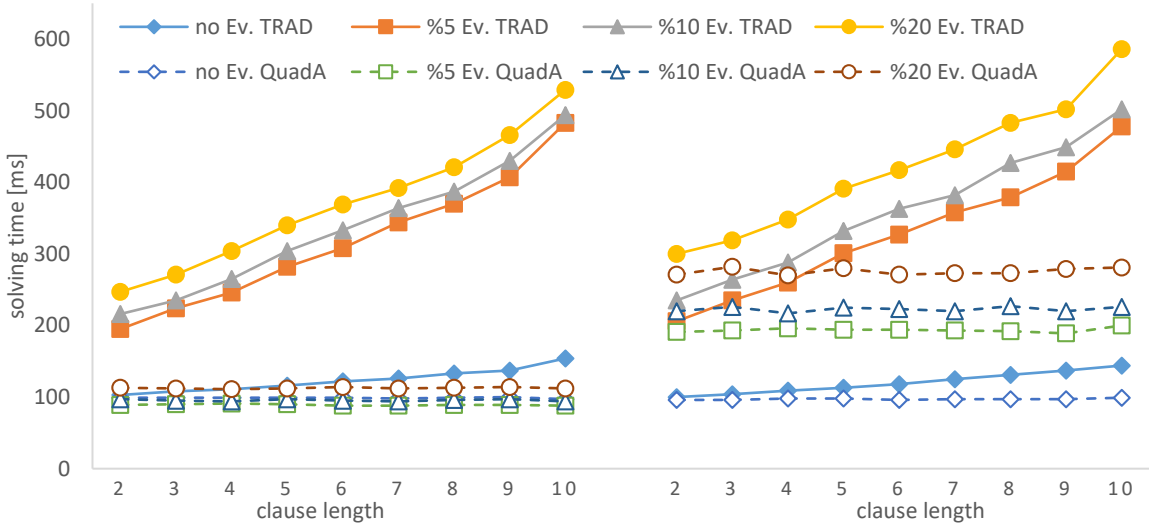

Figure 2: ILP solving times (average on 5 runs) of TRAD and QuadA on sets of 100000 ground clauses with aggregation order of 2 for positive (left) and negative (right) weights and various lengths and portions of random evidence

5.2 Benchmarks

As stated in previous sections, our MLN system performs MAP inference within three phases. We analytically and empirically showed improvements in the last phase compared to the traditional method and CPA. The overall process evaluation on multiple engineering applications is a further necessity. CPA was previously evaluated by Noessner et al. (2013) and reported to outperform other state-of-the-art inference algorithms on five established benchmark MLNs. The benchmarks included Relational Classification (RC), Link Prediction (LP), Information Extraction (IE), Protein Interaction (PR), and Entity Resolution (ER). These benchmarks except PR were also the test cases for evaluating the TUFFY system (Niu et al, 2011). RC was built for the classification problem on the CORA (McCallum, Nigam, Rennie, & Seymore, 2000) dataset. LP performs prediction of the relations holding between UW-CSE students, faculty, and staff (Richardson & Domingos, 2006). IE (Poon & Domingos, 2007) extracts database records from parsed sources. PR contains information on the yeast protein location, function, class, phenotype, and enzymes, from the MIPS (Munich Information center for Protein Sequence) Comprehensive Yeast Genome Database, as of February 2005 (Mewes et al., 2000). We used the MLN version (Davis & Domingos, 2009) which addresses the problem of predicting interactions between proteins, enzymes, and phenotypes. ER is used to find records corresponding to the same real-world entity (Singla & Domingos, 2006).

In order to show empirically to what extent FCA preserves efficiency, we compare FCA with CPA and TRAD with various configurations on these benchmarks and in addition, on the EKAW dataset whose confidence values were gathered from crowd-sourcing services to be used in ontology learning (Noessner, 2014; Niepert, Noessner, & Stuckenschmidt, 2011). The six benchmarks altogether cover a wide variety of MLNs in different aspects. Table 1 summarizes their characteristics.

