Pairwise Markov Logic

Daan Fierens[1], Kristian Kersting[23], Jesse Davis[1], Jian Chen[1], and Martin Mladenov[3]

[1] Dept. of Computer Science, KULeuven, Belgium
fierens.daan@gmail.com
jesse.davis@cs.kuleuven.be
[2] Fraunhofer IAIS, Germany
kristian.kersting@iais.fraunhofer.de
[3] University of Bonn, Germany
mladenov@igg.uni-bonn.de

Abstract. For many tasks in fields like computer vision, computational biology and information extraction, popular probabilistic inference methods have been devised mainly for propositional models that contain only unary and pairwise clique potentials. In contrast, statistical relational approaches typically do not restrict a model's representational power and use high-order potentials to capture the rich structure of relational domains. This paper aims to bring both worlds closer together.
We introduce pairwise Markov Logic, a subset of Markov Logic where each formula contains at most two atoms. We show that every non-pairwise Markov Logic Network (MLN) can be transformed or 'reduced' to a pairwise MLN. Thus, existing, highly efficient probabilistic inference methods can be employed for pairwise MLNs without the overhead of devising or implementing high-order variants. Experiments on two relational datasets confirm the usefulness of this reduction approach.

1 Introduction

In the probabilistic graphical models literature, many inference algorithms have been designed specifically for *low-order models*. The term *order* is used here in the probabilistic sense (as opposed to the logical sense of 'first-order' logic). Concretely, the order of a factor graph is defined as the maximum number of arguments of a factor [1]; the order of a Markov Random Field (MRF) is the size of the largest clique. A probabilistic model is called *pairwise* if its order is (at most) two.

Pairwise graphical models first became popular in the field of statistical physics and are nowadays used for many applications, in diverse fields like computer vision, computational biology and information extraction [1]. The typical 'pairwise approach' is to write a probabilistic model with unary and pairwise clique potentials, understand the Bayesian priors it incorporates, and then perform inference [2]. This is especially common in the case of *MAP inference*. MAP is the task of, given the observed state of some random variables, finding the

most likely state of all other random variables occurring in the model.[4] Many state-of-the-art methods for MAP inference were mainly developed for pairwise models. High-order variants of such methods (capable of handling non-pairwise models) often do not exist and if they do, they are typically more complex and lack implementations. This is particularly true for MAP methods based on Linear Programming [3], Quadratic Programming [4], graph cuts [5, 6], etc. While most of these methods can in principle work on non-pairwise models, pairwise models receive the most attention because this facilitates implementation and theoretical analysis (e.g., convergence analysis for iterative methods like belief propagation [3]). Table 1 provides an overview of some inference methods and their ability to handle non-pairwise models.

Table 1. An (incomplete) overview of MAP inference methods and their support for non-pairwise models.

Method	Supports non-pairwise models
Max Product Variable Elimination	✓
Max Product Belief Propagation	✓
Linear programming methods [3]	✓/−
Graph cut methods [6] (QPBO [5], ...)	−
Quadratic programming [4]	−
...	

In statistical relational learning (SRL), the situation is different. Typical SRL approaches, like Markov Logic [7], do not restrict a model's representational power and use high-order potentials to capture the rich structure of relational domains. The notion of pairwise models has so far not been considered in the SRL literature. With this paper, we aim to bridge this gap between SRL and probabilistic graphical models. To this end, we introduce *pairwise Markov logic*. We show that any high-order Markov logic network can be reduced to a model in pairwise Markov logic, where we can then bring to bear the powerful inference techniques for low-order models. Section 3 explains the reduction for propositional MLNs and Section 4 explains the lifted reduction for first-order MLNs. Section 5 empirically demonstrates the utilty of the reduction approach for both ground and lifted inference on some common relational datasets.

2 Pairwise Markov Logic

A Markov Logic Network (MLN) is a set of pairs (φ_i, w_i), where φ_i is a formula in first-order logic and w_i a real-valued weight. Following the propositional case,

[4] This is sometimes also called *MPE (Most Probable Explanation)* inference, or *full MAP* inference (to distinguish it from *partial MAP*, in which some of the random variables need to be summed out).

we define the *order* of an MLN as the maximum number of atoms in a formula, i.e., the maximum 'length' of a formula in the MLN. We call an MLN (or a single MLN formula) *pairwise* if its order is two. For instance, the formula $Smokes(x) \Rightarrow Asthma(x)$ is pairwise, but $Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$ is not. We call the latter a *triplewise* formula since it has length three.

Pairwise MLNs have advantages for both ground and lifted inference. Ground inference typically applies methods from the graphical models literature, many of which focus on pairwise models. A similar argument holds for lifted inference, where some recent work uses graph-theoretical notions that assume that the network is pairwise [8] (as it can then be represented as a simple graph rather than a hypergraph).

Despite the advantages of pairwise MLNs, one should not discard non-pairwise MLNs. It is difficult, if not impossible, to capture the rich structure of many relational domains using a pairwise model. For example, when performing structure learning, restricting the hypothesis space to pairwise MLNs would simply ignore too many relevant patterns in relational datasets. Many typical relational patterns require triplewise formulas. Common examples are found in collective classification (e.g., $Class(x, c) \wedge Link(x, y) \Rightarrow Class(y, c)$), in link prediction (e.g., $Property(x) \wedge Property(y) \Rightarrow Link(x, y)$), in social networks (e.g., the above Smokes formula), etc.

