Journal of Artificial Intelligence Research 55 (2016) 685–714 Submitted 11/15; published 03/16

Quadratization and Roof Duality of Markov Logic Networks

Roderick de Nijs RSDENIJS@TUM.DE
Institute of Computer Science
Albrechtstr. 28, 49076 Osnabrück, Germany

Christian Landsiedel CHRISTIAN.LANDSIEDEL@TUM.DE
Dirk Wollher DW@TUM.DE
Martin Buss MB@TUM.DE
Lehrstuhl für Steuerungs- und Regelungstechnik
Theresienstr. 90, 80333 München, Germany

Abstract

This article discusses the quadratization of Markov Logic Networks, which enables efficient approximate MAP computation by means of maximum flows. The procedure relies on a pseudo-Boolean representation of the model, and allows handling models of any order. The employed pseudo-Boolean representation can be used to identify problems that are guaranteed to be solvable in low polynomial-time. Results on common benchmark problems show that the proposed approach finds optimal assignments for most variables in excellent computational time and approximate solutions that match the quality of ILP-based solvers.

1. Introduction

First-order probabilistic models are a promising paradigm for overcoming the limitations of classical first-order logic because of their ability to capture the uncertainty often present in real-world problems. They allow describing relational knowledge compactly, such that the size of the representation is independent of the number of objects in the domain. The knowledge in these models can be defined through the use of *parfactors* (Poole, 2003), which are templates representing large numbers of factors in a graphical model that describes a probability distribution over possible world configurations. When the underlying graphical model is finite, it is possible to *ground* the first-order model and to perform inference at the propositional level. For this reason, it is of interest to identify tractable cases of propositional problems at first-order level as well as finding efficient approximate algorithms for the case where exact inference is not possible. Although computing typical inference queries on the propositional model is NP-Hard in general, such problems lend themselves to traditional optimization approaches.

This article deals with models whose MAP problem can be represented as an optimization over a finite number of binary variables. This is the case for Markov Logic Networks (MLNs), where the parfactors are weighted logic rules that can be propositionalized to define a Markov Random Field over Boolean random variables. The contributions in this work stem in great part from representing the parfactors using Boolean polynomials known as pseudo-Boolean functions. First, it is shown that for models with certain parfactors,

the MAP computation is equivalent to maximizing a *polar* or *unimodular* pseudo-Boolean function, which can be done in low-polynomial time. This allows to identify MLNs with a tractable MAP problem. Secondly, it is shown how *quadratization* techniques for pseudo-Boolean functions can be generalized to parfactors. One benefit of these transformations is that they enable using Quadratic Pseudo-Boolean Optimization (QPBO) on the ground model, a popular algorithm in computer vision that has not yet been evaluated for MLNs. The literature of such quadratizations is reviewed, and we show that the previous work discussing the quadratization of MLNs (Fierens, Kersting, Davis, Chen, & Mladenov, 2013) is equivalent to a particular choice of pseudo-Boolean quadratization within our quadratization framework for parfactors. Based on the *generalized roof duality* (Kahl & Strandmark, 2012), a new quadratization is also introduced.

Experimental evaluation of the quadratization techniques in combination with the QPBO algorithm show that large benefits in performance can be attained on real-world problems, and that the more sophisticated quadratization techniques deliver better results than the one employed by Fierens et al. (2013).

The combination of quadratization and QPBO is shown to be a competitive strategy for approaching the MAP problem in MLNs.

1.1 Outline

The remainder of this article is organized as follows: The rest of this section reviews the optimization methods available for MLNs and other related work. In Section 2, the mathematical background, Markov Random Fields and pseudo-Boolean functions are presented, along with the notation used in this article. Section 3 describes the transformation of parfactors to pseudo-Boolean form, and discusses cases that can be identified as tractable. Section 4 presents a general framework for quadratization of parfactors as well as a comparison with an existing approach for MLNs. Different quadratization strategies are discussed. In Section 5, a thorough computational evaluation of the approach is performed on datasets from literature as well as new problems. The article concludes with a discussion of the methods and results in Section 6.

1.2 Related Work

The use of first-order representations provides a compact and flexible way to encode knowledge into a model in the design phase. The trade-off for this convenience is that the ground models generally have have large treewidths, making MAP estimation and other common queries NP-hard in general. Because the ground model may have thousands or millions of variables and interactions, there is a need for fast and memory-efficient optimization algorithms. This section tries to give an overview over the recent most prominent methods for inference in these models.

Various algorithms based on heuristic random search have been used to approximate the MAP solution. The Alchemy system (Richardson & Domingos, 2006) implements a probabilistic hill-climbing algorithm named MaxWalkSat. A *lazy* variant of MaxWalkSat (Singla & Domingos, 2006b) was also developed, which, by splitting the network into an active and inactive part, considerably reduces the memory footprint of the algorithm. The Tuffy system (Niu, Ré, Doan, & Shavlik, 2011) reformulated the algorithm within

a relational database for faster grounding and additional scalability. It is also capable of detecting weakly connected components in the ground network, which can be used for parallelizing inference. A further extension of the parallelization of inference in MLNs was based on a partitioning of the network before grounding found using minimum cuts, which is used in an importance sampling inference framework (Beedkar, Del Corro, & Gemulla, 2013). An alternative approach is based on the conversion of ground factors to linear constraints, such that the MAP problem can be formulated as an integer linear program (ILP). One advantage of this formulation is that the solution of its linear relaxation gives an optimistic estimate of optimal cost. The cost of the optimal solution then lies between that of any particular assignment and the optimistic estimate. To make this approach practical for larger problems, it is necessary to reduce the number of linear constraints to be considered by the solver. To this purpose, a cutting plane algorithm for MLNs was presented (Riedel, 2009). This iterative approach ignores constraints that have so far been satisfied by previous intermediate solutions, and only includes them if they become unsatisfied in the next iteration. The RockIt system (Noessner, Niepert, & Stuckenschmidt, 2013) further shows that structurally similar constraints created by the first-order model can be aggregated into a single one, and presents a parallelization scheme that splits the problem into multiple ILPs. The combination of these techniques achieves excellent execution times. A third type of algorithms do not perform queries on a propositionalized network, but operate instead on a potentially much smaller *lifted* network. Algorithms for the lifted MAP have seen significant advances in recent years (Apsel & Brafman, 2012; Sarkhel, Venugopal, Singla, & Gogate, 2014; Mittal, Goyal, Gogate, & Singla, 2014). However, these approaches can only applied efficiently for problems with specific types relations and evidence, and will not be discussed in this article.

For most of the article, we have used a polynomial representation of the potential functions in FOPMs. The use of polynomials for the representation of Bayesian networks was suggested under the name of network polynomials (Darwiche, 2003). Such polynomials can be compiled into arithmetic circuits and used for inference (Huang, Chavira, & Darwiche, 2006). This idea is carried to first-order models to create a tractable subset of Markov Logic Networks (Domingos & Webb, 2012), whose network polynomial can be used to make queries efficient.

We have also used the fact that the maximization of any pseudo-Boolean function can be transformed to a maximization of a *quadratic* pseudo-Boolean function. Because pseudo-Boolean functions can be interpreted as factor graphs, such a transformation can be seen as a *reduction* of the MAP problem in a general binary factor graph to a MAP problem in a pairwise binary factor graph. A thorough study of reductions of inference problems in general factor graphs to more restricted factor graph models was presented by Eaton and Ghahramani (2013). Close to our work is also the idea of *pairwise MLNs* (Fierens et al., 2013), which relies on a transformation of logical formulas to compute a quadratic MLN that is equivalent to an original MLN of higher order. A detailed comparison of this approach to ours is given in Section 4.4.

2. Preliminaries

In this section we review the concepts required for the understanding of the rest of the article and define useful notational conventions.

2.1 First-Order Logic Concepts

First-order logic makes statements about *objects* in the world. Each object belongs to a certain *domain*, where domains can be seen as the semantic type of the object. References to objects are made by the use of *terms*. Terms can be either *constants*, which refer to a specific object, *logical variables*, which can represent a range of objects, or *functions*, which map terms to other terms. Like objects, logical variables and constants are typed, meaning that they represent objects from a certain domain. A *predicate* represents a relation between its arguments. A predicate applied to specific terms is an *atom*. Atoms are also called *positive literals* and their logical negation *negative literals*.

A first-order logic formula is an expression involving atoms, connected through *connectives* ($\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow, =$) and *quantifiers* ($\forall$,$\exists$). Atoms or formulas are said to be a *ground* expression if all contained terms are constants.

2.2 Notation and Conventions

An atom $P(t_1, \ldots, t_n)$ is created by applying a predicate P of arity n to a tuple of terms $(t_1, \ldots, t_n)$. The arguments of a predicate are typed, each having an associated domain. Ground atoms are represented by x and literals (first-order or ground) are represented by u; their negation is written as $\overline{u}$. Logical variables (logvars) are denoted with capital letters X, Y, Z. Vectors of atoms, ground atoms, and terms are denoted by $\mathbf{a}, \mathbf{x}, \mathbf{t}$, respectively; for instance, a first-order expression F that involves multiple atoms is written as $F(\mathbf{a}) = F(a_1, a_2, \ldots, a_n)$. To easily go from Boolean to real values, logical *True* and *False* are reinterpreted to represent 1 and 0 where necessary. Replacing a literal by its negated equivalent, e.g., u by $1 - \overline{u}$, is the *complementation* operation. Also, the superscript (γ), with $\gamma \in \mathbb{B}$, can be used to specifically refer to the positive and negated literals, e.g., $u^{(1)} = u$ and $u^{(0)} = \overline{u}$. Observe that $\overline{u^{(\gamma)}} = u^{(1-\gamma)}$.

The *substitution* of terms from a set $\mathcal{T}$ by different terms $\mathcal{T}'$ according to a mapping $\mathcal{T} \to \mathcal{T}'$ is denoted by θ. The substitution operation applied to a first-order expression f is written as $f\theta$.

A *ground substitution* is a mapping $\mathcal{T} \to \mathcal{C}$ from non-constant terms $\mathcal{T}$ to constant terms $\mathcal{C}$. Given a set of logical variables L, the set of all possible ground substitutions that satisfy constraints C is written as $gr(L : C)$. In the case where $C = \varnothing$, the number of ground substitutions is the size of the Cartesian product of the domains of L.

2.3 Markov Random Fields

A Markov Random Field is an undirected graphical model defined as

$$P(\mathbf{x}) = \frac{1}{Z} \prod_{i}^{N} \pi_i(\mathbf{x}_i),$$

where the *factors* π_i are nonnegative functions, $\mathbf{x}_i$ are tuples of binary random variables, and Z is a normalization constant. Normally π_i are exponential functions, and the quantity $-\sum_i^N \log \pi_i(\mathbf{x}_i)$ is referred to as the energy function, whose minimum defines the MAP state of $P(\mathbf{x})$.