Table 1: Characteristics of the six benchmark MLNs

characteristic	RC	LP	IE	PR *	ER *	EKAW
# predicates	4	22	18	7	10	21
# formulas	17	24	1,024	2,460/1,689	1,268/1,265	21
# evidence atoms	99,161	1,031	258,079	12,999	12,892	10,987
# clauses	202,215	354,587	340,737	40,234,321	390,720	2,922,601
# 1st ord. aggr. cnstr.	261,871	352,190	269,817	3,145,198/ 1,507,573	422,959/ 280,011	301
# 2nd ord. aggr. cnstr.	0	0	0	0/0	0/32,407	11,171
# 3rd ord. aggr. cnstr.	0	0	0	0/0	0/0	391,235

* multi-valued characteristics refer to simplified/not simplified long formulas with negative weights

5.3 Experimental Setup

We implemented the FCA algorithm in XEGGORA as an evolutionary extension of ROCKIT, which is invoked as the inference engine on each benchmark. We release the code as an open source project for further investigation[5]. In order to cancel the measurement error caused by the other processes running in parallel, all runs shorter than 10 minutes are repeated five times and runtimes are obtained on the average. In addition to the solver's gap bound parameter, we set its time limit to stop optimization if the gap is not reached in 10 minutes. For CPI, this limit is applied separately in each iteration thus the overall inference process may last much longer. As an external observer, TUFFY is also invoked on all benchmarks. It leverages RDBMS for grounding and performs MaxWalkSAT for MAP inference. Out of the MAP solution of each MLN, we evaluate the objective of a unified ILP equivalent to its ground model. By linear scaling of the objective, we obtain a *grade* for each approximate solution (displayed in parenthesis) as a criterion for analyzing its quality; a naïve empty instance (assigning false value to all query atoms) is graded 0, while the TUFFY output solution with default parameters is graded 100. TUFFY typically achieves higher quality solutions but in longer runtimes. Two exceptions are for the LP and IE benchmarks where some ILP approaches reach better approximate solutions (i.e., are graded above 100) on LP and the exact solution on IE.

5.4 Results

Experiments show different treatments of our approach on the benchmarks which can be discussed in three categories. The first category consists of RC, LP, and IE. Table 2 shows the comparison results on these benchmarks. We applied a 0.01 gap for LP while the two other models were solved exactly. Each method was firstly tested alone and then jointed with CPI.

Table 2: ILP/total Inference runtimes with various configurations on RC, LP, and IE benchmarks [ms]

benchmark	TRAD	CPA	FCA	TRAD +CPI	CPA +CPI	FCA +CPI	Tuffy
RC	2,166 / 6,015	**746** / **4,479**	918 / 5,697	12,348 / 15,671	4,642 / 7,645	5,572 / 9,518	– / 21,563
LP	9,309 / 11,487 (86.2)	**1,285** / **3,045** (99.3)	1,364 / 3,343 (82.5)	81,235 / 84,014 (107.3)	5,927 / 8,383 (106.5)	3,369 / 5,889 (106.0)	– / 230,630 (100)
IE	1,975 / 21,359	949 / 19,751	1,049 / 20,627	53 / **18,286**	29 / 18,397	**26** / 18,484	– / 14,400 *

* failed to reach the exact solution even in 40 seconds and by testing various maxFlips values

5 The source code is available at https://github.com/amirian/xeggora.

The runtime values of TRAD and CPA verify the experimental results reported by Noessner et al. (2013)[6]. The minimum value in each row is shown bolded. FCA executes in close runtimes to CPA and both of them run faster than TRAD. The reason is that FCA operates on sets of clauses with higher aggregation orders; in the absence of such sets, FCA behaves the same as CPA. Among these benchmarks, the IE inference runtimes are not much affected by the method because the optimization phase constitutes a small portion (0.14% - 9.2%) of the overall runtime.

The second category includes the EKAW ontology. We tested FCA and its competitors on an EKAW MLN with 10,987 evidence atoms. The MLN is the only test case for aggregation order of 3. It contains 8 sets of ground clauses with aggregation order of 1, 1105 sets of ground clauses with aggregation order of 2, and additionally 5 sets of ground clauses with aggregation order of 3 which totally result in aggregating more than 400,000 hard clauses in each CPI iteration. Without applying CPI, all methods ran out of memory in the grounding phase. The inference runtimes of different methods with various gap values are listed in Table 3. This category demonstrates a rare paradigm in which aggregation has more cons than pros. However, the FCA progressive selection strategy of aggregation targets prevents the runtime blow-up as that of CPA's. CPA shows poor results on EKAW, meanwhile FCA retains the higher solution quality and better runtimes of TRAD as we reduce the gap bound.

