RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models

Jan Noessner
Data and Web Science Research Group
University of Mannheim
Mannheim, 68131, Germany

Mathias Niepert
Computer Science & Engineering
University of Washington
Seattle, WA 98195-2350, USA

Heiner Stuckenschmidt
Data and Web Science Research Group
University of Mannheim
Mannheim, 68131, Germany

Abstract

RockIt is a maximum a-posteriori (MAP) query engine for statistical relational models. MAP inference in graphical models is an optimization problem which can be compiled to integer linear programs (ILPs). We describe several advances in translating MAP queries to ILP instances and present the novel meta-algorithm cutting plane aggregation (CPA). CPA exploits local context-specific symmetries and bundles up sets of linear constraints. The resulting counting constraints lead to more compact ILPs and make the symmetry of the ground model more explicit to state-of-the-art ILP solvers. Moreover, RockIt parallelizes most parts of the MAP inference pipeline taking advantage of ubiquitous shared-memory multi-core architectures.

We report on extensive experiments with Markov logic network (MLN) benchmarks showing that RockIt outperforms the state-of-the-art systems Alchemy, Markov TheBeast, and Tuffy both in terms of efficiency and quality of results.

Introduction

Maximum a-posteriori (MAP) queries in statistical relational models ask for a most probable possible world given evidence. In this paper, we will present novel principles and algorithms for solving MAP queries in SRL models. MAP inference (or, alternatively, MPE inference) is an important type of probabilistic inference problem in graphical models which is also used as a subroutine in numerous weight learning algorithms (Lowd and Domingos 2007). Being able to answer MAP queries more efficiently often translates to improved learning performance.

Since Markov logic (Richardson and Domingos 2006) is arguably the most widely used statistical relational learning (SRL) formalism, we use Markov logic as representation formalism throughout this paper. However, the proposed approach is also applicable to numerous alternative SRL formalisms. Due to its expressiveness and declarative nature, numerous real-world problems have been modeled with Markov logic. Especially in the realm of data management applications such as entity resolution (Singla and Domingos 2006), data integration (Niepert, Meilicke, and Stuckenschmidt 2010), ontology refinement (Wu and Weld 2008), and information extraction (Poon and Domingos 2007), Markov logic facilitates rapid prototyping and achieves competitive empirical results.

The main contributions of the presented work are as follows. First, we present a more compact compilation of MAP problems to integer linear programs (ILPs) where each ground clause is modeled by a single linear constraint. Secondly, we introduce cutting plane aggregation (CPA). CPA exploits evidence-induced symmetries in the ground model that lead to context-specific exchangeability of the random variables. CPA results not only in more compact ILPs but it also makes the model’s symmetries more explicit to symmetry detection heuristics of state of the art ILP solvers.

Thirdly, we parallelize most parts of the MAP query pipeline so as to leverage multi-core architectures. Finally, we have designed and implemented the presented theory resulting in a novel and robust MLN engine RockIt that integrates cutting plane aggregation with cutting plane inference. Numerous experiments on established benchmarks show that CPA leads to significantly reduced run times. RockIt outperforms the state-of-the-art systems Alchemy, Markov TheBeast, and Tuffy both in terms of running time and quality of results on each of the MLN benchmarks.

Related Work

MaxWalkSAT (MWS), a random walk algorithm for solving weighted SAT problems (Kautz, Selman, and Jiang 1997), is the standard inference algorithm for MAP queries in the Markov logic engine Alchemy (Domingos et al. 2012). The system Tuffy (Niu et al. 2011) employs relational database management systems to ground Markov logic networks more efficiently. Tuffy also runs MWS on the ground model which it initially attempts to partition into disconnected components.