2.4 First-Order Probabilistic Models

First-order probabilistic models are a way of expressing probability distributions, such as MRFs, with a large degree of structure. These models can be conveniently specified using *parfactors* (Poole, 2003). Parfactors are composed of a (parametrized) potential function and constraints on the valid ground substitutions of the parameters. A parfactor g is represented as a tuple $(C, \phi(\mathbf{a}))$, where C is a set of constraints and $\phi(\mathbf{a})$ is the potential of the parfactor, a real-valued function on first-order atoms $\mathbf{a}$. The potential function can be given as a table that associates a value to each of the 2^n truth states of its atoms (Poole, 2003; de Salvo Braz, 2007) or as a function. Each ground substitution then defines a factor or clique potential in a Markov Random Field or Bayes Network. The logical variables that appear in a parfactor are denoted as L, and $LV(\mathbf{a})$ is used to specifically refer to the logical variables that appear in the atoms. The set of valid ground substitutions of a parfactor with logical variables L and for constraints C (discussed in Section 2.4.2) is denoted by $gr(L : C)$. For our purposes, a set of parfactors G defines a Markov Random Field in log-linear form through the sum of all of its groundings as

$$P(\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{g \in G} \sum_{\theta \in gr(L_g : C_g)} \phi_g(\mathbf{a}_g)\theta \right), \tag{1}$$

where $\mathbf{x}$ contains the propositional variables that arise from grounding the first-order atoms.

Markov Logic Networks are first-order probabilistic models that use a set of weighted first-order logic rules to specify a Markov Random Field. The rules are generally specified manually and capture the available knowledge or intuitions about the domain. The weights capture the relative importance of the rules and can be set manually or be learnt from data. Markov Logic Networks can be easily described using parfactors. For this, each weighted first-order logic rule in a Markov Logic Network is associated to a parfactor with an empty constraint set and a potential function that takes the value of the weight for satisfying assignments to the rule, and 0 otherwise. For these models, (1) assigns high probabilities to world states that satisfy many ground parfactors with a positive weight and few with a negative weight.

The most common types of queries to (1) are the marginal probabilities for each of the propositional variables $(P(x_1), P(x_2), \ldots)$ and the most probable configuration of the unknown variables $\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x})$, also referred to as the Maximum-a-Posteriori (MAP) assignment.

In order to be able to ground the models, we take similar assumptions to the original MLN formulation, namely that domains are assumed to be finite, unique names and domain closure. In practice these also allow to assume the logical atoms to be function-free. However, although potential functions in MLNs are $\{0, w\}$-valued, $w \in \mathbb{R}$, our formulation allows potentials to take different values in $\mathbb{R}$ for every assignment.

2.4.1 Evidence

The model may be conditioned on the truth value of certain ground atoms. The symbols P_T and P_F represent the sets of ground atoms for a predicate P that are known to be *True* and *False*, respectively. For instance, $P(\mathbf{o}) \in P_T$ denotes that the ground atom $P(\mathbf{o})$ is known to be *True*. In case no evidence is available, P_T and P_F are empty. The set of ground atoms with predicate P and unknown truth value is represented by P_U. If P_U is empty, the predicate is *fully observed*.

2.4.2 Constraints

The ground substitutions for the logical variables in a parfactor can be subject to *constraints*. In our representation, the substitution constraint for each parfactor is the conjunction of a set of individual constraints associated with that parfactor. Disjunctions of constraints can be expressed through multiple parfactors under this representation. Constraints are used for the following cases:

1. Expressing the (in)equality relation between logical variables. For two logical variables X, Y belonging to the same domain, $X = Y$ and $X \neq Y$ respectively restrict the ground substitutions to those that map X and Y to either the same or different constants.

2. Expressing that ground substitutions must map logical variables $\mathbf{t}$ to elements in a set of objects $P_\circ$ is denoted as $\mathcal{C}_\in(\mathbf{t}, P_\circ)$, where $P_\circ$ generally is one of the evidence groups P_T, P_F, P_U for the predicate P.

Constraints can in principle also be expressed in the potential function using fully observed auxiliary predicates, as normally done in the MLN formalism. For instance, $X \neq Y$ can be enforced by taking the conjunction between the potential of the parfactor and a new fully observed atom $\mathit{AreDifferent}(X, Y)$. Similarly, the constraint $\mathcal{C}_\in(X, P_F)$ can be represented by taking the conjunction of the potential with a new atom $\mathit{BelongsToFalseEvidence}P(X)$. However, expressing these relationships in the form of constraints simplifies the discussion in Section 3.

Example 1. *That friends have similar smoking behavior can be described by a parfactor g with potential function $Friends(X, Y) \wedge Smokes(X) \rightarrow Smokes(Y)$ and constraint $X \neq Y$. For people domain $\{A, B\}$, the ground substitutions associated to the parfactor are $gr((X, Y) : \{X \neq Y\}) = \{(X, Y) \rightarrow (A, B), (X, Y) \rightarrow (B, A)\}$.*

2.5 Pseudo-Boolean Functions

A function $f : \mathbb{B}^n \rightarrow \mathbb{R}$ is called a pseudo-Boolean function. Let $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ be a vector of n binary variables. Consider also the set of literals, $\mathbf{L} \coloneqq \{x_1^{(1)}, \ldots, x_n^{(1)}, x_1^{(0)}, \ldots, x_n^{(0)}\}$. A pseudo-Boolean function where terms are expressed with literals in $\mathbf{L}$ and coefficients $e_i \in \mathbb{R}, i = 0, \ldots, n$ can be written as

$$\phi(\mathbf{x}) = e_0 + e_1 m_1(\mathbf{x}_1) + e_2 m_2(\mathbf{x}_2) + \ldots + e_n m_n(\mathbf{x}_n) \tag{2}$$

where $m_i(\mathbf{x}_i)$ are monomials over the literals in $\mathbf{L}$,

$$m_i(\mathbf{x}_i) \coloneqq \prod_j x_{i,j}^{(\gamma_{i,j})}.$$

If $e_i \geq 0$ for $0 < i \leq n$ in (2), the pseudo-Boolean function is said to be a *posiform*. There are many possible posiform representations for a pseudo-Boolean function. Just as with standard polynomials, the *order* (or degree) of a pseudo-Boolean function is that of the term with highest degree. If the pseudo-Boolean function is expressed only over the set of positive literals, i.e. $\boldsymbol{\gamma}_i = \mathbf{1}$ for all i, it is called a *multi-linear polynomial* representation

$$\phi(\mathbf{x}) = e_0 + \sum_{i \leq n} e_i x_i + \sum_{1 \leq i < j \leq n} e_{ij} x_i x_j + \sum_{1 \leq i < j < k \leq n} e_{ijk} x_i x_j x_k + \ldots,$$

where $e_i, e_{ij}, e_{ijk}, \ldots \in \mathbb{R}$. This representation is unique and can always be obtained from another representation by eliminating the negated literals using the complementation $x^{(0)} = 1 - x^{(1)}$. Conversely, a posiform can always be obtained for any pseudo-Boolean function. Complementing any literal in a term $e_i m_i(\mathbf{x}_i)$ with $e_i < 0$ produces two new terms, one of the same order with $e_i' > 0$ and one of one order lower and $e_i'' < 0$. By applying this procedure starting at the highest-order terms, all negative terms can be eliminated.

3. Parfactors with Pseudo-Boolean Potentials

This chapter describes how the potential functions of parfactors can be described and manipulated in terms of pseudo-Boolean functions, and how equivalent model representations can be translated to the pseudo-Boolean formulation. This representation allows easy recognition of some cases in which inference in the ground probabilistic model is tractable. These are detailed in Section 3.3.

3.1 First-Order Pseudo-Boolean Functions

Potential functions in parfactors are defined over first-order atoms. Therefore, we call a pseudo-Boolean function over first-order atoms a first-order pseudo-Boolean function. Substitutions are applied to each individual term, so that for a pseudo-Boolean function $\phi(\mathbf{a})$ in form (2) and a substitution θ

$$\phi(\mathbf{a})\theta = e_0 + e_1 m_1(\mathbf{a}_1\theta) + e_2 m_2(\mathbf{a}_2\theta) + \cdots + e_n m_n(\mathbf{a}_n\theta).$$

Remark 1. *The notions of* term *and* order *are different for pseudo-Boolean functions and in first-order logic. However, we expect the context to generally be clear enough to avoid confusions.*

3.2 Conversion from Other Potential Representations

In general, a potential function with n variables given in the form of a table can be converted to pseudo-Boolean form by a simple technique (Boros & Hammer, 2002), which creates one term for each of the at most 2^n configurations with a nonzero coefficient in the table. The number of terms can then be reduced by converting the pseudo-Boolean function to

a multi-linear polynomial. However, the potentials of Markov Logic Networks are given as first-order logic sentences and can be expressed in conjunctive normal form. This allows them to be translated directly into pseudo-Boolean form. Namely, each clause U can be replaced by

$$\bigvee_{u \in U} u = 1 - \bigwedge_{u \in U} \overline{u}, \tag{3}$$

and all conjunctions can be replaced by products. For the typical case of a single-clause formula $u_1 \vee u_2 \ldots \vee u_n$ that takes a value w when satisfied, the equivalent compact pseudo-Boolean representation is $w - w\overline{u}_1\overline{u}_2 \ldots \overline{u}_n$. Of course, the inverse procedure can be used to transform a potential in this form back to a logic representation.

The Markov Random Field that results from grounding a model like (1) can be represented as

$$P(\mathbf{x}) = \frac{1}{Z} \exp\left(-\phi_T\left(\mathbf{x}\right)\right),$$

where the energy function ϕ_T is obtained by summing the groundings of the parfactors in G, and the normalization constant Z is unknown.

3.3 Tractable Classes

The tractability of the MAP estimate of an MLN can be analyzed by considering classes of tractable pseudo-Boolean functions. First, we discuss classes of polynomial-time optimizable pseudo-Boolean functions and then the general case.

3.3.1 Classes of Tractable Pseudo-Boolean Functions

This section gives an overview of the relevant tractable classes of pseudo-Boolean functions. Figure 1 illustrates the relations between these function classes.

- *Supermodular functions* satisfy $f(\mathbf{x}_1) + f(\mathbf{x}_2) \leq f(\mathbf{x}_1 \cap \mathbf{x}_2) + f(\mathbf{x}_1 \cup \mathbf{x}_2)$, for element-wise AND/OR operations and binary $\mathbf{x}_1, \mathbf{x}_2$. These functions are maximizable in strong polynomial time (Orlin, 2009). However, the *recognition* of supermodularity for polynomials of order ≥ 4 is co-NP-Complete (Gallo & Simeone, 1989).

- *Supermodular functions expressible over quadratic functions* can be written as maximizations over auxiliary variables of *quadratic supermodular* functions. For instance, $x_1x_2x_3 = \max_w x_1w + x_2w + x_3w - 2w$, with $w \in \mathbb{B}$. For functions with a certain structure, expressibility can be recognized efficiently (Živnỳ & Jeavons, 2008). However, it is unknown whether the recognition of *expressible* functions is easier than the general supermodularity recognition problem (Živnỳ, Cohen, & Jeavons, 2009). One may try to obtain the equivalent quadratic supermodular function by solving a linear program (Ramalingam, Russell, Ladicky, & Torr, 2011).