In summary: we would like to use non-pairwise MLNs during modelling and learning but use only pairwise MLNs during inference. In the graphical models literature, this apparent contradiction is typically solved by *reducing* the non-pairwise model to an equivalent pairwise model when performing inference. In this paper we (for the first time) show that this can also be done for Markov Logic: any MLN can be reduced to a pairwise MLN.

3 Reduction for Propositional MLNs

A propositional MLN is a set of pairs (φ_i, w_i) where φ_i is a propositional logic formula (using connectives $\neg$, $\wedge$, $\vee$, $\Rightarrow$ and $\Leftrightarrow$) and $w_i \in \mathbb{R}$. A propositional MLN defines a probability distribution on the set of possible worlds (interpretations): the probability of a world ω is $P(\omega) = \frac{1}{Z} exp(\sum_i w_i \delta_i(\omega))$, with Z a normalization constant and $\delta_i(\omega)$ the indicator function being 1 if formula φ_i is true in world ω and 0 otherwise.

Below we show how to reduce a given propositional non-pairwise MLN to a pairwise MLN. In graphical models, reduction is typically done by converting the model (or its energy function) to a pseudo-Boolean function or multi-linear polynomial [1]. While our reduction for propositional MLNs can also be phrased in this terminology, we instead chose for a more self-contained formulation, referring only to Markov Logic. This will allow us to extend our approach from propositional MLNs to first-order MLNs (lifted reduction, Section 4).

3.1 Outline of the reduction algorithm

The reduction is done by means of a rewriting process that modifies the MLN knowledge base, i.e., the set of weighted formulas. This is a two-step process. The first step is an enabling step that brings the MLN into a certain normal form. The second step does the actual reduction to pairwise form.

Algorithm 1 provides an outline of the procedure, which we explain in detail in the following sections. We will also discuss in what sense the obtained pairwise MLN is 'equivalent' to the original MLN. To simplify the discussion, we first show how to reduce *triplewise* MLNs (i.e., with maximum formula length 3). Section 3.5 explains how to reduce MLNs with order higher than 3.

3.2 Step 1: write triplewise formulas in positive normal form (PNF)

Since we for now only consider triplewise MLNs, each formula in the MLN is either triplewise, pairwise or unary. The goal of Step 1 is to bring all triplewise formulas into what we call positive normal form.[5] This will simplify the actual reduction to pairwise form in Step 2. A formula is in *positive normal form* or *PNF* if it is a conjunction of atoms (not involving negation), e.g., $P \wedge Q \wedge R$.

Step 1 loops over all weighted formulas (φ, w) in the given MLN M. If φ is triplewise and not in PNF, we execute an *atomic rewriting step* on φ.

Atomic rewriting step. An atomic rewriting step executed on a non-PNF triplewise formula φ with weight w removes φ from the MLN and replaces it by a set of equivalent PNF formulas. In Algorithm 1a, this is denoted by the line $M := M \cup \mathit{RewriteToPNF}(\varphi, w) \setminus \{(\varphi, w)\}$. To do so, we call the function $\mathit{RewriteToPNF}(\varphi, w)$ (see Algorithm 1b) which returns an equivalent set of seven weighted PNF formulas that we add in place of φ in the MLN. The equations for the weights of these formulas, w_1 to w_7, depend on the truth table of φ, which is denoted $v_{...}$ in Algorithm 1b. Concretely, let P, Q and R be the atoms in φ, then v_{pqr} denotes the Boolean truth value (1 or 0) of φ under the interpretation $P = p$, $Q = q$ and $R = r$. For instance, v_{ttf} is the truth value of φ when P and Q are true and R is false. In general the output of $\mathit{RewriteToPNF}()$ consists of seven formulas, but in practice it is often the case that the weights of some of these formulas equal zero, so they can be omitted.

Example. Consider the triplewise non-PNF formula $P \wedge Q \Rightarrow R$ with weight w. This formula is satisfied unless P and Q are true and R is false. Hence, the truth table of this formula is: v_{ttf} equals 0, all other $v_{...}$'s equal 1. Applying the function $\mathit{RewriteToPNF}()$ to this formula, we get an equivalent set of two PNF formulas. The first formula is $P \wedge Q \wedge R$ with weight w (since $w_1 = w(1 - 0 - 1 + 1 - 1 + 1 + 1 - 1) = w$). The second formula is $P \wedge Q$ with weight $-w$ (since $w_2 = w(0 - 1 - 1 + 1) = -w$). The five other formulas get weight zero and hence can be omitted.

Equivalence of MLNs. Let M_0 be the original MLN with triplewise formulas and M_1 be the MLN obtained after applying Step 1 to M_0. M_0 and M_1 are

[5] Our use of the term positive normal form is unrelated to its use in other fields.

Algorithm 1 The algorithm for reducing a triplewise MLN: **1a)** outer loop, **1b)** Step 1, writing to positive normal form or 'PNF', **1c)** Step 2, reduction to pairwise form.
Lines in gray do not apply to the propositional reduction but only to the lifted reduction for first-order MLNs.