MAP queries in SRL models can be formulated as integer linear programs (ILPs). In this context, cutting plane inference (CPI) solving multiple smaller ILPs in several iterations has shown remarkable performance (Riedel 2008). In each CPI iteration, only the ground formulas violated by the current intermediate solution are added to the ILP formulation until no violated ground formulas remain. Since CPI ignores ground formulas satisfied by the evidence, it can be
seen as a generalization of pre-processing approaches that count the formulas satisfied by the evidence (Shavlik and Natarajan 2009). In the context of max-margin weight learning for MLNs (Huynh and Mooney 2009) the MAP query was formulated as a linear relaxation of an ILP and a rounding procedure was applied to extract an approximate MAP state. ROCKIT’s ILP formulation requires less constraints for ground clauses with negative weights and it combines CPI with cutting plane aggregation.

There is a large class of symmetry-aware algorithms for SRL models. Examples of such lifted inference algorithms include first-order variable elimination (FOVE) (Poole 2003) and some of its extensions (Milch et al. 2008; Kisynski and Poole 2009) making use of counting and aggregation parfactors. FOVE has also been adapted to solve MAP problems (FOVE-P) (de Salvo Braz, Amir, and Roth 2006). (Apsel and Brafman 2012) introduced an approach for MAP inference that takes advantage of uniform assignments which are groups of random variables that have identical assignments in some MAP solution. Automorphism groups of graphical models were used to lift variational approximations of MAP inference (Bui, Huynh, and Riedel 2012). We attempted to compute automorphism groups as an alternative method for aggregating constraints but experiments showed that calling a graph automorphism algorithm in each CPI iteration dominated the overall solving time. (Mladenov, Ahmadi, and Kersting 2012) computed approximate solutions to linear programs by reducing the LP problem to a pairwise MRF over Gaussians and applying lifted Gaussian belief propagation. Similar to the approach of (Bui, Huynh, and Riedel 2012) lifted linear programming can be used to approximate LP relaxations (Asano 2006) of the MAP ILP. Contrary to previous work, ROCKIT uses a more compact ILP formulation with a one-to-one correspondence between ground clauses and linear constraints, tightly integrates CPI and CPA, and estimates the optimal aggregation scheme avoiding a costly exact computation in each CPI iteration. Moreover, contrary to lifted inference approaches operating solely on the first-order level, ROCKIT exploits evidence-induced local symmetries on the ground level.

There are several lifted marginal inference approaches such as lifted message passing (Singla and Domingos 2008; Kersting, Ahmadi, and Natarajan 2009), variants of lifted knowledge compilation and theorem proving (van den Broeck 2011; Gogate and Domingos 2011), and lifted MCMC (Niepert 2012; Venugopal and Gogate 2012) approaches. While there are some generic parallel machine learning architectures such as GRAPHLAB (Low et al. 2010) which could in principle be used for parallel MAP inference, ROCKIT is the first system that parallelizes MAP inference in SRL models combining CPI and CPA.

Markov Logic
Markov logic is a first-order template language combining first-order logic with log-linear graphical models. We first review function-free first-order logic (Genesereth andNilsson 1987). Here, a term is either a constant or a variable. An atom \( p(t_1, ..., t_n) \) consists of a predicate \( p \) of arity \( n \) followed by \( n \) terms \( t_i \). A literal \( \ell \) is an atom \( a \) or its negation \( \neg a \). A clause is a disjunction \( \ell_1 \lor ... \lor \ell_k \) of literals. The variables in clauses are always assumed to be universally quantified. The Herbrand base \( H \) is the set of all possible ground (instantiated) atoms. Every subset of the Herbrand base is a Herbrand interpretation.

A Markov logic network \( M \) is a finite set of pairs \((F_i, w_i)\), \( 1 \leq i \leq n \), where each \( F_i \) is a clause in function-free first-order logic and \( w_i \in \mathbb{R} \). Together with a finite set of constants \( C = \{c_1, ..., c_n\} \) it defines the ground Markov logic network \( M_C \) with one binary variable for each grounding of predicates occurring in \( M \) and one feature for each grounding of formulas in \( M \) with feature weight \( w_i \). Hence, a Markov logic network defines a log-linear probability distribution over Herbrand interpretations (possible worlds)

\[
P(x) = \frac{1}{Z} \exp \left( \sum_i w_i n_i(x) \right)
\]

where \( n_i(x) \) is the number of satisfied groundings of clause \( F_i \) in possible world \( x \) and \( Z \) is a normalization constant.