Table 3: ILP/total Inference runtimes with various gaps on EKAW benchmark [sec]

gap	TRAD+CPI	CPA+CPI	FCA+CPI	Tuffy
0.01	$\mathbf{\frac{504}{513}}(86.99)$	$\frac{1{,}750}{1{,}770}(86.65)$	$\frac{976}{994}(87.04)$	$\frac{-}{25{,}415}(100)$
0.001	$\mathbf{\frac{740}{748}}(87.35)$	*	$\frac{1{,}003}{1{,}014}(87.35)$	
0.0005	$\mathbf{\frac{1{,}531}{1{,}538}}(87.36)$	*	$\frac{1{,}892}{1{,}903}(87.36)$	

* failed to reach the gap within the time limit in one CPI iteration.

The PR benchmark appeared to provide similar results by FCA and CPA in the primary experiment with an improvement compared to TRAD, hence categorized in the first group; FCA and CPA both run in near half overall runtime of TRAD. The number of first-order aggregated constraints exceeds 3,000,000. By default, a formula simplification pre-processing option in ROCKIT splits long formulas with negative weights into shorter parts and the corresponding weights are divided between the parts. This simplification procedure is traditionally used by Richardson and Domingos (2006) when a conjunction of literals is present in the model. TUFFY also performs internal transformations to obtain clauses only with positive weights (Niu et al., 2011). It can be shown that the simplification leads to different probabilities of the same possible world (Noessner, 2014); hence the final solution would be approximated. Moreover, having long formulas broken, high orders of aggregation may disappear in the model and consequently, FCA may expose no benefit.

We repeated all experiments on PR ignoring the simplification step which resulted in an intractable inference. Runtimes are listed in Table 4. All executions without applying CPI ran out of memory in the grounding phase and are not included. The non-simplified MLN, providing near 1,500,000 first-order aggregated constraints fits in the second category, i.e., FCA affords no enhancement compared to

6 One exception is for the TRAD+CPI configuration on LP benchmark; a runtime enhancement for this configuration was reported, but our experiments with various gaps and various Gurobi versions resulted in the nearest case a minimum of 30% additive runtime compared to TRAD.

TRAD but prevents inheritance of CPA's runtime blow-up. However, the results show that the simplification should be accomplished for PR; otherwise the ILP methodology actually drops behind MaxWalkSAT of TUFFY in both quality and runtime aspects.

Table 4: ILP/total Inference runtimes on PR benchmark [sec]

configuration	TRAD+CPI	CPA+CPI	FCA+CPI	Tuffy
simplified	$\frac{99}{154}(21.2)$	$\frac{40}{\mathbf{85}}(21.2)$	$\frac{\mathbf{38}}{91}(21.2)$	$\frac{-}{1{,}161}(100)$
non-simplified (gap=1)	$\frac{4{,}039}{4{,}124}(36.0)$	*	$\frac{\mathbf{4{,}028}}{\mathbf{4{,}123}}(36.0)$	

* failed to reach the gap within the time limit in one CPI iteration.

The last category consists of the ER benchmark on which the simplification step again takes effect. The simplified MLN provides 1,612 sets of ground clauses with aggregation order of 1, resulting in more than 420,000 aggregated constraints, as both FCA and CPA run in near half runtime of TRAD (similar to that of the first category). It is remarkable that no better approximation for the original model is reachable in this way because the simplified model has gained its optimum solution. From the non-simplified MLN, FCA provides about 100,000 fewer aggregated constraints by 3,030 sets with aggregation order of 1 and 967 sets with aggregation order of 2, but with a meaningful runtime enhancement compared to CPA and TRAD; FCA facilitates obtaining high quality approximation (grade 99.8) in a short time. Results are shown in Table 5. An excessive competition including 16-minutes time limit (that of TUFFY's duration) was also held to compare between the gained objectives; none of the methods were graded 100, i.e., reached the objective of TUFFY's solution, yet FCA did the best. CPI-jointed methods were not included because the deadline is not applicable to the whole multi-iteration process.