---

procedure *ReduceMLN*(M)
in: a triplewise MLN M

1a)

// Step 1: rewrite to PNF
for each triplewise weighted formula $(\varphi, w) \in M$ that is not in PNF
  $M := M \cup \mathit{RewriteToPNF}(\varphi, w) \setminus \{(\varphi, w)\}$ // an atomic rewriting step

// Step 2: reduce to pairwise
for each triplewise weighted (PNF) formula $(\varphi, w) \in M$
  $M := M \cup \mathit{ReduceToPairwise}(\varphi, w) \setminus \{(\varphi, w)\}$ // an atomic reduction step

---

function *RewriteToPNF*(φ, w)
in: a triplewise formula φ with weight w
  let P, Q and R denote the atoms in φ
  let $v_{\ldots}$ denote the truth table of φ (see text)

return $\{\ (w_1, P \wedge Q \wedge R),\ (w_2, P \wedge Q),\ (w_3, P \wedge R),$
$(w_4, Q \wedge R),\quad (w_5, P),\quad (w_6, Q),\quad (w_7, R)\ \}$
with

1b)

$$
\begin{aligned}
w_1 &= w(v_{ttt} - v_{ttf} - v_{tft} + v_{tff} - v_{ftt} + v_{ftf} + v_{fft} - v_{fff}) \\
w_2 &= w(v_{ttf} - v_{tff} - v_{ftf} + v_{fff}) \\
w_3 &= w(v_{tft} - v_{tff} - v_{fft} + v_{fff}) \\
w_4 &= w(v_{ftt} - v_{ftf} - v_{fft} + v_{fff}) \\
w_5 &= w(v_{tff} - v_{fff}) \\
w_6 &= w(v_{ftf} - v_{fff}) \\
w_7 &= w(v_{fft} - v_{fff})
\end{aligned}
$$

For the 2nd up to 7th formula (formulas with weights $w_2, \ldots, w_7$):
multiply the weight with a correction factor for lost logvars (see text)

---

function *ReduceToPairwise*(φ, w)
in: a triplewise PNF formula φ of the form $P \wedge Q \wedge R$ with weight w

Introduce a new auxiliary atom A Logvars of A:
union of all logvars in P, Q and R

1c)

if $w > 0$
  return $\{\ (w, A \wedge P),\ (w, A \wedge Q),\ (w, A \wedge R),\ (-2w, A)\ \}$
else
  return $\{\ (w, P \wedge Q),\quad (w, P \wedge R),\quad (w, Q \wedge R),$
  $(-w, A \wedge P),\ (-w, A \wedge Q),\ (-w, A \wedge R),\ (w, A)\ \}$

For the first three formulas ($P \wedge Q$, $P \wedge R$ and $Q \wedge R$):
multiply the weight with a correction factor for lost logvars (see text)

---

equivalent in the sense that they determine the same probability distribution over possible worlds. To prove this, we show that executing one atomic rewriting step preserves the distribution of the MLN. As Step 1 is just a sequence of atomic rewriting steps, it follows by transitivity of equality that M_1 is equivalent to M_0.

Recall that the probability of a world ω is $P(\omega) = \frac{1}{Z} exp(\sum_i w_i \delta_i(\omega))$ where $\sum_i w_i \delta_i(\omega)$ is the sum of weights of satisfied formulas in the world ω and Z is the normalization constant, i.e., $Z = \sum_\omega exp(\sum_i w_i \delta_i(\omega))$. Hence, any rewriting step that leaves the *sum of weights of satisfied formulas* or *SWSF* of every world unchanged, preserves the distribution of the MLN. In fact, this is also the case if the rewriting step adds a constant number to the SWSF of every world, since this constant will disappear into the normalization constant, i.e., the resulting MLN will have a different normalization constant but will still define the same distribution over possible worlds.

An atomic rewriting step only has a *local* effect: it replaces one triplewise non-PNF formula φ, involving only three atoms P, Q and R, by a set of at most seven new formulas involving P, Q and R. Because of this locality, the SWSF of each world remains largely unchanged under an atomic rewriting step: we only need to show that the contribution of the seven new formulas to the SWSF in each world is the same (up to a constant) as the contribution of φ. This can be seen from the following table over the possible worlds of P, Q and R.

P	Q	R	φ	new formulas
t	t	t	$w\ v_{ttt}$	$w(v_{ttt} - v_{fff})$
t	t	f	$w\ v_{ttf}$	$w(v_{ttf} - v_{fff})$
t	f	t	$w\ v_{tft}$	$w(v_{tft} - v_{fff})$
t	f	f	$w\ v_{tff}$	$w(v_{tff} - v_{fff})$
f	t	t	$w\ v_{ftt}$	$w(v_{ftt} - v_{fff})$
f	t	f	$w\ v_{ftf}$	$w(v_{ftf} - v_{fff})$
f	f	t	$w\ v_{fft}$	$w(v_{fft} - v_{fff})$
f	f	f	$w\ v_{fff}$	$w(v_{fff} - v_{fff})$

This table is read as follows. The first three columns determine the possible world ('t' is *true*, 'f' is *false*). The fourth column gives the contribution of the original formula φ to the SWSF for that world, which by definition is the weight w multiplied by either 1 or 0, depending on the truth value $v_{\ldots}$ of φ in the considered world. The last column gives the summed contribution of the seven new formulas to the SWSF, this can be calculated from the equations given in Algorithm 1b. Let us illustrate this for the first row in the table, i.e., for possible world in which P, Q and R are all true. In this world, one can verify that all seven new formulas are satisfied. Hence the contribution of these formulas to the SWSF is the sum of their weights, $\sum_{i=1}^{7} w_i$. Calculating this sum using the equalities in Algorithm 1b, and simplifying the resulting expression, we obtain $w(v_{ttt} - v_{fff})$, as indicated in the first row of the table. In the same way, the reader can verify the other rows in the table.