In order to answer a MAP query given evidence \( E = e \) one has to solve the maximization problem

\[
\arg \max_x P(X = x \mid E = e)
\]

where the maximization is performed over possible worlds (Herbrand interpretations) \( x \) compatible with the evidence.

Cutting Plane Aggregation
Each MAP query corresponds to an optimization problem with linear constraints and a linear objective function and, hence, we can formulate the problem as an instance of integer linear programming. The novel cutting plane aggregation approach is tightly integrated with cutting plane inference (CPI) a meta-algorithm operating between the grounding algorithm and the ILP solver (Riedel 2008). Instead of immediately adding one constraint for each ground formula to the ILP formulation, the ILP is initially formulated so as to enforce the given evidence to hold in any solution. Based on the solution of this more compact ILP one determines the violated constraints, adds these to the ILP, and resolves. This process is repeated until no constraints are violated by an intermediate solution.

We begin by introducing a novel ILP formulation of MAP queries for Markov logic networks. In contrast to existing approaches (Riedel 2008; Huynh and Mooney 2009), the formulation requires only one linear constraint per ground clause irrespective of the ground clause being weighted or unweighted. Moreover, we introduce the notion of context-specific exchangeability and describe the novel cutting plane aggregation (CPA) algorithm that exploits this type of local symmetry. Contrary to most symmetry-aware and lifted inference algorithms that assume no or only a limited amount of evidence, the presented approach specifically exploits model symmetries induced by the given evidence.

General ILP Formulation
In order to transform the MAP problem to an ILP we have to first ground, that is, instantiate, the first-order theory specified by the Markov logic network. Since we are employing
weight | ground clause | max \(1.1z_1 - 0.5z_2\)  
\[1.1\] \(x_1 \lor \neg x_2 \lor x_3\) \(\sim\) \(x_1 + (1 - x_2) + x_3 \geq z_1\)  
\[-0.5\] \(\neg x_1 \lor x_2\) \((1 - x_1) + x_2 \leq 2 \cdot z_2\)  
\(\infty\) \(\neg x_1 \lor x_2\) \((1 - x_1) + x_2 \geq 1\)

Table 1: An example of the ILP formulation.

cutting plane inference, ROCKIT runs in each iteration several join queries in a relational database system to retrieve the ground clauses violated by the current solution. Hence, in each iteration of the algorithm, ROCKIT maintains a set of ground clauses \(G\) that have to be translated to an ILP instance.

Given such a set of ground clauses \(G\), we associate one binary ILP variable \(x_\ell\) with each ground atom \(\ell\) occurring in some \(g \in G\). For the sake of simplicity, we will often denote ground atoms and ILP variables with identical names. For a ground clause \(g \in G\) let \(\ell^+(g)\) be the set of ground atoms occurring unnegated in \(g\) and \(\ell^-(g)\) be the set of ground atoms occurring negated in \(g\). Now, we encode the given evidence by introducing linear constraints of the form \(x_\ell \leq 0\) or \(x_\ell \geq 1\) depending on whether the evidence sets the corresponding ground atom \(\ell\) to false or true. For every ground clause \(g \in G\) with weight \(w > 0\), \(w \in \mathbb{R}\), we add a novel binary variable \(z_g\) and the following constraint to the ILP:

\[
\sum_{\ell \in \ell^+(g)} x_\ell + \sum_{\ell \in \ell^-(g)} (1 - x_\ell) \geq z_g.
\]

Please note that if any of the ground atoms \(\ell\) in the ground clause is set to false (true) by the given evidence, we do not include it in the linear constraint.