- *Polar functions* (Billionnet & Minoux, 1985) are supermodular functions where each term has a positive coefficient and is composed of only positive or only negative literals, e.g., $x_1x_2x_3 + 2\bar{x}_1\bar{x}_2 + \bar{x}_2\bar{x}_4\bar{x}_5$. They form a strict subset of the set of expressible functions for orders ≥ 4.

- *Unimodular functions* are functions that can be converted to polar functions by *switching* a subset of the variables. For instance, $f(x_1, x_2, x_3) = x_1\bar{x}_2x_3 + 2\bar{x}_1x_2$ can be associated to the polar function $g(x_1, \tilde{x}_2, x_3) = x_1\tilde{x}_2x_3 + 2\bar{x}_1\bar{\tilde{x}}_2$ using the switched variable $\tilde{x}_2 = 1 - x_2$. Undoing the switching operation on the maximizer of this polar function gives a solution to the original problem. Unimodular functions are recognizable in polynomial time (Crama, 1989). If the switching operations can be used to make the function supermodular (but not polar) it is a *permutable supermodular function* (Schlesinger, 2007), which is mainly interesting for the optimization of functions with nonbinary discrete variables.

Remark 2. *If f is a supermodular function, $-f$ is submodular, and thus the results for the* maximization *of a supermodular function and the* minimization *of a submodular function are interchangeable. In the overview given in this section, we keep the context of maximization of supermodular functions, since it reflects that of many of the original publications.*

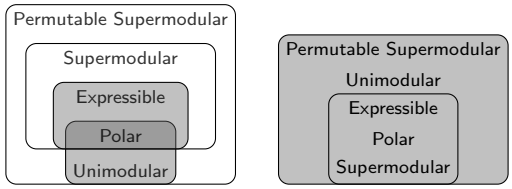

Figure 1: Relations between classes of functions. Left: For pseudo-Boolean functions of any order. Right: The third-order (cubic) case. Shaded regions are optimizable through maximum flows.

The main interest for expressible supermodular functions is that quadratic supermodular functions can be optimized by computing a maximum flow in $\mathcal{O}(n^3)$, which is considerably faster than the $\mathcal{O}(n^6)$ complexity of general supermodular maximization. However, the process of transforming an expressible supermodular function into an equivalent quadratic representation introduces additional variables. Consequently, if k auxiliary variables are introduced, the true complexity is $\mathcal{O}((n+k)^3)$.

Note that although the recognition of general supermodularity and expressibility are hard problems, these classes are closed under conical combinations, such that sums of supermodular functions are also supermodular.

3.3.2 Tractable Parfactor Models

Although inference is NP-hard in general, it is possible to guarantee that certain inference tasks are tractable by restricting the expressiveness of the formalism of the potential functions (Domingos & Webb, 2012). Translating results for some of the classes of pseudo-Boolean functions above, it is possible to easily identify models with tractable MAP inference. For this, the following result is used

Proposition 1. *If all parfactors have expressible potentials, the maximization of the ground model can be expressed as a maximization over a supermodular quadratic function.*

Proof. It follows from the facts that a) by definition, expressible functions can be written as a maximization over a supermodular quadratic pseudo-Boolean function and b) sums of supermodular functions are also supermodular. □

Consequently, if all potentials can be converted to a supermodular expressible pseudo-Boolean function, then the model can be optimized by computing a maximum flow.

This result applies to any potential, not only the $\{0, w\}$-valued potentials used in MLNs. It is hence possible to consider any potentials defined over a table and taking up to 2^n distinct values. However, this flexibility makes it harder to enforce expressibility at design level. Some classes of logic rules used in MLNs can be guaranteed to be expressible, and can be used to recognize or design MLNs with a tractable MAP problem.

3.3.3 Polar MLNs

Let $S = \{a_1, a_2, \ldots, a_n\}$ be a set of first order atoms, I and $J \subseteq \{1, 2, \ldots, n\}$ and $\gamma \in \mathbb{B}$. The logic potentials of the form

$$\bigwedge_{i \in I} a_i^{(\gamma)} \vee \bigwedge_{j \in J} a_j^{(1-\gamma)}. \tag{4}$$

can be transformed to an equivalent polar posiform for the following cases

- $J = \varnothing$. In this case (4) is a single conjunction and its pseudo-Boolean form is

$$\prod_{i \in I} a_i^{(\gamma)}.$$

- $I \cap J \neq \varnothing$. Here the conjunctions are in conflict. In this case the disjunction can be replaced by a sum without changing the underlying truth table of the expression. The equivalent polar posiform is

$$\prod_{i \in I} a_i^{(\gamma)} + \prod_{j \in J} a_j^{(1-\gamma)}.$$

- $I \cap J = \varnothing$ and $|J| = 1$. It is easy to see that

$$\bigwedge_{i \in I} a_i^{(\gamma)} \vee a_j^{(1-\gamma)} = \bigwedge_{i \in I \cup \{j\}} a_i^{(\gamma)} \vee a_j^{(1-\gamma)},$$

  which can be employed to transform the expression to the previous case.

MLNs for which all parfactors are polar have a tractable MAP problem that can be computed through a maximum flow.

Example 2. *An MLN potential with positive weight and potential* $(P(X) \wedge Q(Y) \wedge T(Z)) \vee (\overline{R(X)} \wedge \overline{T(Z)})$ *is of the form* (4) *with* $\gamma = 1$, $I = \{P(X), Q(Y), T(Z)\}$, $J = \{R(X), T(Z)\}$. *Because* $I \cap J = \{T(Z)\} \neq \varnothing$, *it has the equivalent polar potential* $P(X)Q(Y)T(Z) + \overline{R(X)}\,\overline{T(Z)}$.

3.3.4 Unimodular MLNs

Unimodular functions can also be solved efficiently once converted to polar functions. Finding the variables that need to be switched to obtain a polar function can be performed in polynomial-time (Crama, 1989). This recognition procedure may be more efficient if performed on first-order level, but would require to describe switches of the ground atoms before grounding. Here we show that the switches of some of the ground variables can also be represented compactly on first-order level, which makes it possible to decide whether a model is unimodular without analyzing the ground model. Define the switching operation as replacing a literal $l = P(\mathbf{t})^{(\gamma)}$ by expression $\tilde{P}(\mathbf{t})^{(1-\gamma)}$, where $\tilde{P}$ is a new predicate. The new literal with the switched predicate is interpreted as referring to switched ground atoms. However, the representation may become inconsistent if the ground atoms represented by a switched and a non-switched atom intersect, as a ground atom and its switch can not be treated as independent variables.

Definition 1 (Shattering)**.** *(de Salvo Braz, 2007) A set of parfactors G is* shattered *if the groundings of every pair of atoms appearing in the parfactors of G are either identical or disjoint.*

A model can be shattered by partitioning the parfactors (de Salvo Braz, 2007).

Definition 2 (Consistent switch)**.** *A shattered model with switched atoms is* consistent *if for every pair of atoms with identical groundings, either both or neither of the atoms are switched.*

In a shattered model, inconsistencies can be avoided by ensuring that each switching operation preserves the consistency. Namely, starting from a shattered model with no switched atoms, switching atoms with the same groundings ensures that the consistency requirement is fulfilled at all times. If for some switching sets all parfactors are polar, the ground model is unimodular and a polar representation is directly available.

Example 3. *Consider a model G with parfactors $(\varnothing, P(X)\overline{Q(Y)}\,\overline{T(Z)})$ and $(\varnothing, Q(X)T(Y))$. This model is shattered and we can obtain the switched model G' with parfactors $(\varnothing, P(X)\tilde{Q}(Y)\,\tilde{T}(Z))$ and $(\varnothing, \overline{\tilde{Q}(X)}\,\overline{\tilde{T}(Y)})$. Because after switching the terms in the parfactors have either only positive or only negative literals and assume only nonnegative values, the ground model is a polar pseudo-Boolean function.*

Unfortunately, the shattering condition is not a strong enough partitioning of the parfactors to guarantee that any unimodular model can be recognized. For example, the model with the single parfactor $(\{X \neq Y\}, -F(X,Y)F(Y,X))$ is shattered and can be shown to have a switching set that makes it polar, but it can not be made polar by switching the atoms. A finer partitioning of the parfactors than the one given by shattering may allow to represent the desired switching set. However, how to find such a partitioning in the general case is unknown.

3.4 Non-tractable Case

In the following we discuss the optimization of pseudo-Boolean functions that do not fall within the described tractable classes. It has been observed that multiple linear programming relaxations of quadratic pseudo-Boolean problems have the same optimum (Boros &

Hammer, 2002), known as the *roof dual bound*. This is generally an optimistic bound to the true optimum of the problem, as linear programming relaxes the integrality constraint on the variables. The roof dual bound can also be obtained very efficiently by solving a maximum flow problem on a specially constructed network. In a minimization setting, this network represents the tightest *submodular relaxation* of the original quadratic function (Kahl & Strandmark, 2012). In addition to a lower bound, the solution to the relaxed problem gives *persistencies*. Persistencies are assignments to a subset of the variables that form part of at least one optimal solution. If persistencies are found for the full set of variables, they form a minimizer of the problem. Otherwise, the problem can be simplified by fixing the persistencies to their value to produce a smaller problem that preserves the minimum.

This preprocessing step reduces the size of the problem or solves it completely, and can be combined with any other optimization method. This approach is known as *Quadratic Pseudo-Boolean Optimization* (QPBO) (Kolmogorov & Rother, 2007) in the computer vision literature.

Furthermore, the same network that is used to compute the initial roof dual bound and the persistencies can be used to search for additional persistencies or an approximate configuration for the remaining variables by means of the *probing* (Boros, Hammer, & Tavares, 2006) and *improving* (Rother, Kolmogorov, Lempitsky, & Szummer, 2007) techniques. Probing heuristically chooses a variable x from the residual problem and recomputes the maximum flow under the assumptions of $x = 0$ and $x = 1$. Analyzing the value of the remaining variables under each of these assumptions, it may be possible to find additional persistencies and improve the lower bound. The *improve* method fixes a subset of the variables to any given assignment and efficiently searches for configurations of the remaining variables that improve the cost with respect to the original assignment. This procedure can be executed multiple times and is guaranteed to not decrease the quality of the approximate solution.

The described techniques are only applicable for quadratic problems. The next section is concerned with models that can not be described by purely pairwise interactions between the variables. For such models, the use of quadratization techniques can be employed, which enables the use of the QPBO algorithm and its extensions on such problems. Also, a connection between this approach and an alternative based on *generalized* roof duality is shown in Section 4.2.

4. Quadratization

In this section the quadratization of pseudo-Boolean functions is reviewed and it is then shown how these methods can be applied to parfactor models. The quadratization procedure is also known as *order reduction*.