As the table shows, the contribution to the SWSF of the seven new formulas is the same as that of φ, up to a constant (namely $-w\ v_{fff}$). Hence, although the normalization constant is different, the distribution of the MLN is preserved.[6]

3.3 Step 2: reduce triplewise formulas to pairwise form

Step 2, shown in Algorithm 1a, reduces all triplewise formulas in the MLN to pairwise form. It loops over all formulas in the MLN. When encountering a triplewise formula φ, we execute an *atomic reduction step* on φ.

Atomic reduction step. An atomic reduction step executed on a triplewise PNF formula φ with weigth w removes φ from the MLN and replaces it by an equivalent set of pairwise (or unary) formulas. To find this set, we call $\mathit{ReduceToPairwise}(\varphi, w)$, as defined in Algorithm 1c. If w is positive, this returns four formulas; if w is negative, seven formulas are needed.

The key part of the reduction is that we introduce an *auxiliary atom* (*aux-atom*) into the vocabulary. This is analogous to an auxiliary random variable from the traditional reduction in the graphical models literature [1]. In the function $\mathit{ReduceToPairwise}()$, this aux-atom is denoted A.

Example. We continue our previous example. After Step 1, we had an MLN with a triplewise formula $P \wedge Q \wedge R$ with weight w, and a pairwise formula $P \wedge Q$ with weight $-w$. In Step 2, the pairwise formula is left unchanged, while (assuming $w > 0$) the triplewise formula is replaced by its equivalent set of four pairwise or unary formulas, as given by Algorithm 1c. The end result is hence a set a five formulas: the pairwise formula $P \wedge Q$ with weight $-w$, the pairwise formulas $A \wedge P$, $A \wedge Q$ and $A \wedge R$ each with weight w, and the unary formula A with weight $-2w$.

One important technicality that is not illustrated by this small example is that, for the reduction to be correct, we need to introduce a *separate* aux-atom (i.e., with a new, unique name) for every triplewise formula being reduced. For instance, for the i-th triplewise formula, we can introduce an aux-atom named A_i.

3.4 Equivalence of the reduced MLN (max-equivalence)

Let us call the original MLN M_0, the MLN obtained after Step 1 M_1, and the MLN after Step 2 M_2. Let P_0, P_1 and P_2 denote the probability distributions specified by respectively M_0, M_1 and M_2.

We have already shown (Section 3.2) that M_1 is equivalent to M_0. Now the question is: how does M_2 relate to M_0 and M_1? Because Step 2 introduced aux-atoms, M_2 defines a probability distribution over possible worlds described in terms of a larger vocabulary than M_0 and M_1. We show below that M_2 is *max-equivalent* to M_0 and M_1, i.e., when *maxing-out* all aux-atoms from the probability distribution of M_2, the obtained distribution is the same as that of M_0

[6] We could add the trivial formula *true* (which is satisfied in all worlds) with weight $w\ v_{fff}$ in order to also make the normalization constants equal. However, from the perspective of equality of distributions, there is no need to do this.

and M_1. Maxing-out an atom (or random variable), as used in all max-product algorithms in graphical models, is the counterpart of summing-out (marginalization).

Example (maxing-out). In our running example, the MLN M_2 obtained after Step 2 contains the atoms A, P, Q, and R and hence defines a distribution $P_2(A, P, Q, R)$ over 2^4=16 possible worlds. Maxing-out A from this distribution yields a distribution $P_2'(P, Q, R)$ over 8 possible worlds, where each 'entry' of the distribution $P_2'()$ is defined as the maximum of the two corresponding entries of the distribution $P_2()$. Concretely, $\forall a, p, q, r \in \{true, false\}$:

$$P_2'(P = p, Q = q, R = r)$$
$$\stackrel{def}{=} max(P_2(A = \underline{true}, P = p, Q = q, R = r), P_2(A = \underline{false}, P = p, Q = q, R = r)).$$

We write this more concisely as:

$$P_2'(P, Q, R) \stackrel{def}{=} max_A P_2(A, P, Q, R).$$

Example (max-equivalence). The MLNs M_0 and M_1 for our running example define a distribution $P_0(P, Q, R) = P_1(P, Q, R)$. Max-equivalence means:

$$max_A P_2(A, P, Q, R) = P_1(P, Q, R) = P_0(P, Q, R).$$

As mentioned before, the example illustrates a simple case with only one triplewise formula in the original MLN, and hence one aux-atom. In general, multiple aux-atoms are needed (one per triplewise formula). In such cases, max-equivalence is satisfied after successively maxing-out *all* aux-atoms.

Before we prove that max-equivalence indeed holds, we first clarify why max-equivalence is meaningful.

Inference on the reduced MLN. As mentioned in the introduction, pairwise reductions are mostly used for MAP inference. This is the task of, given the observed truth value of some atoms in the MLN, finding the most likely truth value of all others atoms (the *MAP solution*). Given a non-pairwise MLN M, there are two possible approaches. The *direct approach* runs MAP inference on M. The *pairwise approach* first reduces M to pairwise form and then applies MAP inference. This approach returns the most likely truth value for all non-observed atoms in M, as well as for all aux-atoms. The aux-atoms' truth values can simply be discarded as they are not part of the original inference task. Max-equivalence implies that the pairwise approach returns the same MAP solution as the direct approach (assuming we perform *exact* MAP inference and that M has a single MAP optimum). This is because MAP inferences maximizes over all non-observed atoms including the aux-atoms, and maximizing over all aux-atoms is exactly what the notion of max-equivalence assumes. This explains why max-equivalence is useful in the context of MAP inference.