Table 5: ILP/total Inference runtimes with various gaps on ER benchmark [sec]

configuration	TRAD	CPA	FCA	TRAD +CPI	CPA +CPI	FCA +CPI	Tuffy
simplified	$\frac{2.3}{5.4}(98.3)$	$\frac{\mathbf{0.35}}{\mathbf{2.6}}(98.3)$	$\frac{0.4}{3.7}(98.3)$	$\frac{3.3}{7.2}(98.3)$	$\frac{0.6}{3.6}(98.3)$	$\frac{0.6}{4.0}(98.3)$	$\frac{-}{960}(100)$
non-simplified (gap=0.3)	$\frac{14}{16}(99.6)$	$\frac{20}{22}(98.6)$	$\frac{\mathbf{5.2}}{\mathbf{7.2}}(98.5)$	*	*	*	
non-simplified (gap=0.2)	$\frac{581}{584}(99.9)$	$\frac{20}{22}(99.6)$	$\frac{\mathbf{5.4}}{\mathbf{7.4}}(99.8)$	*	*	*	
960-seconds competition	(99.84)	(99.72)	(**99.86**)	–	–	–	

* failed to reach the gap within the time limit in one CPI iteration.

It can be observed from the results that with increasing the number of higher-order aggregated constraints, FCA shows its role of reducing inference runtime, yet retaining the optimality. This happens by ignoring the simplification pre-process on the ER benchmark, hence allowing long formulas to take place in the model.

6. Related Works

A vast line of works with connections to ours has been done on symmetry exposition and exploitation in SRL. Bui et al. (2013) propose a general algebraic lifting framework based on the group theory and

graph isomorphism for variational inference. Kersting, Mladenov, and Tokmakov (2017) define a syntax for relational domains of linear programs and propose its translation into an equivalent Lifted Linear Program (Mladenov, Ahmadi, & Kersting, 2012). By computing the Coarsest Equitable Partition (CEP) for the coefficient graph of the LP, they provide a smaller lifted LP which recovers an exact optimal solution of the original LP. Mladenov, Globerson, and Kersting (2014) argue that despite the enhancements from lifting a MAP LP, existing message passing solvers such as MPLP and Tree-ReWeighted Belief Propagation (TRW-BP) will not work on the lifted version because it contains constraints that do not conform to the MAP LP template. Imposing ignorable overhead, they find a small Markov Random Field with a ground optimal value equal to the value of the lifted program and produce an equivalent LP conforming to the required template. Bui, Huynh, and Sontag (2014) implement the idea of Bui et al. (2013) for marginal inference using the Kruskal's algorithm and lifting TRW problem again with CEP. Mladenov and Kersting (2015) also apply equitable partitions to lift the message passing algorithm for marginal inference with the ability to distribute and parallelize inference computations. Mladenov, Kersting, and Globerson (2014) suggest a different way of finding symmetries and lifting MAP LPs compared to Lifted Linear Programs, considering higher-order neighborhood to reach tighter approximation. To find finer partitions, they apply the k-dimensional Weisfeiler-Lehman – an algorithm previously used for approximating the graph automorphism problem (Cai, Fürer, & Immerman, 1992) – to the original graphical model instead of the larger graph of the LP. Mladenov, Kleinhans, and Kersting (2017) extend the idea of relational linear programming (Kersting et al., 2017) over convex quadratic programs and provide the relational version.

A common deficiency of Linear (or Quadratic) Programming lifting approaches for MAP inference in SRL rises from the fact that MAP inference is equivalent to solving Integer Programs. If solving any relaxed IP gives the exact solution, which in turn would lead to an approximation of the underlying IP, the lifted version would also provide an exact solution, say, with fewer computations. However, it will not go farther for more tightening the approximation. We showed in contrast that as aggregation is performed before program relaxation, it can result in tighter approximation for the corresponding ILP. Moreover, modern IP solvers involve much more than just solving the LP relaxation of the given ILP. Lifting the mathematical program before relaxation allows speeding up pre-processing techniques such as exploiting integrality requirements and primal/dual dominance arguments and even program re-formulation to be applied to the compact lifted version. Similar to our idea, Sarkhel et al. (2014b) translate the MAP inference problem to an Integer Polynomial Program, but by adapting the theorem proving technique (Gogate & Domingos, 2011) and removing its search and conditioning steps. They define each variable in the program to represent an assignment to a set of indistinguishable variables. The integer polynomial program is further converted to an ILP with classic linearization methods with no relaxation except the LP relaxation by the ILP solver. In contrast, we perform an extra relaxation of the upper bound or lower bound constraints in the aggregation rules.