4.1 Quadratization of a Pseudo-Boolean Function

A quadratization of a pseudo-Boolean function $\phi(\mathbf{x})$ is a new function $\rho(\mathbf{x}, \mathbf{w})$ such that

$$\phi(\mathbf{x}) = \min_{\mathbf{w}} \rho(\mathbf{x}, \mathbf{w}), \tag{5}$$

where $\rho(\mathbf{x}, \mathbf{w})$ is a quadratic pseudo-Boolean function defined over auxiliary *slack variables* $\mathbf{w}$.

For any $\phi(x)$, there always exists a $\rho(\mathbf{x}, \mathbf{w})$ satisfying (5) (Rosenberg, 1975; Ishikawa, 2011). Furthermore, if $\phi(\mathbf{x})$ belongs to the expressible set of submodular functions described by Živnỳ et al. (2009), a submodular $\rho(\mathbf{x}, \mathbf{w})$ can also be found. After quadratization, the problem of finding $\min_{\mathbf{x}} \phi(\mathbf{x})$ is replaced by the quadratic problem $\min_{\mathbf{x},\mathbf{w}} \rho(\mathbf{x}, \mathbf{w})$.

Example 4. *The function* $f(x_1, x_2, x_3) = -x_1 x_2 x_3$ *has a quadratization*

$$\min_w \rho(x_1, x_2, x_3, w) = \min_w -x_1 w - x_2 w - x_3 w + 2w.$$

This can be verified by minimizing $\rho(x_1, x_2, x_3, w)$ *with respect to* w *for each assignment of* $(x_1, x_2, x_3) \in \mathbb{B}^3$ *and verifying that it evaluates to* -1 *at* $(1, 1, 1)$ *and* 0 *everywhere else. Now* $\min_{x_1,x_2,x_3} f(x_1, x_2, x_3)$ *can be conveniently computed through the quadratic optimization* $\min_{x_1,x_2,x_3,w} \rho(x_1, x_2, x_3, w)$. *This quadratization is obtained with the ISH technique discussed in the following.*

In general, a quadratization is not necessarily determined for the whole function at once; instead, single or multiple terms with a specific algebraic form can be replaced at each step with an equivalent quadratic representation. Because each of these substitutions introduces a minimization over independent slack variables, these minimizations can be distributed over the whole expression, such that the final quadratization is a joint optimization over all slack variables.

For a given $\phi(\mathbf{x})$ there are many possible ways to obtain a quadratization. In practice, quadratizations should a) have few slack variables, b) have few positive quadratic and thus non-submodular terms and c) be easily computable. Condition a) is important if the quadratization needs to be applied many times, e.g., if there are many higher order terms; b) their number and magnitude is experimentally known to correlate with the complexity of the resulting optimization (Kolmogorov & Rother, 2007; Gallagher, Batra, & Parikh, 2011); c) to satisfy the above conditions, finding a quadratization may become more complex. For instance, finding the quadratization with the smallest number of slack variables is an NP-complete problem for some approaches (Boros & Hammer, 2002).

4.2 Quadratization Techniques

Quadratization techniques for pseudo-Boolean functions rely on identifying subexpressions that match a template for which a quadratic form is known. Potentials that are known to be expressible can be quadratized using the submodularity-preserving functions presented by Živnỳ and Jeavons (2008). In the following we review existing techniques valid for any pseudo-Boolean function and introduce a new quadratization.

4.2.1 General Quadratization for Individual Terms (ISH)

This technique can be used to quadratize any individual higher-order term. The general formulas are given by Ishikawa (2011), and more restricted cases are given by Kolmogorov and Zabin (2004) and Freedman and Drineas (2005).

For binary variables $x_1, \ldots, x_d$ and negative coefficient ($a < 0$),

$$ax_1^{(\gamma_1)} \cdots x_d^{(\gamma_d)} = \min_{w \in \mathbb{B}} aw\{S_1^{(\boldsymbol{\gamma})} - (d-1)\}.$$

Example 4 illustrates this case for the cubic function. For a positive coefficient ($a > 0$) the quadratization is

$$ax_1^{(\gamma_1)} \cdots x_d^{(\gamma_d)} = a \min_{w_1,\ldots,w_{n_d} \in \mathbb{B}} \sum_{i=1}^{n_d} w_i(c_{i,d}(-S_1^{(\boldsymbol{\gamma})} + 2i) - 1) + aS_2^{(\boldsymbol{\gamma})}$$

with the definitions

$$S_1^{(\boldsymbol{\gamma})} = \sum_{i=1}^{d} x_i^{(\gamma_i)}, \qquad S_2^{(\boldsymbol{\gamma})} = \sum_{i=1}^{d-1} \sum_{j=i+1}^{d-1} x_i^{(\gamma_i)} x_j^{(\gamma_j)} = \frac{S_1^{(\boldsymbol{\gamma})}(S_1^{(\boldsymbol{\gamma})} - 1)}{2}$$

$$n_d = \left\lfloor \frac{d-1}{2} \right\rfloor, \; c_{i,d} = \begin{cases} 1, & \text{if } d \text{ is odd and } i = n_d \\ 2, & \text{otherwise} \end{cases}$$

The reduction of a negative term can always be performed with a single slack variable, and results in a quadratic submodular function. For positive terms, the number of slack variables n_d is proportional to $\frac{1}{2}d$. In general, the resulting function is not submodular because the quadratization creates positive quadratic terms. However, because these terms do not involve slack variables, they may be canceled by existing quadratic terms of opposite sign. In fact, it can be seen that this quadratization is always submodularity-preserving for cubic functions (Živnỳ & Jeavons, 2008). On the other hand, if the original function is not submodular, this quadratization may produce more non-submodular terms and worse experimental results than other methods.

4.2.2 Asymmetric Quadratization (ASM)

(Gallagher et al., 2011) An asymmetric quadratization for a term $ax_1x_2x_3$ with $a > 0$ is given by

$$ax_1x_2x_3 = \min_w a(w - x_2w - x_3w + x_1w + x_2x_3).$$

This can be obtained by transforming $ax_1x_2x_3$ into $ax_2x_3 - a\overline{x_1}x_2x_3$ and then using the ISH method. Observe that the left-hand side of the equality is symmetric, but the right-hand side is not. Reordering the variables thus creates three different quadratizations. Also, the right-hand side only has two non-submodular terms (compared to three with the *ISH* method), one of which does not involve the slack variable w. If there is an existing term with a negative coefficient and the same variables as the non-submodular term without w, these can be combined or may cancel out, leaving just a single non-submodular term. Asymmetric quadratizations can be created for terms of any order using a similar procedure.

The possibility of eliminating non-submodular terms motivates the search for the combination of quadratizations that optimizes a certain cost representing the quality of the quadratization. It has been shown that the total magnitude of the non-submodular terms is a good cost function (Gallagher et al., 2011). We call ASM the procedure of minimizing

this quantity when finding a quadratization for an expression, where each term can select between one of the asymmetric or the ISH quadratic forms. Note that, on propositional level, the number of such terms may be very large, resulting in a large optimization problem for the choice of quadratization per term.

4.2.3 Preprocessing for Positive Terms (FIX)

(Fix, Gruber, Boros, & Zabih, 2011) This approach can be seen as a preprocessing method to avoid the quadratization *ISH* for positive terms. It transforms multiple positive higher-order terms with a common subset of variables into negative higher-order terms. We summarize this procedure for the case where the common subset of variables consists of a single variable. Consider a set of terms S that contain a common variable x_1, where each $H \in S$ has a positive coefficient $\alpha_H > 0$. Then, it holds that for any assignment to the binary variables $x_1, \ldots, x_n$

$$\sum_{H \in S} \alpha_H \prod_{j \in H} x_j = \min_{w \in \{0,1\}} \left(\sum_{H \in S} \alpha_H \right) x_1 w + \sum_{H \in S} \alpha_H \prod_{j \in H \setminus \{1\}} x_j - \sum_{H \in S} \alpha_H w \prod_{j \in H \setminus \{1\}} x_j.$$

Observe that the last sum is over terms that have the same order as the left-hand side but with negative sign, and that all positive terms have a lower order. This transformation is repeatedly applied until all positive higher-order terms have been eliminated. At this point the ISH quadratization for negative terms is applied, introducing one additional slack variable per term. There is no method to decide on what common variable to perform the transformation, and x_1 is thus selected arbitrarily.

4.2.4 Generalized Roof Duality (GRD)

This new quadratization method is based on the generalized roof duality theory (Kolmogorov, 2012) and its practical implementation (Kahl & Strandmark, 2012). For a non-submodular, higher-order function $\phi(\mathbf{x})$, the approach finds a submodular relaxation $\tau(\mathbf{x}, \mathbf{y})$ for which

$$\begin{aligned} \phi(\mathbf{x}) &= \tau(\mathbf{x}, \overline{\mathbf{x}}) \\ \tau(\mathbf{x}, \mathbf{y}) &= \tau(\overline{\mathbf{y}}, \overline{\mathbf{x}}) \qquad \text{(symmetry)} \\ &\tau \text{ is submodular and expressible.} \end{aligned} \tag{6}$$

Under these constraints, the solution of the relaxed problem provides a lower bound and persistent assignments for some variables, similar to the roof dual relaxation of a quadratic function. By restricting the search for relaxations over expressible functions, the result can be optimized by solving a maximum flow problem. The following proposition presents a link between the submodular relaxations computed by GRD and quadratizations. The proposition can be used to compute a quadratization of a function such that its roof dual bound is equivalent to the GRD bound of the function.