Proof of max-equivalence. For any given (propositional triplewise) MLN M_0 it holds that the MLN M_2 obtained after Step 2 of our reduction is max-equivalent to M_0. Step 2 repeatedly applies atomic reduction steps. Each step

introduces a separate aux-atom, which is independent of any previous aux-atoms. Hence, maxing-out one aux-atom does not influence maxing-out any other aux-atom. Thus, to show that max-equivalence holds, it suffices to show that a single atomic reduction step applied to some MLN leads to a max-equivalent MLN. We show this below.

Each atomic reduction step takes a triplewise PNF formula $P \wedge Q \wedge R$ with weight w and replaces it by a set of four (if $w > 0$) or seven (if $w < 0$) pairwise or unary formulas. As we did for Step 1, we will again focus on the contribution of the involved formulas to the SWSF of each world. We again use a table over possible worlds to show this. We first consider the case $w > 0$. According to Algorithm 1c, $P \wedge Q \wedge R$ is replaced by four new formulas: $A \wedge P$, $A \wedge Q$ and $A \wedge R$ each with weight w, and A with weight $-2w$. The left table below show the contribution to the SWSF for (some of) the 16 possible worlds of A, P, Q and R. The right table shows the contribution after maxing-out A.

P	Q	R	A	SWSF	
t	t	t	t	w	} $max = w$
t	t	t	f	0	
t	t	f	t	0	} $max = 0$
t	t	f	f	0	
		...		...	
f	f	f	t	$-2w$	} $max = 0$
f	f	f	f	0	

$\Rightarrow$

P	Q	R	SWSF	maxed-out
t	t	t		w
t	t	f		0
	...			...
f	f	f		0

These tables should be read as follows. Consider the first row of the left table, which is for the possible world in which P, Q, R and A are all true. The four new formulas are all satisfied in this world, so their total contribution to the SWSF is the sum of their weights, namely $w + w + w - 2w = w$. For the other worlds, the contribution can be computed similarly. If we max-out A, the table over 16 possible worlds (left) collapses into one over 8 possible worlds (right). For instance, the first two rows of the left table are both for the case where P, Q and R are all true. Taking the maximum of their SWSF contributions, we obtain $max(w, 0) = w$, which is the value in the first row of the right table. The other entries of the maxed-out table (right) can be computed similarly. The end result is that the SWSF contribution is w for the first world, and zero for all other worlds.

The goal of the atomic rewriting step is to replace the formula $P \wedge Q \wedge R$. The contribution of this formula is exactly the same as that in the maxed-out table: w for the first world, and zero for the other worlds. This establishes max-equivalence.

This is for $w > 0$; the reader can verify in the same way that max-equivalence also holds if $w < 0$ (using the seven formulas of Algorithm 1c).

3.5 Beyond triplewise MLNs

The above shows how to reduce triplewise propositional MLNs to pairwise form. When given a non-triplewise MLN (i.e., when the maximum formula length is

larger than 3), we apply a *preprocessing step* that converts the MLN to triplewise form. This can be done using existing techniques: we first convert every non-triplewise formula to clausal form [7], then we reduce each non-triplewise clause to a set of triplewise formulas using the classic method with auxiliary atoms of Karp [9]. For instance, consider the clausal formula $P \vee \neg Q \vee R \vee \neg S$, with weight w. This can be converted to two triplewise formulas, namely a formula $P \vee \neg Q \vee T$ with weight w, and a hard formula $T \Leftrightarrow R \vee \neg S$, with T an auxiliary atom.[7] Note that this method can be used to bring any MLN into triplewise form, but not to reduce it further to pairwise form. For this, our reduction algorithm is needed.

4 Lifted Reduction for First-Order MLNs

So far, we only considered *propositional* MLNs. A *first-order* MLN is a set of pairs (φ_i, w_i) where each φ_i is a formula in first-order logic (we do not allow existential quantifiers or functors).[8] The resulting probability distribution is: $P(\omega) = \frac{1}{Z} exp(\sum_i w_i n_i(\omega))$, with $n_i(\omega)$ the number of satisfied groundings of formula φ_i in world ω. We refer to logical variables as *logvars* and write them in lowercase[9], e.g., x.

Reducing a first-order MLN can be done at the *ground level* or at the *lifted level*. Reduction at the ground level simply consists of using existing methods to ground the MLN and then carrying out the reduction as in the previous section, treating each ground atom as a proposition (this requires using a separate auxiliary proposition for each grounding of each triplewise formula). More interestingly, we can also do the reduction at the first-order level, i.e., in a lifted way.

Lifted reduction. This reduction is useful when performing lifted inference [8]. The approach is very similar to the propositional case, below we focus on the differences.

Given a non-triplewise MLN, we apply a preprocessing step to make it triplewise, as in the propositional case. Concretely, we convert it to first-order clausal form and then reduce non-triplewise clauses by introducing auxiliary variables using Karp's method on the first-order level [9]. For instance, consider the clausal formula $P(x) \vee Q(x, y) \vee R(y, z) \vee S(z)$, with weight w. We replace this by two triplewise formulas, namely $P(x) \vee Q(x, y) \vee T(y, z)$ with weight w, and a hard formula $T(y, z) \Leftrightarrow R(y, z) \vee S(z)$, with T an auxiliary predicate.

Once we have a triplewise MLN, the outline of the reduction algorithm is the same as in the propositional case, although there are some differences in the actual steps, see the colored comments in Algorithm 1. Step 1 is the same as before, except for the complication of 'lost logvars', which we discuss later in this section. Step 2 is also similar, the difference being that in the propositional

[7] A 'hard' formula in an MLN has infinite weight [7].