Another major challenge in the area of lifted inference is examining the role of evidence, which is rarely targeted in the symmetry exposition. Apsel, Kersting, and Mladenov (2014) discuss the high cost of obtaining symmetry with automorphism groups. To solve, they offer a costly graph which can be generated once and used multiple times separately from the problem model, concluding with curves that confirm the evidence problem. Venugopal and Gogate (2014a) propose a heuristic similarity function on groundings to apply K-Means clustering and perform approximation on the evidence in order to increase the number of symmetries. Venugopal and Gogate (2014b) utilize the approach to build an informed distribution for importance sampling. Bui, Huynh, and de Salvo Braz (2012) consider the

models with symmetries that are broken by the evidence. They show that exact lifted inference based on distinct soft evidence consisting only of unary predicates can be done in polynomial time. Van den Broeck and Niepert (2015) also provide a solution in the presence of distinct soft evidence for marginal inference approximation by lifting Metropolis Hastings at the propositional level on asymmetric graphical models where symmetry is a hard condition. Another approach considering evidence on unary predicates is taken by Habeeb, Anand, Mausam, and Singla (2017) who introduce an application specific approximate lifted MAP inference. They adapt the general color passing algorithm for symmetry detection by Kersting et al. (2009) in a way to make effective partitioning in the presence of evidence. Van den Broeck and Darwiche (2013) extend the results of Bui et al. (2012) and point out the hardness of dealing with binary evidence. They show it can be done efficiently if there is a corresponding low rank Boolean Matrix Factorization (BMF). The evidence approximation scheme is further proposed to reduce the rank. As an important drawback of dealing with distinct soft evidence and the corresponding BMF, all objects with evidence go beyond closed-world assumption, i.e., each predicate is assumed either hidden or fully observed. Mladenov, Globerson, and Kersting (2014) also demonstrate that the cost of evidence rises in their lifting approach with an increase in the Boolean rank. In contrast to Bui et al. (2013) which introduce symmetries on unobserved constants, Kopp, Singla, and Kautz (2015) explicitly find symmetries among constants that appear in the evidence. Another schema alongside the evidence is the notion of context (variable-value assignment). Contextual symmetry is introduced which allows for exploitation of count (Anand, Grover, Mausam, & Singla, 2016) and non-count (Anand, Noothigattu, Singla, & Mausam, 2017) symmetries using graph isomorphism in the MCMC framework.

Addressing the evidence problem comes with building the notion of symmetry given a specific form of evidence in some of the above approaches, while partly manipulating (i.e., approximating) evidence in the others. All methods are subject to alteration with any change in the evidence (e.g. via incremental perception) or its structure (e.g. by adding evidence for unobserved predicates). To our knowledge, CPA (Noessner et al., 2013) is the only approach that exploits symmetries independently of the notion of evidence. In contrast to CPA which will only aggregate clauses that differ by a single literal, our higher-order aggregation method can aggregate clauses with an arbitrary number of different literals. FCA, as an additional facility, retains all benefits of the CPA approach, such as parallelization, control on the error bound, the ability of merging with orthogonal approaches, and leveraging several state-of-the-art optimization and data mining techniques.

Finally, a single approach that deals with long formulas, follows the idea of *pairwise MLNs* (Fierens et al., 2013), performing inference through *quadratization* for pseudo-Boolean functions by the means of first-order *slack predicates* (de Nijs, Landsiedel, Wollherr, & Buss, 2016). It produces a new model with *quadratic parfactors*, at the cost of additional optimization over slack variables and the benefit of better bounds and more persistencies. Similar to our approach, it reports higher quality approximation and shorter inference runtime compared to state-of-the-art inference engines on models with long formulas. Unfortunately, no source or executable code as well as benchmarks for testing and comparing the methods are made public. Moreover, the configurations during experiments are not described in detail, ceasing the option of an estimated comparison by performing similar tests with XEGGORA. Among the common benchmarks, ER is however the only test case on which its significant runtime enhancement compared to ROCKIT is reported, reducing the inference runtime from 1902 seconds to 101 seconds. Our experiments show this can be also verged by FCA, setting various gaps in $(0.1\ ,\ 0.2)$, where FCA runs 10 times faster than CPA on average.