For every \(g\) with weight \(w_g < 0\), \(w \in \mathbb{R}\), we add a novel binary variable \(z_g\) and the following constraint to the ILP:

\[
\sum_{\ell \in \ell^+(g)} x_\ell + \sum_{\ell \in \ell^-(g)} (1 - x_\ell) \leq ((\ell^+(g)) + (\ell^-(g)))z_g.
\]

For every \(g\) with weight \(w_g = \infty\), that is, a hard clause, we add the following linear constraint to the ILP:

\[
\sum_{\ell \in \ell^+(g)} x_\ell + \sum_{\ell \in \ell^-(g)} (1 - x_\ell) \geq 1
\]

If a ground clause has zero weight we do not have to add the corresponding constraint.

Finally, the objective of the ILP is:

\[
\max \sum_{g \in G} w_g z_g,
\]

where we sum over weighted ground clauses only, \(w_g\) is the weight of \(g\), and \(z_g \in \{0, 1\}\) is the binary variable previously associated with ground clause \(g\). We compute a MAP state by solving the ILP whose solution corresponds one-to-one to a MAP state \(x\) where \(x_1 = \text{true}\) if the corresponding ILP variable is 1 and \(x_1 = \text{false}\) otherwise.

For example, Table 1 depicts three clauses with \(w > 0\), \(w < 0\), and \(w = \infty\), and the respective ILP formulations.

**Constraint Aggregation**

In this section we optimize the compilation of sets of weighted ground clauses to sets of linear constraints. More concretely, we introduce a novel approach that aggregates sets of ground clauses so as to make the resulting ILP have (a) fewer variables (b) fewer constraints and (c) its context-specific symmetries more exposed to the ILP solver’s symmetry detection heuristics.

We first demonstrate that evidence often introduces symmetries in the resulting sets of ground clauses and, therefore, at the level of ILP constraints. The proposed approach aggregates ground clauses, resulting in smaller constraint matrices and aiding symmetry detection algorithms of the ILP solvers. The solvers apply heuristics to test whether the ILP’s constraint matrix exhibits symmetries in form of permutations of its columns and rows. For a comprehensive overview of existing principles and algorithms for detecting and exploiting symmetries in integer linear programs we refer the reader to (Margot 2010; Margot 2003; Ostrowski et al. 2011; Bodí, Herr, and Joswig 2013). We describe cutting plane aggregation in two steps. First, we explain the aggregation of ground formulas and, second, we describe the compilation of aggregated formulas to ILP constraints.

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

**Example 1** Table 2 lists a set of 5 ground clauses. The set of clauses \(\{g_1, g_2, g_3\}\) can be aggregated with respect to \(-y_1 \lor y_2\) since we can write each of these ground clauses as \(\ell_1 \lor -y_1 \lor y_2\) with \(\ell_1 := x_1, \ell_2 := x_2\), and \(\ell_3 := -\neg x_3\).

Before we describe the advantages of determining ground clauses that can be aggregated and the corresponding ILP formulation encoding these sets of clauses, we provide a typical instance of a Markov logic network resulting in a large number of clauses that can be aggregated.

**Example 2** Let us consider the clause 
\(-\text{smokes}(x) \lor \text{cancer}(x)\)
and let us assume that there are 100 constants \(C_1, \ldots, C_{100}\) for which we have evidence \(\text{smokes}(C_i), 1 \leq i \leq 100\). For \(1 \leq i \leq 100\), let \(y_i\) be the ILP variable corresponding to the ground atom \(\text{cancer}(C_i)\). The naive formulation would contain 100 constraints \(y_i \geq z_i\) and the objective of the ILP with respect to these clauses would be \(\max 1.5z_1 + \ldots + 1.5z_{100}\). Instead, we can aggregate the ground clauses \(\text{cancer}(C_i), 1 \leq i \leq 100, c = \text{false and } \ell_i = \text{cancer}(C_i)\), \(1 \leq i \leq 100\).