[8] This is a common restriction in most of the work on Markov Logic.

[9] This is the MLN (and first-order logic) convention, and is the opposite of the Prolog convention.

case the introduced aux-atom is a proposition A, while in the lifted case it is an atom containing the necessary logvars, as illustrated in the following example.

Example. Consider the first-order formula $P(x) \wedge Q(x,y) \Rightarrow R(y)$. This type of formula is used often in SRL, for instance in collective classification: $Class_i(x) \wedge Link(x,y) \Rightarrow Class_j(y)$. Note that this formula has exactly the same structure (connectives) as our earlier propositional example $P \wedge Q \Rightarrow R$. The reduction is thus also very similar. In Step 1, we rewrite the formula to PNF, using Algorithm 1b. This yields two formulas: $P(x) \wedge Q(x,y) \wedge R(y)$ with weight w and $P(x) \wedge Q(x,y)$ with weight $-w$. In Step 2, we reduce the first formula to pairwise form using Algorithm 1c, i.e., we replace it by four formulas: the pairwise formulas $A(x,y) \wedge P(x)$, $A(x,y) \wedge Q(x,y)$, $A(x,y) \wedge R(y)$ each with weight w, and the unary formula $A(x,y)$ with weight $-2w$. The only actual difference with the propositional example is that, as an aux-atom, we need to use $A(x,y)$, with A an auxiliary predicate. It is necessary that this atom contains all the logvars that occur in the original triplewise formula, i.e., x and y in this case. This ensures that, if we ground the obtained pairwise MLN, it will contain a separate aux-atom for every instantiation of (x,y), just as it would if we would have done the reduction at the ground level. If we would not use a separate aux-atom for each instantiation, the ground reductions would become inter-dependent and max-equivalence would no longer hold. For this reason, it is also necessary to use a separate aux-predicate for every first-order triplewise formula in the MLN.

Lifted versus ground reduction. The above lifted reduction is equivalent to the reduction on the ground level in the sense that first reducing an MLN on the lifted level and then grounding it will yield exactly the same result as first grounding the MLN and then reducing it with our ground/propositional reduction. This implies as a corollary that the max-equivalence that we had for our propositional reduction carries over to the lifted level: when reducing a first-order MLN to first-order pairwise form, the result will be max-equivalent, provided that we max-out *all instantiations* of all aux-atoms.

Correction factors for lost logvars. Finally, there is one technical complication that we have not discussed yet. This complication occurs only in special cases; it does for instance not occur in any of our previous examples, nor in any of our experiments (Section 5). For completeness, we illustrate it on another example. Consider the triplewise formula $P(x) \wedge Q(x) \Leftrightarrow R$ with weight w. If we apply Step 1 (Algorithm 1b), we obtain, after some calculations, three PNF formulas: $P(x) \wedge Q(x) \wedge R$ with weight $2w$, $P(x) \wedge Q(x)$ with weight $-w$, and R with weight $-w$. The problem is with the last formula (namely R): we have '*lost*' the logvar x in the reduction (i.e., x does not occur anymore in this formula). This is incorrect. If we would do the reduction on the ground level, we would separately rewrite every instantiation of the original formula to PNF, and we would (correctly) get one occurrence of the formula R (with weight $-w$) *for each possible instantiation* of x. Such duplicate occurrences are equivalent to a single occurrence of R with weight $-wN_x$, where N_x is the *domain size* (number of constants) of logvar x. However, in the above lifted reduction, we get only one occurrence of R with weight $-w$. The solution is simple: we correct for the lost

logvar x in our lifted reduction by setting the weight of the formula R not to $-w$ but to $-wN_x$, as it would be in the ground case.

In more general terms, whenever a formula in our lifted reduction has a lost logvar, we need to compensate by multiplying the weight with the domain size of the lost logvar. Note that lost logvars can arise in Step 1 (as in the above example) or in Step 2. The solution with the correction factor applies to both cases.

5 Experiments

To evaluate the usefulness of our reduction, we used it in experiments with MAP inference. We used ground inference algorithms (with our propositional reduction) as well as lifted algorithms (with our lifted reduction).

5.1 Ground inference: Setup

MLNs and datasets. We used two triplewise MLNs. The first is the MLN for the Smokers domain [7]. On the first-order level, this contains one triplewise formula, namely $\mathit{Friends}(x, y) \Rightarrow (\mathit{Smokes}(x) \Leftrightarrow \mathit{Smokes}(y))$, and several pairwise or unary formulas. While this is a synthetic domain, it is of interest because its formulas are similar to those in real-world MLNs (e.g., the above formula follows the common pattern $\mathit{Link}(x, y) \Rightarrow (\mathit{Property}(x) \Leftrightarrow \mathit{Property}(y))$). We generated ground MLNs for eight different domain sizes (number of people): 10, 20, 40, 60, 80, 100, 120 and 150 (beyond that, the experiments timed-out). For each domain size, we defined MAP tasks by selecting the required number of people, sampling the truth value for all ground atoms, and randomly selecting 80% of all resulting ground atoms as evidence. The MAP task was to max-out the remaining 20% of the atoms. We repeated this construction 20 times, yielding 20 different MAP tasks per domain size, so 160 in total.