7. Conclusion and Future Directions

Dealing with long formulas has been a challenging task in lifted inference. CPA is a symmetry exploiting algorithm for inference in Markov Logic which is widely used in real-world applications. It exposes first-order aggregate-able sets of ground clauses. We showed that in addition to the aggregation of order one, there often exist higher aggregation orders in MLNs with long formulas that are ignored by CPA. We introduced FCA, a superior meta-algorithm that is empowered with any possible orders of aggregation with respect to the model. We proposed complementary techniques within the FCA algorithm, including heuristics and RDBMS leverage to choose efficiently among multiple candidates the one in which aggregation fits the best. We implemented the proposed methods within an evolutionary extension of ROCKIT, namely XEGGORA, besides fixing minor bugs of the previous system and supporting numerical constraints (Chekol et al., 2016). Finally, we proved enhancements with comprehensive experiments upon two benchmark sets. The first set of artificial benchmarks explores the effect of higher order aggregation as compared to the first order aggregation with respect to the length of clauses. The results confirm the intuition developed through the theoretic side. On practical benchmarks for the entire system, it is shown that FCA retains the enhancements of previous works on instances where higher order aggregation is not applicable, but provides a large benefit in instances where it is.

Future directions consist of four extensions. One is to define a better ILP encoding for negative weighted clauses whereas aggregation brings less efficiency as shown by the analysis on the artificial datasets. Another direction is to adapt the conventional lifting LP methods to lift ILPs as well, before any relaxation, and see if this provides any time savings or tighter approximation. A third direction would be to develop techniques to make symmetry exploitation even more adaptive. For instance, whenever a set of ground clauses is clustered to expose symmetries, the partitioning of each cluster to identical and distinct parts can be different and independent of the other clusters. FCA treats all clusters the same. Fixing the partitions for all clusters won't promise best constraint aggregation.

Finally, an extension may be to generalize the constraint aggregation technique for marginal inference. Marginal inference for MLNs with long formulas is also a challenging purpose for several applications including natural language semantics inference (Beltagy & Mooney, 2014) and automatic question answering (Khot et al., 2015). Researchers already try to avoid heavy process on long formulas by either splitting them into shorter parts, or manipulating an inference engine in a way to especially solve their precise problem. A general-purpose technique for inferring such MLNs would be a promising direction.

Acknowledgements

We would like to thank the anonymous reviewers for their comments and helpful suggestions.

References

Aissi, H., Bazgan, C., & Vanderpooten, D. (2009). Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European journal of operational research*, 197(2), 427-438.

Anand, A., Grover, A., Mausam, & Singla, P. (2016). Contextual symmetries in probabilistic graphical models. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 3560-3568.

Anand, A., Noothigattu, R., Singla, P., & Mausam. (2017). Non-Count Symmetries in Boolean & Multi-Valued Prob. Graphical Models. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, in PMLR*, 54, 1541-1549.

Apsel, U., Kersting, K., & Mladenov, M. (2014). Lifting Relational MAP-LPs Using Cluster Signatures. In *Proceedings of AAAI*, 2403-2409.

Beltagy, I., & Mooney, R. J. (2014). Efficient Markov Logic Inference for Natural Language Semantics. In *Proceedings* of *the 13th AAAI Conference on Statistical Relational AI*, 9-14.

Bui, H. H., Huynh, T. N., & de Salvo Braz, R. (2012). Exact lifted inference with distinct soft evidence on every object. In *Proceedings of AAAI*, 1875-1881.

Bui, H. H., Huynh, T. N., & Riedel, S. (2013). Automorphism groups of graphical models and lifted variational inference. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, 132-141.

Bui, H. H., Huynh, T. N., & Sontag, D. (2014). Lifted Tree-Reweighted Variational Inference. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 92-101.

Cai, J. Y., Fürer, M., & Immerman, N. (1992). An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4), 389-410.

Chekol, M. W., Huber, J., Meilicke, C., & Stuckenschmidt, H. (2016). Markov logic networks with numerical constraints. *Frontiers in artificial intelligence and applications*, 285, 1017-1025.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4), 393-410.