Proposition 2. *Let $\tau(\mathbf{x}, \mathbf{y})$ be a relaxation of $\phi(\mathbf{x})$ that satisfies* (6)*. Then there exists a quadratic $\rho(\mathbf{x}, \mathbf{w})$ s.t. a) $\texttt{roofdual}(\rho(\mathbf{x}, \mathbf{w})) \geq \min_{\mathbf{x},\mathbf{y}} \tau(\mathbf{x}, \mathbf{y})$, and b) $\min_{\mathbf{w}} \rho(\mathbf{x}, \mathbf{w}) = \phi(\mathbf{x})$.*

Proof. Let $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{w})$ be a submodular quadratization of $\tau(\mathbf{x}, \mathbf{y})$. Define

$$\varphi'(\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{v}) = \frac{1}{2}(\varphi(\mathbf{x}, \mathbf{y}, \mathbf{w}) + \varphi(\overline{\mathbf{y}}, \overline{\mathbf{x}}, \overline{\mathbf{v}})).$$

It is easy to check that $\min_{\mathbf{w},\mathbf{v}} \varphi'(\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{v}) = \tau(\mathbf{x}, \mathbf{y})$. Also, because $\varphi(\mathbf{x}, \mathbf{y}, \mathbf{w})$ is quadratic and submodular, it is easy to see that $\varphi(\overline{\mathbf{y}}, \overline{\mathbf{x}}, \overline{\mathbf{v}})$ must also be submodular (negative quadratic terms in multi-linear form). Consequently, $\varphi'(\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{v})$ is submodular.

a) Let $\rho(\mathbf{x}, \mathbf{w}) = \varphi'(\mathbf{x}, \overline{\mathbf{x}}, \mathbf{w}, \overline{\mathbf{w}})$ and verify that

$$\varphi'(\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{v}) = \varphi'(\overline{\mathbf{y}}, \overline{\mathbf{x}}, \overline{\mathbf{v}}, \overline{\mathbf{w}})$$

Thus, $\varphi'(\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{v})$ is a relaxation of $\rho(\mathbf{x}, \mathbf{w})$ under conditions (6). However, the roof dual bound is known to be larger than all other submodular relaxations (Kahl & Strandmark, 2012),

$$\texttt{roofdual}(\rho(\mathbf{x}, \mathbf{w})) \geq \min_{\mathbf{x},\mathbf{y},\mathbf{w},\mathbf{v}} \varphi'(\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{v}) = \min_{\mathbf{x},\mathbf{y}} \tau(\mathbf{x}, \mathbf{y}).$$

b) follows from

$$\min_{\mathbf{w}} \rho(\mathbf{x}, \mathbf{w}) = \min_{\mathbf{w}} \varphi'(\mathbf{x}, \overline{\mathbf{x}}, \mathbf{w}, \overline{\mathbf{w}}) = \min_{\mathbf{w}} \frac{1}{2}(\varphi(\mathbf{x}, \overline{\mathbf{x}}, \mathbf{w}) + \varphi(\mathbf{x}, \overline{\mathbf{x}}, \mathbf{w})) = \tau(\mathbf{x}, \overline{\mathbf{x}}) = \phi(\mathbf{x}).$$

□

Condition b) in Proposition 2 ensures that $\rho(\mathbf{x})$ is a quadratization of $\phi(\mathbf{x})$, and condition a) ensures that the roof dual bound of this quadratization is at least that of the relaxation $\tau(\mathbf{x}, \mathbf{y})$.

Example 5. *For a cubic term $x_1x_2x_3$ we can compute a third-order submodular relaxation using (Kahl & Strandmark, 2012). Because the* ISH *quadratization is submodularity-preserving for cubic functions, this method can be used to make the relaxation quadratic, and then the procedure from Proposition 2 results in*

$$\begin{aligned} x_1x_2x_3 = \min_{w_1,w_2} \frac{1}{2}(&w_1x_1 + w_1x_2 - w_1x_3 - w_1 - w_2x_1 - w_2x_2 + w_2x_3 \\ &+ w_2 + 2x_1x_2 - x_1 - x_2 + x_3 + 1). \end{aligned}$$

The relaxation obtained using the generalized roof duality theory can be expressed as a quadratization of the function. In practice, this method allows us to use the GRD approach to design quadratizations that are specially suited for the potentials at hand.

4.3 Quadratization of MLNs

A quadratization of an MLN expresses its probability distribution using only parfactors with quadratic pseudo-Boolean potentials and an optimization over slack atoms.

Definition 3 (First-order quadratization)**.** *Let $\phi(\mathbf{a})$ be a pseudo-Boolean potential ϕ applied to first-order atoms $\mathbf{a}$, and construct $l = (l_1, l_2, \ldots, l_{|LV(\mathbf{a})|})$, $l_i \in LV(\mathbf{a})$ with an arbitrary ordering. From a quadratization ρ for ϕ, such that $\phi(\mathbf{x}) = \min_{\mathbf{w}} \rho(\mathbf{x}, w_1, w_2, \ldots, w_k)$ we*

define the first-order quadratization *as* $\phi(\mathbf{a}) = \min_{\mathbf{b}} \rho(\mathbf{a}, b_1, b_2, \ldots, b_k)$*, where* b_i*, for* $i = 1, 2, \ldots, k$ *are* slack atoms*. Slack atoms are created using a new predicate* B_i *with arity* $|LV(\mathbf{a})|$ *for every slack variable* w_i *as* $b_i = B_i(l_1, l_2, \ldots, l_{|LV(\mathbf{a})|})$*, with* $i = 1, 2, \ldots, k$*.*

This definition allows to compactly represent the individual quadratizations of many ground potentials with a similar structure. Also, because no particular form of the original or quadratic representation is assumed, any quadratization technique can be used. For instance, applying the ISH quadratization on the first-order potential $-P(X)Q(X,Z)Q(Y,Z)$, as in Example 4, results in the quadratization $-P(X)B_1(Y,X,Z) - Q(X,Z)B_1(Y,X,Z) - Q(Y,Z)B_1(Y,X,Z) + 2B_1(Y,X,Z)$.

To obtain a convenient quadratization of the whole model the following remarks are useful.

Remark 3. *Because each quadratization is performed with new first-order slack predicates, different parfactors are never grounded to expressions with the same ground slack atom.*

Remark 4. *Slack predicates are applied to all logical variables appearing in the expression that is quadratized. Consequently, two ground substitutions* θ, θ' *that create the same ground slack atoms also define minimizations over the same function:* $\mathbf{b}\theta = \mathbf{b}\theta' \Rightarrow \rho(\mathbf{a}, \mathbf{b})\theta = \rho(\mathbf{a}, \mathbf{b})\theta'$*.*

With the definition of first-order quadratizations, we can express the quadratization of the pseudo-Boolean probabilistic first-order logic model with parfactors G as

$$
\begin{aligned}
P(\mathbf{x}) &= \frac{1}{Z} \exp\left(\sum_{g \in G} \sum_{\theta \in gr(L_g : C_g)} \phi(\mathbf{a}_g)\theta \right) \\
&= \frac{1}{Z} \exp\left(-\sum_{g \in G} \sum_{\theta \in gr(L_g : C_g)} -\phi(\mathbf{a}_g)\theta \right) \\
&= \frac{1}{Z} \exp\left(-\sum_{g \in G} \sum_{\theta \in gr(L_g : C_g)} \min_{\mathbf{b}_g} \rho_g(\mathbf{a}_g, \mathbf{b}_g)\theta \right) \\
&= \frac{1}{Z} \exp\left(-\sum_{g \in G} \sum_{\theta \in gr(L_g : C_g)} \min_{\mathbf{b}_g\theta} \rho_g(\mathbf{a}_g\theta, \mathbf{b}_g\theta) \right) \qquad (7) \\
&= \frac{1}{Z} \exp\left(-\min_{\mathbf{w}} \sum_{g \in G} \sum_{\theta \in gr(L_g : C_g)} \rho_g(\mathbf{a}_g\theta, \mathbf{b}_g\theta) \right), \qquad (8)
\end{aligned}
$$

which can be optionally expressed as a maximization

$$
P(\mathbf{x}) = \frac{1}{Z} \exp\left(\max_{\mathbf{w}} \sum_{g \in G} \sum_{\theta \in gr(L_g : C_g)} -\rho_g(\mathbf{a}_g\theta, \mathbf{b}_g\theta) \right),
$$

where $\mathbf{w}$ contains all ground slack variables $\{\mathbf{b}_g\theta|\theta \in gr(L_g : C_g), g \in G\}$. The step from (7) to (8) follows because the sets of variables produced by different grounding substitutions are either identical or disjoint. For different parfactors, the minimizations are always independent, as implied by Remark 3. For a single parfactor g and two groundings $\min_{\mathbf{b}_g\theta} \rho(\mathbf{a}_g\theta, \mathbf{b}_g\theta)$ and $\min_{\mathbf{b}_g\theta'} \rho(\mathbf{a}_g\theta', \mathbf{b}_g\theta')$, either $\mathbf{b}_g\theta \neq \mathbf{b}_g\theta'$ or $\mathbf{b}_g\theta = \mathbf{b}_g\theta'$. In the first case, we can combine two independent minimizations as $\min_{\mathbf{b}_g\theta, \mathbf{b}_g\theta'} \rho(\mathbf{a}_g\theta, \mathbf{b}_g\theta) + \rho(\mathbf{a}_g\theta', \mathbf{b}_g\theta')$. In the second case, Remark 4 implies $\min_{\mathbf{b}_g\theta} 2\rho(\mathbf{a}_g\theta, \mathbf{b}_g\theta)$.

4.4 Relation to Pairwise MLNs

The concept of *pairwise MLNs* (Fierens et al., 2013) and the one presented here both produce a new model with quadratic parfactors, at the cost of introducing an additional optimization over slack variables. The pairwise MLN approach shows that any MLN can be transformed to a "max-equivalent" MLN with only quadratic clauses, where the concept of max-equivalence is similar to that of (5). To review this approach, first assume that there are no clauses with more than three literals. In this case, each clause with three literals is rewritten into new clauses in *positive normal form*, which are conjunctions of the three positive literals. From this form, a transformation is suggested that splits the conjunction of the three positive literals into new rules that involve only up to two atoms and use a new auxiliary slack atom. For this third-order case, the procedure described above is a special case of our approach. Namely, we show that it is equivalent to using the ISH quadratization in Section 4.2.1 on an MLN in pseudo-Boolean form.

Consider an MLN rule with weight $\alpha > 0$ as the parfactor $(\varnothing, (\alpha, P(\mathbf{t}_1) \wedge Q(\mathbf{t}_2) \wedge R(\mathbf{t}_3)))$. The pairwise MLN approach would transform this into the parfactors with quadratic potential functions

$$\{(\alpha, P(\mathbf{t}_1) \wedge B(\mathbf{t}_4)), (\alpha, Q(\mathbf{t}_2) \wedge B(\mathbf{t}_4)), (\alpha, R(\mathbf{t}_3) \wedge B(\mathbf{t}_4)), (-2\alpha, B(\mathbf{t}_4))\}. \tag{9}$$

The parfactor can also be represented in pseudo-Boolean form as $(\varnothing, (\alpha P(\mathbf{t}_1) Q(\mathbf{t}_2) R(\mathbf{t}_3)))$. This allows to use the ISH quadratization presented in Example 4 as

$$\begin{aligned}
\alpha P(\mathbf{t}_1) Q(\mathbf{t}_2) R(\mathbf{t}_3) &= -\left(-\alpha P(\mathbf{t}_1) Q(\mathbf{t}_2) R(\mathbf{t}_3)\right) \\
&= -\min_{B(\mathbf{t}_4)} \left(-\alpha P(\mathbf{t}_1) B(\mathbf{t}_4) - \alpha R(\mathbf{t}_3) B(\mathbf{t}_4) - \alpha Q(\mathbf{t}_2) B(\mathbf{t}_4) + 2\alpha B(\mathbf{t}_4)\right) \\
&= \max_{B(\mathbf{t}_4)} (\alpha P(\mathbf{t}_1) B(\mathbf{t}_4) + \alpha R(\mathbf{t}_3) B(\mathbf{t}_4) + \alpha Q(\mathbf{t}_2) B(\mathbf{t}_4) - 2\alpha B(\mathbf{t}_4)).
\end{aligned}$$

It can observed that the each of the terms of the potential function correspond one-to-one to the ones created by the pairwise MLN approach in (9). In fact, decomposing this parfactor into new ones for each of the quadratic terms would then give the same form as in pairwise MLN. An analogous procedure for the case where $\alpha < 0$ shows that for the special case with three literals, pairwise MLNs are replicated in pseudo-Boolean form by expressing the formulas in multi-linear form and then using the ISH quadratization for the third-order case.