The second MLN is for WebKB (`http://www.cs.cmu.edu/~webkb/`), a dataset about collective classification. On the first-order level, it contains 49 triplewise formulas of the form $\mathit{Class}_i(x) \wedge \mathit{Link}(x, y) \Rightarrow \mathit{Class}_j(y)$ (with i and j one of 7 classes), and a number of pairwise and unary formulas. We varied the domain size, by subsampling the set of webpages, from 10 to 70 in multiples of 10 (beyond that, the experiments timed-out). We learned the parameters of the MLNs from data, using 4-fold cross validation over the different data-subsets. We defined MAP tasks for each test fold by randomly selecting 80% of all ground atoms in the data as evidence and maxing-out the other 20%. We constructed 5 different MAP tasks per fold, so 20 per domain size, so 140 in total.

Algorithms. We used two ground approximate MAP algorithms, MaxWalkSAT (MWS) and MPLP. MWS [7] works in the same way irrespective of whether the MLN is pairwise or not. MPLP [3] is a Max Product variant based on Linear Programming. While the MPLP paper only discusses the pairwise case, we obtained an implementation from the authors that supports also non-pairwise models. We ran MPLP on the original triplewise MLNs (*MPLP-t*) as well as the

pairwise MLNs returned by our reduction algorithm (*MPLP-p*), with identical settings. We also ran MWS on both types of MLNs (*MWS-t* and *MWS-p*). Since MWS is an anytime algorithm, we had to choose the time-budget: for each MAP task, we set the budget equal to the runtime of MPLP-p on that task.

Evaluation measure. We evaluated the quality of the MAP assignment returned by an algorithm by computing the sum of weights of satisfied formulas in the original triplewise MLN under this assignment, as this sum is proportional to the probability of the MAP assignment, which we want to maximize [7].

5.2 Ground inference: Results

MLN characteristics. We recorded the characteristics of the triplewise MLNs and their pairwise reductions. There is of course a certain blow up when doing the reduction, e.g., the aux-atoms are introduced. How big this effect is depends on the formulas involved, on the evidence, etc. On Smokers, the effect is very small: the ratio of the number of ground atoms in the pairwise MLN versus in the triplewise MLN is always in the interval $[1.04, 1.07]$. On WebKB, the effect is larger, with ratios in the interval $[2.22, 2.87]$. The fact that these ratios are larger than 2 means that, on WebKB, the majority of all atoms in the pairwise MLNs are aux-atoms.

Algorithm comparison. When comparing the quality of the MAP solution returned by the 4 methods, we found that MPLP-p is the best. On Smokers, MPLP-p is the best method on 93% of all tasks, versus 62% for MPLP-t, 59% for MWS-t, and 3% for MWS-p. These percentages sum up to more than 100% because of ties: often multiple methods give the same MAP solution and are jointly the best. On WebKB, MPLP-p is the best on 99% of all tasks, versus 82% for MPLP-t, and 15% for both MWS-t and MWS-p. Hence, overall MPLP-p is the preferred method in terms of solution quality. However, in terms of time till convergence, MPLP-p is significantly slower than MPLP-t: a factor 4.07 slower on Smokers and a factor 1.62 slower on WebKB. This is because the pairwise MLNs are larger than the triplewise ones. We now analyze closer the difference between the triplewise and pairwise approaches.

MPLP-p vs MPLP-t. Figure 1 (top row) shows the wins/ties/losses for MPLP-p versus MPLP-t. A win (green) means that MPLP-p, i.e. the pairwise approach, is better; a loss (red) means that the triplewise approach is better. The pairwise approach clearly gives the best results, especially on larger domains. It seems that MPLP cannot deal well with many triplewise formulas, and the solution quality suffers. This proves the usefulness of our reduction to pairwise form.

MWS-p vs MWS-t. Figure 1 (bottom row) shows the results for MWS. Here the opposite trend holds: the triplewise approach gives the best results. Unlike MPLP, MWS does local search to find the MAP optimum. For the pairwise MLNs, the search space is significantly larger due to the extra aux-atoms. The results show that MWS suffers from this enlarged search space and hence performs poorly on the pairwise MLNs.

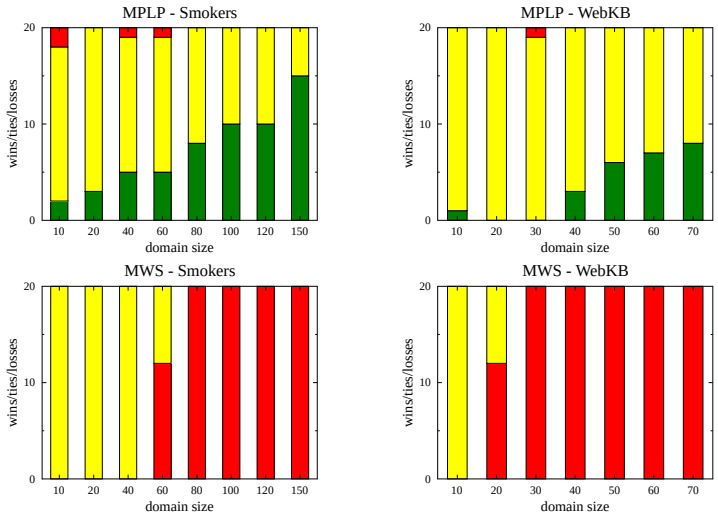

Fig. 1. *(Best viewed in color.)* Comparison of solution quality for pairwise versus triplewise MLNs. Each bar shows, for the corresponding domain size, how many of the 20 MAP tasks are 'wins' (green; pairwise is better), 'losses' (red; triplewise is better) and 'ties' (yellow).