Let \(G \subseteq \mathcal{G}\) be a set of ground clauses with weight \(w\) and let \(c\) be a ground clause. Moreover, let us assume that \(G\) can be aggregated with respect to \(c\), that is, that each \(g \in G\) can be written as \(\ell_i \lor c\). The aggregated feature \(f^{G_c}\) for the clauses \(G\) with weight \(w\) maps each interpretation \(I\) to an integer value as follows.

\[
f^{G_c}(I) = \begin{cases} 
|\{\ell_i \lor c \in G \mid I = \ell_i\}| & \text{if } I = c \\
0 & \text{otherwise}
\end{cases}
\]
The feature resulting from the aggregation, therefore, counts the number of literals \( \ell_i \) that are satisfied if the ground clause \( c \) is not satisfied and returns the number of aggregated clauses otherwise. Please note that an encoding of this feature in a factor graph would require space exponential in the number of ground atoms even though the feature only has a linear number of possible values. The feature, therefore, is highly symmetric – each assignment to the random variables corresponding to the negated (negated) literals that has the same Hamming weight results in the same feature weight contribution. This constitutes a feature-specific local form of finite exchangeability (Finetti 1972; Diaconis 1977) of random variables induced by the evidence. Therefore, we denote this form of finite exchangeability as context-specific exchangeability. Please note that the concept is related to counting formulas used in some lifted inference algorithms (Milch et al. 2008). While standard models such as factor graphs cannot represent such symmetric features compactly, one can encode these counting features directly with a constant number of ILP constraints. We now describe this translation in more detail.

As before, for any ground clause \( c \), let \( L^+(c) (L^-(c)) \) be the set of ground atoms occurring unnegated (negated) in \( c \). We first show the formulation for clauses with positive weights. Let \( G \subseteq \mathcal{G} \) be a set of \( n \) ground clauses with weight \( w < 0 \) that can be aggregated with respect to \( c \), that is, for each \( g \in G \) we have that \( g = x_i \lor c \) or \( g = \neg x_i \lor c \) for some ground atom \( x_i \) and a fixed clause \( c \). We now add the following two linear constraints to the ILP:

\[
\sum_{(x_i, \lor) \in G} x_i + \sum_{(\neg x_i, \lor) \in G} (1 - x_i) + \sum_{\ell \in L^-(c)} n x_\ell + \sum_{\ell \in L^+(c)} n(1 - x_\ell) \geq z_g \tag{2}
\]

and

\[
z_g \leq n \tag{3}
\]

Linear constraint (2) introduces the novel integer variable \( z_g \) for each aggregation. Whenever a solution satisfies the ground clause \( c \) this variable has the value \( n \) and otherwise it is equal to the number of literals \( \ell_i \) satisfied by the solution. Since constraint (2) alone might lead to values of \( z_g \) that are greater than \( n \), the linear constraint (3) ensures that the value of \( z_g \) is at most \( n \). However, linear constraint (3) only needs to be added if clause \( c \) is not the constant false.

We describe the aggregation of clauses with negative weight. Let \( G \subseteq \mathcal{G} \) be a set of \( n \) ground clauses with weight \( w < 0 \) that can be aggregated with respect to \( c \), that is, for each \( g \in G \) we have that \( g = x_i \lor c \) or \( g = \neg x_i \lor c \) for a ground atom \( x_i \) and a fixed clause \( c \). We now add the following linear constraints to the ILP:

\[
\sum_{(x_i, \lor) \in G} x_i + \sum_{(\neg x_i, \lor) \in G} (1 - x_i) \leq z_g \tag{4}
\]

and

\[
n x_\ell \leq z_g \text{ for every } \ell \in L^+(c) \tag{5}
\]

and

\[
n(1 - x_\ell) \leq z_g \text{ for every } \ell \in L^-(c) \tag{6}
\]

Linear constraint (4) introduces an integer variable \( z_g \) that counts the number of ground clauses in \( G \) that are satisfied. For each of the integer variables \( z_g \) representing an aggregated set of clauses we add the term \( w_g z_g \) to the objective function where \( w_g \) is the weight of each of the aggregated clauses. Table 3 shows a set of aggregated ground clauses and the corresponding ILP formulation. It is not difficult to verify that each solution of the novel formulation corresponds to a MAP state of the MLN it was constructed from.