Although the ISH quadratization is often recommended in the computer vision literature for the case where $\alpha > 0$ (in a maximization setting), it is known not to lead to the tightest roof dual relaxations when $\alpha < 0$ (Fix et al., 2011; Gallagher et al., 2011). In the

experimental section we extend such observations to problems in the domain of probabilistic logic and verify that other quadratization methods lead to improvements in inference quality.

If the model contains clauses with $n > 3$ literals, the pairwise MLN approach suggests to use auxiliary atoms to recursively transform such clauses into new ones with $n - 1$ literals, until no clause has more than three literals. For instance, a clause with $n = 4$ literals is transformed into two clauses with three literals, one of which has an infinite weight. At this point, the procedure from above can be applied to each of the third-order clauses, creating a total of 3 slack atoms.

In contrast, our approach works on the pseudo-Boolean representation of the potential. In this case, the additional preprocessing to first transform the clause to third-order is not required, as there are quadratizations for pseudo-Boolean functions of any order. Recall that clauses can be compactly represented in pseudo-Boolean form using (3). Consequently, the ISH method can be used to quadratize clauses with n literals with either one or $\left\lfloor \frac{n}{2} \right\rfloor$ slack atoms, depending on the sign and whether the optimization is formulated as a maximization or a minimization. For the particular case of a clause with four literals only a single slack atom is required. Because each slack atom can have many propositionalizations, reducing the number of slack atoms creates a large difference in the number of propositional slack variables. Finally, our approach does not create 'hard' rules, which can lead to bad conditioning for some optimization algorithms.

4.5 Normalization of the Parfactors before Quadratization

Consider the parfactor $(\varnothing, P(X)P(Y)Q(Z) + P(Z)Q(Z))$ and true evidence for predicate Q, $Q_T \neq \varnothing$. Although the potential is cubic in general, there are particular ground substitutions for which it is quadratic. This is the case for all groundings for which $Q(Z)\theta \in Q_T$, as for those cases the potential simplifies to $P(X)P(Y) + P(Z)$. The potential is also quadratic for ground substitutions for which $X\theta = Y\theta$, because it then becomes $P(X)Q(Z) + P(Z)P(Z)$. Finally, groundings for which $Y\theta = Z\theta$ have the cubic potential $P(X)P(Z)Q(Z) + P(Z)Q(Z)$, which factorizes to $(P(X)+1)P(Z)Q(Z)$. For such groundings, a quadratization exploiting the structure of the potential could be computed.

These examples highlight that the structure of the potentials can change for particular groundings, and that some might not even require slack variables to become quadratic. This motivates putting the parfactors in a form for which the potentials have the same structure for all groundings. The effect of such a normalization is a reduction in the number of slack variables and the possibility to compute quadratizations tailored to each different form of the potentials. This can be achieved by combining two types of splittings of the parfactors, creating an equivalent first-order representation that makes different forms of potentials explicit at first-order level.

4.5.1 Splitting on Atoms

The first preprocessing method simplifies a parfactor by incorporating evidence for an atom $P(\mathbf{t})$. It proceeds by replacing the parfactor with three new ones, each with an additional constraint $\mathcal{C}_{\in}(\mathbf{t}, P_T), \mathcal{C}_{\in}(\mathbf{t}, P_F), \mathcal{C}_{\in}(\mathbf{t}, P_U)$. Because together these constraints cover all possible ground substitutions for the atom, the new parfactors produce the same groundings as the original one, and the ground model is not changed. Importantly, the potentials of

the parfactors with constraints $\mathcal{C}_{\in}(\mathbf{t}, P_T), \mathcal{C}_{\in}(\mathbf{t}, P_F)$ can be simplified by replacing the atom $P(\mathbf{t})$ by *True* and *False*, respectively.

Definition 4 (Fixed atoms)**.** *An atom a with predicate P is* fixed *if $\mathcal{C}_{\in}(\mathbf{t}, P_T), \mathcal{C}_{\in}(\mathbf{t}, P_F)$ or $\mathcal{C}_{\in}(\mathbf{t}, P_U)$ is in the constraint set C.*

For a parfactor with n atoms, fixing all atoms can create up to 3^n new parfactors. In practice, the number is smaller because a) the potential may become 0 or constant after the simplification (Shavlik & Natarajan, 2009) or b) some of the sets P_T, P_F, P_U may be empty and have no groundings associated to it.

4.5.2 Splitting on the Logical Variables

Atoms constructed with the same predicate may ground to the same ground atom under specific grounding substitutions. For example, the parfactor $(\varnothing, P(X)P(Ann))$ represents a quadratic potential for all ground substitutions of X except when $\theta = \{X \to Ann\}$, when it becomes a linear potential. These simplifications can be identified before grounding by observing the form in which atoms of the same predicate may unify, and splitting the parfactor accordingly. The example creates the parfactors $(X \neq Ann, P(X)P(Ann))$ and $(\varnothing, P(Ann))$. Splitting on the logical variables can be repeated until the unification of atoms is no longer possible. This ensures that any valid ground substitution for a parfactor maps the atoms of the parfactor to distinct ground atoms.

Example 6. *Consider the parfactor $(\varnothing, P(X,Y)P(X,Z)\overline{Q(X)}\,\overline{S(Z)})$ and assume that all predicates have arguments from domain D. No knowledge is available for P, i.e., ($P_T = \varnothing$, $P_F = \varnothing$, $P_U = D \times D$), there is evidence for some groundings of Q, i.e., ($Q_T \subset D$, $Q_F = \varnothing$,$Q_U = D \setminus Q_T$), and full knowledge of the groundings of S, i.e., ($S_T \subset D$, $S_F \subset D$, $S_U = \varnothing$).*

We start by splitting the parfactor on the atom $S(Z)$, which is fully observed. The case where $S(Z)$ is constrained to be True *cancels the potential and is left out. Because $S_U = \varnothing$, the case with constraint $\mathcal{C} \in (X, S_U)$ can also be ignored. If $S(X)$ is constrained to be* False*, the potential simplifies to $P(X,Y)P(X,Z)\overline{Q(X)}$. Proceeding similarly for the other atoms, there remains one parfactor with the simplified potential and additional constraints $\mathcal{C}_{\in}(Z, S_F)$, $\mathcal{C}_{\in}(X, Q_U)$, $\mathcal{C}_{\in}((X,Y), P_U)$, $\mathcal{C}_{\in}((X,Z), P_U)$.*

After fixing all atoms, we observe that the two atoms constructed from predicate P produce the same ground substitutions if $Y = Z$. Splitting on this condition produces two new parfactors, where the potential simplifies to $P(X,Y)\overline{Q(X)}$ for the case where $Y = Z$. After both steps, the original parfactor is transformed into two new parfactors:

$$
\begin{aligned}
\phi(\mathbf{a}) =& P(X,Y)P(X,Z)\overline{Q(X)} \\
C =& \{\mathcal{C}_{\in}(Z, S_F), \mathcal{C}_{\in}(X, Q_U), \mathcal{C}_{\in}((X,Y), P_U), \mathcal{C}_{\in}((X,Z), P_U), Y \neq Z\} \\
& \textit{and} \\
\phi(\mathbf{a}) =& P(X,Y)\overline{Q(X)} \\
C =& \{\mathcal{C}_{\in}(Y, S_F), \mathcal{C}_{\in}(X, Q_U), \mathcal{C}_{\in}((X,Y), P_U)\}
\end{aligned}
$$

Note that after the preprocessing step, the potential function has gone from being a quartic to a cubic function. Computing the quadratization of the potential in the original form and substituting all possible groundings produces $|D|^3$ ground slack atoms. In contrast, splitting on the variables results in fewer ground substitutions for the parfactor, since they are curtailed by each of the constraints (and so is the number of ground slack variables).

4.6 Benefits and Limitations of First-Order Quadratization

Except for the ISH method, the introduced methods compute a quadratization considering the interactions between the terms in the expression. By considering the interactions, these methods can create a quadratization whose roof dual relaxation is tighter, producing better bounds and more persistencies.

However, these methods also need to do additional processing on the problem. When this processing is performed on a propositional model, it may become very large. For instance, Gallagher et al. (2011) suggested to implement the ASM method by constructing a second Markov Random Field in which the variables represent the possible quadratizations for each term. Also, the approach based on generalized roof duality needs to solve an auxiliary linear program to find the best submodular relaxation. The FIX approach does not solve an optimization, but needs to perform multiple transformation steps on the model, and it is not clear in which order to perform them.

Instead, for MLNs, the general procedure presented in Section 4 can be used to obtain a quadratic MLN by applying quadratization techniques on individual parfactors. An advantage of this procedure is that the computational cost of obtaining the quadratization is independent of the size of the ground problem. Furthermore, the parfactors provide a compact description of the structure of the ground model, which can be exploited with the more sophisticated quadratization methods. The auxiliary problems required for some of the quadratization methods become extremely small when solved for individual parfactors, but still produce a good quadratic potential for all of the groundings of the parfactor.

On the other hand, the parfactor representation of the model is not unique, and thus the resulting quadratic MLN may depend on the manipulations performed on the parfactors. In fact, shattering the model may give a more representative template of the structure of the ground problem. In this work, we merge parfactors that have the same constraints after the preprocessing step, but do not take any further steps to partition or combine the parfactors. In Section 5 we evaluate each of these quadratization methods separately. However, in practice methods may be combined with no restrictions, and it is possible that the combination of different methods may give the best results.

4.7 Quadratize-Solve-Simplify-Repeat (QSSR)

When the QPBO algorithm is applied to a general quadratic minimization problem, there can be variables for which no persistency is found. As discussed in Section 3.4, in those cases the *probe* and *improve* procedures can be used to increase the number of persistencies or compute an approximate solution. However, their computational cost can make them inefficient on problems with a large number of unsolved variables. In quadratized models, this can be mitigated by observing that many of the slack variables are no longer necessary after computing the persistencies given by the roof duality. This is because fixing a variable

x to its persistent value in the higher-order problem simplifies or cancels any terms containing x. Thus, constructing a new problem that incorporates the information about the solved variables can have a much smaller number of slack variables, and be used to obtain additional persistencies. This procedure of *quadratizing* the model on the first-order level, *solving* the problem partially with the QPBO algorithm and *simplifying* the higher-order problem with the obtained persistencies can be *repeated* until no additional persistencies are found. We denote this iterative computational procedure by *Quadratize-Solve-Simplify-Repeat* (QSSR). When using this algorithm in the experiments, we stop after the problem has been solved again after the first simplification, as additional iterations generally produce few additional persistencies in comparison to *probing*. Kahl and Strandmark (2012) used a similar procedure of iteratively solving and fixing variables in the context of computing the generalized roof dual bound.