5.3 Lifted inference: Setup

We also tested our reduction in combination with *lifted inference*. For this, we used the lifted version of our reduction. As inference algorithm, we used MAP via *Lifted Linear Programming* (using the sparse version of CVXOPT as the underlying solver) [8]. We used the Smokers MLN with domain sizes 50, 100 and 150 (beyond that, some of the experiments timed-out).

5.4 Lifted inference: Results

Triplewise versus pairwise. Table 2 shows the results. Lifted Linear Programming (LLP) performs approximate MAP inference. To indicate the quality of the solution, it computes an upper bound on the MAP objective. Since the MAP objective needs to be maximized, it holds that the lower the upper bound is, the 'tighter' LLP's approximation is. Hence, for the upper bound, lower is better. As the last column of Table 2 shows, the results are slightly better for pairwise MLNs than for the original triplewise MLNs. This shows that also in the lifted case our reduction to pairwise form is useful.
Lifting. The measured speed-ups obtained due to lifting go up to a factor 1.53 for the triplewise MLNs and 1.27 for the pairwise MLNs. The fact that these speed-ups are rather modest, might say more about the practical implementation than about the theoretical potential for lifting: the reduction of the number of variables (number of unknowns) in the LPs is very drastic, as Table 2 shows. Hence, while the actual benefit of lifting in this experiment is modest, we see

Table 2. Lifted inference results. The 1st and 2nd column (*MLN* and *domain size*) specify the input. The 3rd column (*speed-up*) gives the relative speed-up factor achieved due to lifting. The 4th and 5th column (*vars-ground* and *vars-lifted*) give the number of variables in the LP for respectively the ground and lifted case. The 6th column (*upper bound*) gives the upper bound on the LP objective, lower is better (see text); this bound is, by construction of LLP, identical for the ground and lifted case.

MLN	Domain size	Speed-up	Vars-ground	Vars-lifted	Upper bound
	50	1.22	25,200	19	$7.83e03$
Triplewise	100	1.39	100,400	19	$3.12e04$
	150	1.53	225,600	19	$7.00e04$
	50	1.07	60,400	24	$7.77e03$
Pairwise	100	1.22	240,800	24	$3.10e04$
	150	1.27	541,200	24	$6.96e04$

this mainly as a proof-of-concept that our reduction can also be combined with lifted inference.

6 Conclusion

We introduced Pairwise Markov Logic, a new subset of Markov Logic. This subset is of special interest since working with pairwise MLNs has advantages in the context of MAP/MPE inference. Our experiments with the MPLP algorithm confirm this. Since allowing only pairwise MLNs is too restrictive during modelling and learning, we have shown how to reduce a non-pairwise MLN to an equivalent pairwise MLN for running inference, both on the ground level and lifted. While we focussed on inference here, also learning will benefit from this work, as MAP/MPE inference is a sub-procedure in many learning tasks (e.g., discriminative weight learning for MLNs [7]).

We presented some first experiments on combining our lifted reduction with lifted inference. Further research in this direction is interesting future work.

Acknowledgements. We thank the reviewers for useful comments and suggestions. DF is a post-doctoral fellow of the Research Foundation-Flanders (FWO-Vlaanderen). KK was supported by the Fraunhofer ATTRACT fellowship STREAM and by the EC (FP7-248258-First-MM). MM and KK were supported by the German Research Foundation DFG (KE 1686/2-1) within the SPP 1527. JD is supported by the Research Fund K.U.Leuven (CREA/11/015 and OT/11/051), EU FP7 Marie Curie Career Integration Grant (#294068) and FWO-Vlaanderen (G.0356.12).

References

1. Gallagher, A.C., Batra, D., Parikh, D.: Inference for order reduction in Markov random fields. In: IEEE Computer Society Conference on Computer Vision and

Pattern Recognition, Los Alamitos, CA, USA, IEEE Computer Society (2011) 1857–1864
2. Kumar, M., Kolmogorov, V., Torr, P.: An analysis of convex relaxations for MAP estimation of discrete MRFs. Journal of Machine Learning Research **10** (2009) 71–106
3. Sontag, D., Meltzer, T., Globerson, A., Jaakkola, T., Weiss, Y.: Tightening LP relaxations for MAP using message passing. In: Proceedings of the 24th Annual Conference on Uncertainty in Artificial Intelligence, Corvallis, Oregon, AUAI Press (2008) 503–510
4. Cour, T., Shi, J.: Solving Markov random fields with spectral relaxation. Journal of Machine Learning Research - Proceedings Track **2** (2007) 75–82
5. Rother, C., Kolmogorov, V., Lempitsky, V.S., Szummer, M.: Optimizing binary MRFs via extended roof duality. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (2007)
6. Kolmogorov, V., Rother, C.: Minimizing nonsubmodular functions with graph cuts - A review. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(7) (2007) 1274–1279
7. Domingos, P., Kok, S., Lowd, D., Poon, H., Richardson, M., Singla, P.: Markov Logic. Lecture Notes in Computer Science. In: Probabilistic Inductive Logic Programming - Theory and Applications. Springer (2008)
8. Mladenov, M., Ahmadi, B., Kersting, K.: Lifted linear programming. In: 15th International Conference on Artificial Intelligence and Statistics, La Palma, Canary Islands, Spain (2012) Volume 22 of Journal of Machine Learning Research: Workshop & Conference Proceedings 22.
9. Karp, R.: Reducibility among combinatorial problems. In Complexity of Computer Computations. Plenum Press (1972) 85–103