**Example 3** In Example 2 we aggregate the ground clauses cancer(\( C_i \)), \( 1 \leq i \leq 100 \), for \( c \) = false and \( \ell_i \) = cancer(\( C_i \)), \( 1 \leq i \leq 100 \). Now, instead of 100 linear constraints and 100 summands in the objective function the aggregated formulation only adds the linear constraint \( y_1 + \ldots + y_{100} \leq z_g \) and the term \( 1.5 z_g \) to the objective.
We observed that the computation and aggregation of the violated constraints often dominated the ILP solving time. Since, in addition, state-of-the-art ILP solvers such as GUROBI already parallelize their branch and bound based algorithms we developed an additional method for parallelizing the CPI and CPA phases.

**Parallelism and Implementation**

We will (a) briefly explain the parallelization framework of ROCKIT\(^1\) (b) outline the implementation of the aggregation strategy, and (c) provide some details about the third-party components we employed.

Figure [1](#) depicts the computational pipeline of the system. After pre-processing the input MLN and loading it into the relational database system, ROCKIT performs CPI iterations until no new violated constraints are found. The violated constraints are computed with joins in the relational database system where each table stores the predicate groundings of the intermediate solutions. In each CPI iteration, ROCKIT performs CPA on the violated constraints. We can parallelize the aggregation steps by processing each first-order formula in a separate thread. To this end, each first-order formula is initially placed on a stack \(S\). ROCKIT creates one thread per available core and, when idle, makes each of the threads (i) pop a first-order formula from the stack \(S\), (ii) compute the formula’s violated groundings, and (iii) perform CPA on these groundings. The aggregated groundings are compiled into ILP constraints and added to the ILP formulation. When the stack \(S\) is empty and all threads idle we solve the current ILP in parallel, obtain a solution, and begin the next CPI iteration.

There are different possible strategies for finding the ground clauses \(c\) (see Definition [2](#)) that minimize the number of counting constraints per first-order formula. While this problem can be solved optimally with algorithms that detect symmetries in propositional formulas such as SAUCY (Darga, Sakallah, and Markov 2008) or, alternatively, by reducing it to a frequent itemset mining problem for which several algorithms exist, these algorithms need to be called in each CPI iteration and for each first-order formula and extensive experiments showed that these algorithms dominated ILP solving. Therefore, we implemented a greedy algorithm that only estimates the optimal aggregation scheme. The algorithm stores, for each first-order clause, the violated groundings of the form \(c_1 \lor \ldots \lor c_n\) in a table with \(n\) columns where each column represents one literal position of the clause. For each column \(k\), ROCKIT computes the set of distinct rows \(R_k\) of the table that results from the projection onto the columns \(\{1, \ldots, n\} \setminus \{k\}\). Let \(d = \arg\min_k |R_k|\). The clause groundings are then aggregated with respect to the rows in \(R_d\).

ROCKIT employs MySQL’s in-memory tables for the computation of violated constraints and the aggregation. Most tables are hash indexed to facilitate highly efficient join processing. We use GUROBI\(^2\) as ROCKIT’s internal ILP solver due to its ability to parallelize its branch, bound, and cut algorithm, its remarkable performance on standard ILP benchmarks (Koch et al. 2011), and its symmetry detection heuristics. In addition to the project code, ROCKIT is made available as a web-service where users can upload MLNs. Furthermore, programmers can integrate the MLN query engine in their own applications via a REST interface.

**Experiments**

With the following experiments we assess whether and to what extent ROCKIT (a) reduces the number of constraints in the ILP formulation, (b) reduces the overall runtime, and (c) outperforms state-of-the-art MLN systems. We compare ROCKIT to three MLN systems ALCHEMY (Domingos et al. 2012), MARKOV THEBEAST (Riedel 2008), and TUFFY (version 3) (Niu et al. 2011). To ensure a fair comparison, we made MARKOV THEBEAST also use the ILP solver GUROBI. In addition, we investigate whether and to what extent (d) ROCKIT’s performance increases with the number of available cores of a shared-memory architecture. All experiments were performed on a standard PC with 8 GB RAM and 2 cores with 2.4 GHz each unless otherwise stated.