5. Computational Experiments

The presented quadratization methods are evaluated by their impact on the performance of the QPBO algorithm and its extensions. The performance of the QPBO-based inference is also evaluated on problems for which no quadratization is required. Finally we compare our overall pipeline to existing inference engines.

5.1 Datasets

We evaluate our approach on various standard MLNs and datasets as well as additional problems. The characteristics of these problems are summarized in Table 1. A first set of datasets is similar to the ones employed in the evaluation of state-of-the-art engines Tuffy (Niu et al., 2011) and RockIt (Noessner et al., 2013). The link prediction problem on the UWCSE dataset (LP) tries to find relations between faculty members and students. The relational classification (RC) on the Cora dataset determines the category of research papers. The information extraction (IE) problem models how to obtain dataset records from parsed sources. The webKB dataset is used to predict to which university department a website belongs, given its hyperlink relations and contained words (KB). The entity resolution (ER) problem on the Cora dataset is obtained from the Alchemy website. The goal of this problem is to identify citations referring to the same paper. Because no trained model is available for this problem, it is trained with Alchemy using the first of the five available splits for evaluation (Singla & Domingos, 2006a). Friends and smokers (F&S) is a common test model for a social network with friendship relations, smoking habits and cancer occurrences. Evidence is generated as described by Singla and Domingos (2008) for a domain size of 200 persons. Because the F&S problem is relatively simple, an additional problem in which the weights of all formulas are negated is also considered (-F&S).

In order to gain a broader insight into the performance of the inference algorithms on higher-order problems, we created two additional third-order problems. The first one is based on the KB problem and the webKB dataset mentioned above (KB3). While the original KB inference problem uses words contained in the page contents as well as the link structure to infer page categories, a third-order problem on the webKB dataset is created by not only querying for the class of each page, but by also jointly inferring the links of a page, solely from the word tokens appearing on the page. Learning was performed with

Alchemy, and the size of the problem was reduced such that inference is only performed over atoms that are *True* in the ground truth and the same number of randomly sampled atoms.

The second new third-order problem is the image denoising (ID) model, which tries to restore a noisy binary image. There are rules indicating that the observed value of a pixel should correspond to the denoised value and two rules indicating that groups of three horizontally or vertically neighbouring pixels should take the same value. It is easy to see that the associated MAP problem for these rules fall within the described cases of MLNs whose rules can be converted to polar pseudo-Boolean functions described in Section 3.3.2, and can thus be solved exactly. The unary rules are given weight 1.0. To ensure that the terms of the smoothing rules do not cancel out, a rule for the 'on' pixels is given a weight of 0.35 and the rule for the 'off' pixels 0.3. A 90×90 pixels random binary image is used as evidence, where each pixel has a 50% chance of being on or off.

5.2 Other Engines

We compare our approach with the MAP-inference solvers Alchemy, Tuffy and RockIt. Alchemy is the original solver for MLNs and, in contrast to the other engines, it does not use a relational database to ground the model, which can lead to long grounding times. Alchemy and Tuffy optimize the ground model using MaxWalkSAT, a stochastic search technique that can be made to scale well with large problems. RockIt uses an ILP solver and exploits symmetries in the model to reduce the number of constraints. Because the number of constraints may be very large, it takes an iterative approach where only the constraints that are violated for the current solution are added to the solver.

To make sure that the problem to be solved is the same for all implementations, some preprocessing is required. First, formulas with existential clauses are ignored and all formulas are converted to conjunctive normal form. Then, because Tuffy internally transforms formulas with a negative weight to an approximate formula with a positive weight, we apply the same transformation. Unfortunately, this transformation can not be applied for the higher-order problems, as it reduces the order of the formula. Lastly, the ER and KB3 problems use a method to compactly specify which ground atoms to query, and assumes that all other query atoms are *False*. These query variables, also known as canopies (Singla & Domingos, 2006a), can be used to eliminate a large number of uninteresting variables, and can be created using a cheap distance metric (McCallum, Nigam, & Ungar, 2000). Because neither Tuffy nor RockIt support this input format, they are given the extensive list of *False* evidence atoms instead.

5.3 Results on Quadratic Problems

For quadratic problems from literature, we analyze the performance of QPBO and the additional persistencies computed with the probing extension described in Section 3.4. Table 2 shows that the QPBO algorithm gives a persistent solution for most variables, and even provides an exact solution for the KB problem. The *probe* procedure also solves the IE problem exactly, but still leaves some unsolved variables for the RC and LP problems. In general, the inference times for these problems are extremely short.

5.4 Comparison of Quadratization Methods

For the higher-order problems the performance of each of the described quadratization methods is evaluated. This includes the pairwise MLN approach, which is equivalent to the ISH quadratization for problems with cubic potentials. The potentials of the parfactors are expressed as a multi-linear polynomial before quadratizing the model.

First we evaluate the number of persistencies that can be obtained for the different problems in Table 3. As expected, because of submodularity, the ID problem is completely solved by all methods. Friends and Smokers creates few non-submodular terms, and can also be solved exactly by all methods. The remaining problems can not be solved by all methods, for which in some cases only a small number of variables can be fixed. In general, it can be observed that the methods that are aware of the other terms in the potential produce better results than ISH, which applies a fixed transformation. The final probing step is computationally the most expensive, but may significantly increase the number of solved variables, and even solve some problems exactly. It should be noted that this step is important even when an approximate solution for all variables is subsequently obtained using the improve method. Otherwise, if the number of persistencies is small, the improve method needs to operate on a model with potentially many more variables, as it also needs to optimize the slack variables stemming from the remaining higher-order terms.

Table 1: Summary of the characteristics of the described datasets and the associated ground networks when grounded in their higher-order form and a multi-linear representation. Trivially satisfied or dissatisfied factors are ignored.

	IE	**KB**	**RC**	**LP**	**ID**	**F&S**	**-F&S**	**KB3**	**ER**
Formulas	1024	106	15	24	4	6	6	66	1331
Domains	4	3	3	8	1	1	1	3	5
Query Predicates	2	1	1	1	1	3	3	2	4
Observed Predicates	16	2	3	21	2	0	0	1	6
Ground atoms	336670	9079	9650	4624	8100	40180	40180	8190	10948
Factors	351001	31283	58485	161806	55800	127982	127982	22627	910670
Higher order factors	0	0	0	0	15840	32220	32220	6736	424580

Table 3: Percentage of variables solved. Step 1) Initial QPBO result 2) QPBO result after QSSR simplification 3) Probe. Inference time in seconds for each step in parentheses. (†) Did not complete

	Step	ISH	FIX	ASM	GRD
ID	1	100.0 (0.01)	100.0 (0.01)	100.0 (0.01)	100.0 (0.01)
F&S	1	100.0 (0.02)	100.0 (0.82)	100.0 (0.86)	99.6 (1.0)
	2				100.0 (0.0)
-F&S	1	19.4 (1.79)	19.4 (0.9)	99.6 (0.42)	99.6 (1.11)
	2	19.4 (1.61)	19.4 (0.78)	99.6 (0.02)	99.6 (0.02)
	3	†	†	99.6 (2.66)	99.6 (2.54)
KB3	1	55.0 (0.06)	82.4 (0.03)	82.3 (0.02)	60.6 (0.08)
	2	56.5 (0.04)	86.8 (0.01)	86.6 (0.01)	62.7 (0.04)
	3	65.8 (19.42)	100.0 (0.1)	100.0 (0.05)	96.7 (5.59)
ER	1	92.1 (0.46)	91.9 (1.04)	95.0 (0.47)	95.4 (1.49)
	2	92.3 (0.03)	92.3 (0.04)	95.0 (0.02)	96.1 (0.03)
	3	92.8 (7.6)	93.0 (162.36)	95.3 (5.57)	96.6 (7.08)

5.5 Approximate Inference

We also compared quality of the approximate solutions with those of other engines and their total running times. The problems were formulated as minimizations, and the solutions of all engines evaluated on the same ground model. In Table 4 it can be observed that for the quadratic problems, most engines achieve optimal costs, which are known from the optimality guarantee given by QPBO in Table 2 and from the small MIP gap that was used for RockIt. An exception is the LP problem, where the solver used by RockIt has problems obtaining a tight bound, and Tuffy and QPBO+I provide better solutions.

For the higher-order problems, the ASM quadratization achieves the best cost and the lowest computation time for most cases. Using the GRD reduction performs slightly worse, possibly because for this quadratization the improve step needs to be executed over more

Table 2: Percentage of persistencies given by the QPBO algorithm and after using the probing technique for different quadratic problems.

	IE	KB	RC	LP
Persistencies	99.87	100	90.30	85.58
Persistencies (probe)	100		90.30	86.22
Qpbo time (s)	0.030	0.002	0.006	0.069
Probe time (s)	0.150		0.021	4.800

variables. Tuffy does not perform very well in the higher order problems, possibly because the internal transformation it uses is an approximation of the original formula.

It should be noted that computation times are affected by multiple factors. Whereas Alchemy, Tuffy and our approach make a clear distinction between grounding and inference, RockIt uses a cutting plane algorithm that incrementally grounds factors that are not satisfied by the current solution, which leads to large speedups when many factors are easily satisfied or if the solution is largely homogeneous. On the other hand, the ID problem is an example where this approach produces considerably longer running times. Another factor is the ability to specify the evidence in the form of canopies, which allows the relational database to execute the queries for grounding more efficiently.

Table 4: Resulting cost for different engines on various quadratic and higher-order problems. Alchemy and Tuffy were run for an increasing number of flips until no significant advances were made. RockIt was run with relative gaps 1×10^{-n}, $n = 9, 8, \ldots$ until convergence is achieved within an hour. These are compared against our method using the ASM and GRD quadratization for the higher-order problems, using improve on the residual problem until no advances were made for 20 iterations. Total running times in seconds in parenthesis. (*) Guaranteed optimal cost by persistencies (†) Did not ground within 1 hour.

	Alchemy		**Tuffy**		**RockIt**		**QPBO+I**			
IE	†		**-4511.6**	(17)	**-4511.6**	(19)	**-4511.6**	(22)		
KB	-111113.5	(162)	-111274.1	(115)	**-111312.4***	(27)	**-111312.4***	(6)		
RC	†		-4031.7	(17)	**-4031.8**	(11)	**-4031.8**	(9)		
LP	-480.8	(119)	-686.3	(424)	-507.7	(13)	**-732.6**	(9)		
							ASM+QPBO+I		**GRD+QPBO+I**	
ID	1772.7	(442)	1784.2	(25)	**-1003.8***	(244)	**-1003.8***	(5)	**-1003.8***	(6)
F&S	-3.8	(159)	**-4.2***	(3)	**-4.2***	(5)	**-4.2***	(6)	**-4.2***	(9)
-F&S	-182338.7	(47)	-191856.9	(3230)	-185267.3	(8)	**-193715.3**	(12)	**-193715.3**	(14)
KB3	21.1	(543)	-1045.3	(308)	**-1492.8**	(256)	-1484.4	(57)	-1476.9	(101)
ER	-10739.5	(551)	-14128.9	(433)	-15271.3	(1902)	**-15430.7**	(101)	-15430.5	(113)

In Figure 2, the evolution of the cost of the ER problem against the running time of *improve* is shown for different quadratizations. For this problem, the methods converge to a solution with similar costs, but convergence is much faster in the cases where ISH and GRD quadratizations are used.