Davis, J., & Domingos, P. (2009). Deep transfer via second-order Markov logic. In *Proceedings of the 26th International Conference on Machine Learning*, 217-224.

de Nijs, R., Landsiedel, C., Wollherr, D., & Buss, M. (2016). Quadratization and Roof Duality of Markov Logic Networks. *Journal of Artificial Intelligence Research*, 55, 685-714.

de Salvo Braz, R., Amir, E., & Roth, D. (2006). MPE and partial inversion in lifted probabilistic variable elimination. In *Proceedings of AAAI*, 1123-1130.

Fierens, D., Kersting, K., Davis, J., Chen, J., & Mladenov, M. (2013). Pairwise Markov Logic. In *Postproc. of the 22nd International Conference on Inductive Logic Programming, ILP 2012*, 58-73.

Gogate, V., & Domingos, P. (2011). Probabilistic theorem proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 256-265.

Habeeb, H., Anand, A., Mausam, & Singla, P. (2017). Coarse-to-Fine Lifted MAP Inference in Computer Vision. In Proceedings *of the 26th International Joint Conference on Artificial Intelligence*, 4595-4602.

Huynh, T. N., & Mooney, R. J. (2009). Max-margin weight learning for Markov logic networks. In *Proceedings of the European Conference on Machine Learning and Practice of Knowledge Discovery in Databases*, 564-579.

Kautz, H., Selman, B., & Jiang, Y. (1997). A general stochastic approach to solving problems with hard and soft constraints. In Gu, D., Du, J., & Pardalos, P. (Eds.), *The Satisfiability Problem: Theory and Applications*, 573-586. *American Mathematical Society, New York, NY*.

Kazemi, S. M., Kimmig, A., Van den Broeck, G., & Poole, D. (2016). New liftable classes for first-order probabilistic inference. In *Advances in Neural Information Processing Systems*, 3117-3125.

Kersting, K., Ahmadi, B., & Natarajan, S. (2009). Counting Belief Propagation. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 277-284.

Kersting, K., Mladenov, M., & Tokmakov, P. (2017). Relational linear programming. *Artificial Intelligence*, 244, 188-216.

Khot, T., Balasubramanian, N., Gribkoff, E., Sabharwal, A., Clark, P., & Etzioni, O. (2015). Exploring Markov logic networks for question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 685-694.

Kimmig, A., Mihalkova, L., & Getoor, L. (2015). Lifted graphical models: A survey. *Machine Learning*, 99, 1-45.

Kisynski, J., & Poole, D. (2009). Lifted aggregation in directed first-order probabilistic models. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 1922-1929.

Kopp, T., Singla, P., & Kautz, H. (2015). Lifted symmetry detection and breaking for MAP inference. In *Advances in Neural Information Processing Systems*, 1315-1323.

Lee, J., Talsania, S., & Wang, Y. (2017). Computing LP MLN using ASP and MLN solvers. *Theory and Practice of Logic Programming*, 17, 942-960.

Lüdtke, S., Schröder, M., Krüger, F., Bader, S., & Kirste, T. (2018). State-space abstractions for probabilistic inference: a systematic review. *Journal of Artificial Intelligence Research*, 63, 789-848.

Mangal, R., Zhang, X., Kamath, A., Nori, A. V., & Naik, M. (2016). Scaling relational inference using proofs and refutations. In *Proceedings of AAAI*, 3278-3286.

Martínez-Angeles, C. A., Dutra, I., Costa, V. S., & Buenabad-Chávez, J. (2016). Processing Markov Logic Networks with GPUs: Accelerating Network Grounding. In *Postproc. of the 25th International Conference on Inductive Logic Programming, ILP 2015*, 122-136.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2), 127-163.

Meilicke, C., Leopold, H., Kuss, E., Stuckenschmidt, H., & Reijers, H. A. (2017). Overcoming individual process model matcher weaknesses using ensemble matching. *Decision Support Systems*, 100, 15-26.

Meilicke, C., & Stuckenschmidt, H. (2015). A New Paradigm for Alignment Extraction. In *Proceedings of the 10th International Workshop on Ontology Matching, OM 2015*, 1-12.

Mewes, H. W., Frishman, D., Gruber, C., Geier, B., Haase, D., Kaps, A., Lemcke, K., Mannhaupt, G., Pfeiffer, F., Schüller, C., Stocker, S., & Weil, B. (2000). MIPS: a database for genomes and protein sequences. *Nucleic acids research*, 28(1), 37-40.