We used several established benchmark MLNs for the empirical evaluation. The entity resolution (ER) MLN addresses the problem of finding records corresponding to the same real-world entity (Singla and Domingos 2006).

<table>
<thead>
<tr>
<th></th>
<th>ER</th>
<th>IE</th>
<th>LP</th>
<th>PR</th>
<th>RC</th>
</tr>
</thead>
<tbody>
<tr>
<td>predicates</td>
<td>10</td>
<td>18</td>
<td>22</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>formulas</td>
<td>1,276</td>
<td>1,024</td>
<td>24</td>
<td>2,461</td>
<td>17</td>
</tr>
<tr>
<td>evidence</td>
<td>12,892</td>
<td>258,079</td>
<td>1,031</td>
<td>12,999</td>
<td>99,161</td>
</tr>
<tr>
<td>clauses</td>
<td>390,720</td>
<td>340,737</td>
<td>354,587</td>
<td>40,234,321</td>
<td>202,215</td>
</tr>
<tr>
<td>Number of ILP constraints</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/o CPA</td>
<td>357,056</td>
<td>4,041</td>
<td>31,658</td>
<td>2,688,122</td>
<td>164,047</td>
</tr>
<tr>
<td>w/ CPA</td>
<td>10,782</td>
<td>932</td>
<td>6,617</td>
<td>1,573</td>
<td>10,064</td>
</tr>
</tbody>
</table>

Table 4: Characteristics of the ML benchmark datasets and the number of constraints of the respective ILP formulations.
The information extraction (IE) system (Poon and Domingos 2007) MLN was created for the extraction of database records from text or semi-structured sources. The link prediction MLN (LP) was built to predict the relations holding between faculty, staff, and students of several university departments (Richardson and Domingos 2006). The protein interaction (PR) MLN was designed to predict interactions between proteins, enzymes, and phenotypes. The relational classification (RC) MLN was created for the extraction of database records. The information extraction (IE) and link prediction (LP) MLNs were downloaded from the Tuffy website. The formula weights were learned from text or semi-structured sources. The link prediction MLN, the relative error) and with two cores. * gap not reached within 1 hour, ** out of memory, *** did not terminate within 1 hour.

Figure 2: Running time in seconds of ROCKIt, ALCHEMY, MARKOV TheBEAST, and TUFFY for different gaps (bounds on the relative error) and with two cores. * gap not reached within 1 hour, ** out of memory, *** did not terminate within 1 hour.

The combination of CPA with lifted inference approaches. CPA is a novel algorithm that aggregates constraints. Extensive experiments showed that ROCKIt without CPA is more efficient than ALCHEMY. In all cases, ROCKIt was able to compute an interpretation with highest weight in less time. We conjecture that ROCKIt without CPA is more efficient than MARKOV TheBEAST because of ROCKIt’s more compact ILP formulation and the parallelization of the CPI phase. For the ER, PR, and RC dataset TUFFY and ALCHEMY were not able to achieve the same approximation as ROCKIt. ALCHEMY did not finish grounding within one hour on the RC dataset and ran out of memory on the PR dataset. For the LP dataset, no system was able to achieve a gap of 0.001 or lower.

Figure 3 compares the runtime of ROCKIt with CPA for different number of cores. For each benchmark, the runtime decreases when the number of cores increases with a diminishing reduction in runtime. The LP benchmark has the highest relative decrease of about 53% when comparing the running times on 1 and 8 cores.

Conclusions

We presented ROCKIt, a system for parallel MAP inference in SRL models combining CPI and cutting plane aggregation (CPA). CPA is a novel algorithm that aggregates symmetric constraints. Extensive experiments showed that ROCKIt is more efficient than existing MLN systems. Future work will investigate more aggregation strategies and the combination of CPA with lifted inference approaches.