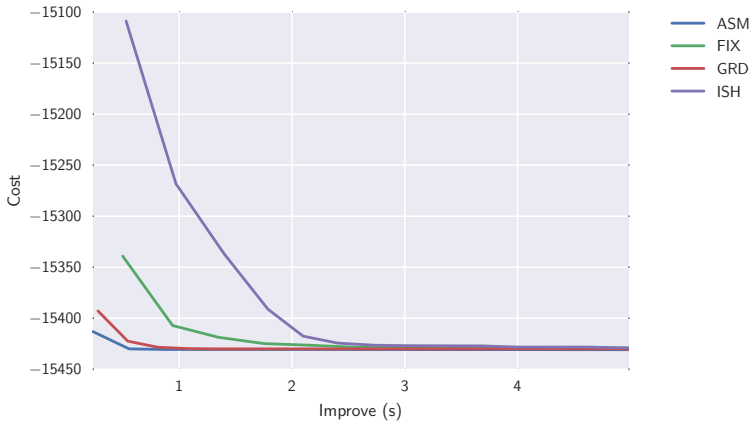

Figure 2: Cost in the higher-order ER model as a function of the time spent on the *improve* method, using different quadratization techniques. Improve starts after solving one iteration of the original problem and removing the redundant slacks, as described in Section 4.7.

6. Conclusions and Future Work

In this article we have discussed the use of pseudo-Boolean functions, quadratization techniques, and MAP inference methods based on roof duality for MLNs. It was shown how any quadratization method for pseudo-Boolean functions can be employed on MLNs at first-order level, generalizing a previously existing approach. This enables the use of quadratization methods that exploit the structure of the problem without the need to solve a possibly large auxiliary optimization problem. Various quadratization approaches were adapted to work with first-order models, including a novel approach that leverages a connection with the generalized roof duality theory.

Additionally, to the best of our knowledge, this is the first work that discusses the recognition of (super-)submodularity and expressibility in MLNs. In particular, this allows to guarantee the expressibility of a restricted set of MLNs. Although the class of potentials that are guaranteed to be expressible using submodular quadratic functions is limited, such potentials have seen wide applications in computer vision.

The presented techniques were extensively evaluated on problems from the literature as well as various additional problems. On higher-order problems, the choice of quadratization approach was shown to be an important factor in the quality of the results. The methods that exploit the first-order representation of the problem often enabled QPBO to solve a larger part of the problem and perform better approximate inference. In all cases, large parts of the problems could be solved exactly, and the approximate solutions matched or improved the quality of other solvers. The optimization times for the problems were observed to be very small, and often much shorter than the rest of the pipeline, for which we did not optimize. In total, timings show very competitive results in relation to other state-of-the-art inference engines.

There are various paths for future work. This work has not explored if performance can be improved by partitioning the model further than the presented shattering techniques, or by using other manipulations on the parfactors. Also, the possibility to obtain better approximate solutions using *move making* algorithms (Lempitsky, Rother, Roth, & Blake, 2010) was not tested, but such algorithms may leverage the first-order representation for finding good moves. An additional interesting aspect to consider is to integrate our approach with cutting plane techniques or lifting techniques, which can be expected to give a significant performance benefit for problems with a very large number of factors.

Acknowledgments

The authors also wish to thank the anonymous reviewers for their valuable comments and helpful suggestions. The research leading to these results has partly received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 26787, Project SHRINE, and the Technische Universität München - Institute for Advanced Study (`www.tum-ias.de`), funded by the German Excellence Initiative.

References

Apsel, U., & Brafman, R. I. (2012). Exploiting uniform assignments in first-order MPE. In *Proc. Conf. Uncertainty in Artificial Intelligence*, pp. 74–83.

Beedkar, K., Del Corro, L., & Gemulla, R. (2013). Fully parallel inference in markov logic networks. In *15th GI-Symposium Database Systems for Business, Technology and Web*, Magdeburg, Germany. Bonner Köllen.

Billionnet, A., & Minoux, M. (1985). Maximizing a supermodular pseudoboolean function: A polynomial algorithm for supermodular cubic functions. *Discrete Applied Mathematics*, *12*(1), 1–11.

Boros, E., Hammer, P. L., & Tavares, G. (2006). Preprocessing of unconstrained quadratic binary optimization. Tech. rep., Rutgers Center for Operations Research.

Boros, E., & Hammer, P. (2002). Pseudo-boolean optimization. *Discrete Applied Mathematics*, *123*(1), 155–225.

Crama, Y. (1989). Recognition problems for special classes of polynomials in 0–1 variables. *Mathematical Programming*, *44*(1-3), 139–155.

Darwiche, A. (2003). A differential approach to inference in bayesian networks. *J. of the ACM*, *50*(3), 280–305.

de Salvo Braz, R. (2007). *Lifted First-Order Probabilistic Inference*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Domingos, P., & Webb, W. A. (2012). A tractable first-order probabilistic logic.. In *Proc. Conf. Artificial Intelligence*. AAAI.

Eaton, F., & Ghahramani, Z. (2013). Model reductions for inference: Generality of pairwise, binary, and planar factor graphs. *Neural computation*, *25*(5), 1213–1260.

Fierens, D., Kersting, K., Davis, J., Chen, J., & Mladenov, M. (2013). Pairwise markov logic. In *Inductive Logic Programming*, pp. 58–73. Springer.

Fix, A., Gruber, A., Boros, E., & Zabih, R. (2011). A graph cut algorithm for higher-order Markov random fields. In *Int. Conf. Computer Vision*, pp. 1020–1027. IEEE.

Freedman, D., & Drineas, P. (2005). Energy minimization via graph cuts: Settling what is possible. In *Conf. Computer Vision and Pattern Recognition*, pp. 939–946. IEEE Computer Society.

Gallagher, A. C., Batra, D., & Parikh, D. (2011). Inference for order reduction in markov random fields. In *Proc. Int. Conf. Computer Vision and Pattern Recognition*, pp. 1857–1864. IEEE.

Gallo, G., & Simeone, B. (1989). On the supermodular knapsack problem. *Mathematical Programming*, *45*(1-3), 295–309.

Huang, J., Chavira, M., & Darwiche, A. (2006). Solving MAP exactly by searching on compiled arithmetic circuits. In *Proc. Conf. Artificial Intelligence*, Vol. 6, pp. 3–7. AAAI.

Ishikawa, H. (2011). Transformation of general binary MRF minimization to the first-order case. *Trans. Pattern Analysis and Machine Intelligence*, *33*(6), 1234–1249.

Kahl, F., & Strandmark, P. (2012). Generalized roof duality. *Discrete Applied Mathematics*, *160*(16-17), 2419–2434.

Kolmogorov, V., & Rother, C. (2007). Minimizing nonsubmodular functions with graph cuts-a review. *Trans. Pattern Analysis and Machine Intelligence*, *29*(7), 1274–1279.

Kolmogorov, V. (2012). Generalized roof duality and bisubmodular functions. *Discrete Applied Mathematics*, *160*(4-5), 416–426.

Kolmogorov, V., & Zabin, R. (2004). What energy functions can be minimized via graph cuts?. *Trans. Pattern Analysis and Machine Intelligence*, *26*(2), 147–159.

Lempitsky, V., Rother, C., Roth, S., & Blake, A. (2010). Fusion moves for markov random field optimization. *Trans. Pattern Analysis and Machine Intelligence*, *32*(8), 1392–1405.

McCallum, A., Nigam, K., & Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. Int. Conf. Knowledge Discovery and Data Mining*, pp. 169–178. ACM.

Mittal, H., Goyal, P., Gogate, V. G., & Singla, P. (2014). New rules for domain independent lifted MAP inference. In *Proc. Advances in Neural Information Processing Systems*, pp. 649–657.

Niu, F., Ré, C., Doan, A., & Shavlik, J. (2011). Tuffy: Scaling up statistical inference in markov logic networks using an RDBMS. *Proc. VLDB Endowment*, *4*(6), 373–384.

Noessner, J., Niepert, M., & Stuckenschmidt, H. (2013). Rockit: Exploiting parallelism and symmetry for MAP inference in statistical relational models.. In *AAAI Workshop: Statistical Relational Artificial Intelligence*.

Orlin, J. B. (2009). A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, *118*(2), 237–251.

Poole, D. (2003). First-order probabilistic inference. In Gottlob, G., & Walsh, T. (Eds.), *Int. Joint Conf. Artificial Intelligence*, pp. 985–991. Morgan Kaufmann.

Ramalingam, S., Russell, C., Ladicky, L., & Torr, P. H. S. (2011). Efficient minimization of higher order submodular functions using monotonic boolean functions. *CoRR*, *abs/1109.2304*.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine learning*, *62*(1), 107–136.

Riedel, S. (2009). Cutting plane map inference for markov logic. In *Int. Workshop Statistical Relational Learning*.

Rosenberg, I. (1975). Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d'etudes de Recherche Operationnelle*, *17*, 71–74.

Rother, C., Kolmogorov, V., Lempitsky, V., & Szummer, M. (2007). Optimizing binary MRFs via extended roof duality. In *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 1–8. IEEE.

Sarkhel, S., Venugopal, D., Singla, P., & Gogate, V. (2014). Lifted MAP inference for markov logic networks. In *Proc. Int. Conf. Artificial Intelligence and Statistics*, pp. 859–867.

Schlesinger, D. (2007). Exact solution of permuted submodular minsum problems. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 28–38. Springer.

Shavlik, J. W., & Natarajan, S. (2009). Speeding up inference in markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *Proc. Int. Joint Conf. Artificial Intelligence*, pp. 1951–1956.

Singla, P., & Domingos, P. (2008). Lifted first-order belief propagation. In *Proc. National Conf. Artificial Intelligence*, Vol. 2, pp. 1094–1099.

Singla, P., & Domingos, P. (2006a). Entity resolution with markov logic. In *Proc. Int. Conf. Data Mining*, pp. 572–582. IEEE.

Singla, P., & Domingos, P. (2006b). Memory-efficient inference in relational domains. In *Proc. National Conf. Artificial Intelligence*, Vol. 21, pp. 488–493.

Živnỳ, S., Cohen, D. A., & Jeavons, P. G. (2009). The expressive power of binary submodular functions. *Discrete Applied Mathematics*, *157*(15), 3347–3358.

Živnỳ, S., & Jeavons, P. G. (2008). Which submodular functions are expressible using binary submodular functions?. Tech. rep. CS-RR-08-08, University of Oxford.