Milch, B., Zettlemoyer, L. S., Kersting, K., Haimes, M., & Kaelbling, L. P. (2008). Lifted probabilistic inference with counting formulas. In *Proceedings of AAAI*, 1062-1068.

Mladenov, M., Ahmadi, B., & Kersting, K. (2012). Lifted linear programming. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics, in PMLR*, 22, 788-797.

Mladenov, M., Globerson, A., & Kersting, K. (2014). Lifted Message Passing as Reparametrization of Graphical Models. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 603-612.

Mladenov, M., & Kersting, K. (2015). Equitable Partitions of Concave Free Energies. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, 602-611.

Mladenov, M., Kersting, K., & Globerson, A. (2014). Efficient Lifting of MAP LP Relaxations Using k-Locality. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics, in PMLR*, 33, 623-632.

Mladenov, M., Kleinhans, L., & Kersting, K. (2017). Lifted Inference for Convex Quadratic Programs. In *Proceedings of AAAI*, 2350-2356.

Niepert, M., Noessner, J., & Stuckenschmidt, H. (2011). Log-linear description logics. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2153-2158.

Niu, F., Ré, C., Doan, A., & Shavlik, J. (2011). Tuffy: Scaling up statistical inference in Markov Logic Networks using an RDBMS. In *Proceedings of the VLDB Endowment*, 4(6), 373-384.

Noessner, J. (2014). *Efficient Maximum A-Posteriori Inference in Markov Logic and Application in Description* Logics. Ph.D. thesis, University of Mannheim.

Noessner, J., Niepert, M., & Stuckenschmidt, H. (2013). RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models. In *Proceedings of AAAI*, 739-745.

Poole, D. (2003). First-order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 985-991.

Poole, D., Bacchus, F., & Kisynski, J. (2011). Towards completely lifted search-based probabilistic inference. In *CoRR*, abs/1107.4035.

Poon, H., & Domingos, P. (2007). Joint inference in information extraction. In *Proceedings of AAAI*, 913-918.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62, 107-136.

Riedel, S. (2008). Improving the accuracy and efficiency of MAP inference for Markov Logic. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, 468-475.

Singla, P., & Domingos, P. (2006). Entity resolution with Markov Logic. In *Proceedings of the 6th IEEE International Conference on Data Mining*, 572–582.

Singla, P., & Domingos, P. (2008). Lifted first-order belief propagation. In *Proceedings of AAAI*, 1094-1099.

Sarkhel, S., Venugopal, D., Singla, P., & Gogate, V. (2014a). Lifted MAP inference for Markov Logic Networks. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics, in PMLR*, 33, 859-867.

Sarkhel, S., Venugopal, D., Singla, P., & Gogate, V. (2014b). An integer polynomial programming based framework for lifted MAP inference. In *Advances in Neural Information Processing Systems*, 3302-3310.

Schoenfisch, J., Meilicke, C., von Stülpnagel, J., Ortmann, J., & Stuckenschmidt, H. (2018). Root cause analysis in IT infrastructures using ontologies and abduction in Markov Logic Networks. *Information Systems*, 74, 103-116.

Van den Broeck, G., & Darwiche, A. (2013). On the complexity and approximation of binary evidence in lifted inference. In *Advances in Neural Information Processing Systems*, 2868-2876.

Van den Broeck, G., & Niepert, M. (2015). Lifted Probabilistic Inference for Asymmetric Graphical Models. In *Proceedings of AAAI*, 3599-3605.

Venugopal, D., & Gogate, V. (2014a). Evidence-based clustering for scalable inference in Markov Logic. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 258-273.

Venugopal, D., & Gogate, V. (2014b). Scaling-up importance sampling for Markov Logic Networks. In *Advances in Neural Information Processing Systems*, 2978-2986.

Visengeriyeva, L., Akbib, A., & Kaul, M. (2016). Improving Data Quality by Leveraging Statistical Relational Learning, In *Proceedings of the 21st International Conference on Information Quality*, 220-236.

Watters, L. J. (1967). Reduction of Integer Polynomial Programming Problems to Zero-One Linear Programming Problems. *Operations Research*, 15, 1171-1174.

Wolsey, L. A. (1998). *Integer Programming*. Wiley-Interscience, New York.