Lifting WALKSAT-Based Local Search Algorithms for MAP Inference

Somdeb Sarkhel
Computer Science Department
The University of Texas at Dallas
Richardson, TX 75080, USA
Somdeb.Sarkhel@utdallas.edu

Vibhav Gogate
Computer Science Department
The University of Texas at Dallas
Richardson, TX 75080, USA
vgogate@hlt.utdallas.edu

Abstract

In this short position paper, we consider MaxWalkSAT, a local search algorithm for MAP inference in probabilistic graphical models, and lift it to the first-order level, yielding a powerful algorithm for MAP inference in Markov logic networks (MLNs). Lifted MaxWalkSAT is based on the observation that if the MLN is monadic, namely if each predicate is unary then MaxWalkSAT is completely liftable in the sense that no grounding is required at inference time. We propose to utilize this observation in a straight-forward manner: convert the MLN to an equivalent monadic MLN by grounding a subset of its logical variables and then apply lifted MaxWalkSAT on it. It turns out however that the problem of finding the smallest subset of logical variables which when grounded will yield a monadic MLN is NP-hard in general and therefore we propose an approximation algorithm for solving it.

Introduction

Statistical relational models (Getoor and Taskar 2007) such as Markov logic networks (Domingos and Lowd 2009) bring the power and compactness of first-order logic to probabilistic graphical models. They are routinely used to solve hard problems in a wide variety of real-world application domains including computer vision, natural language processing, robotics and the Web. Recently, there has been a growing interest in exploiting relational structure during inference in statistical relational models. Unlike propositional inference algorithms which operate on individual random variables, these algorithms, which are often called lifted or first-order inference algorithms (Poole 2003), perform inference over a group of random variables. As a result, they are more scalable, typically exponentially faster when relational structure is present, than propositional inference algorithms.

In this paper, we show how to lift MaxWalkSAT (Kautz, Selman, and Jiang 1997; Selman, Kautz, and Cohen 1996), a state-of-the-art local search algorithm for MAP inference in probabilistic graphical models (MaxWalkSAT is currently the algorithm used in the Alchemy system (Kok et al. 2006) for MAP inference in MLNs). Our approach is based on the observation that when the MLN is monadic, i.e., when each atom in the MLN is unary, MaxWalkSAT is *completely liftable*. In monadic MLNs, the set of full assignments having cardinality $O(2^{nd})$ where n is the number of logical variables and d is the maximum domain size (possible groundings) of the logical variables, can be partitioned into $O((d+1)^n)$ subsets such that each element in each subset has the same probability. Thus, instead of performing a local search over $O(2^{nd})$ assignments as the propositional WalkSAT does, we can perform local search over a much smaller $O((d+1)^n)$ space, yielding the lifted MaxWalkSAT algorithm.

We propose to take advantage of the aforementioned observation in a straight-forward manner: convert the given MLN to a monadic MLN (pre-processing step) by grounding a subset of its logical variables and then apply lifted MaxWalkSAT on the monadic MLN. Unfortunately, it turns out that finding the smallest subset of logical variables which when grounded will yield a monadic MLN is NP-hard in general. Therefore, we propose to solve the optimization problem approximately by encoding it as a weighted maximum satisfiability (weighted MAX-SAT) problem and solving it using a weighted MAX-SAT solver (e.g., MaxWalkSAT). We chose this encoding because it is polynomial in the size of the MLN (i.e., in number of logical variables, predicates and formulas in the MLN) and simplifies the implementation because only one solver (MaxWalkSAT) needs to be implemented.

Preliminaries

In this section, we describe notation and preliminaries on propositional logic, first-order logic, Markov logic networks, MAP inference and WalkSAT architecture. For more details, refer to (Domingos and Lowd 2009; Koller and Friedman 2009; Hoos and Stützle 2004).

The language of propositional logic consists of atomic sentences called propositions or atoms, and logical connectives such as $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), $\Rightarrow$ (implication) and $\Leftrightarrow$(equivalence). Each proposition takes values from the binary domain {`False, True`} (or {0, 1}). A propositional formula f is an atom, or any complex formula that can be constructed from atoms using logical connectives. For example, A, B and C are propositional atoms and $f = A \vee \neg B \wedge C$ is a propositional formula. A *knowledge base* (KB) is a set of formulas. A *world* is a truth assignment to all atoms in the KB.

First-order logic (FOL) generalizes propositional logic

by allowing atoms to have internal structure; an atom in FOL is a predicate that represents relations between objects. A predicate consists of a predicate symbol, denoted by Monospace fonts, e.g., `Friends`, `Smokes`, etc., followed by a parenthesized list of arguments called *terms*. A term is a logical variable, denoted by lower case letters such as x, y, z, etc., or a constant, denoted by upper case letters such as X, Y, Z, etc. We assume that each logical variable, e.g., x, is typed and takes values over a finite set (called *domain*) Δx. The language of FOL also includes two quantifiers: $\forall$ (universal) and $\exists$ (existential) which express properties of an entire collection of objects. A formula in first order logic is a predicate (atom), or any complex sentence that can be constructed from atoms using logical connectives and quantifiers. For example, the formula $\forall x$ `Smokes`$(x) \Rightarrow$ `Asthma`(x) states that all persons who smoke have asthma. Where as $\exists x$ `Cancer`(x) states that there exists a person x who has cancer. A *first-order KB* is a set of first-order formulas.

In this paper, we use a subset of FOL which has no function symbols, equality constraints or existential quantifiers. We also assume that domains are finite (and therefore function-free) and that there is a one-to-one mapping between constants and objects in the domain (Herbrand interpretations). We assume that each formula f is of the form $\forall$x f, where x are the set of variables in f and f is a conjunction or disjunction of literals; each literal being an atom or its negation. For brevity, we will drop $\forall$ from all the formulas. Given variables $\mathbf{x} = \{x_1, ..., x_n\}$ and constants $\mathbf{X} = \{X_1, ..., X_n\}$ where $X_i \in \Delta x_i$, $f[\mathbf{X}/\mathbf{x}]$ is obtained by substituting every occurrence of variable x_i in f with X_i. A *ground formula* is a formula obtained by substituting all of its variables with a constant. A *ground KB* is a KB containing all possible groundings of all of its formulas. For example, the grounding of a KB containing one formula, `Smokes`$(x) \Rightarrow$ `Asthma`(x) where $\Delta x = \{Ana, Bob\}$, is a KB containing two formulas: `Smokes`$(Ana) \Rightarrow$ `Asthma`(Ana) and `Smokes`$(Bob) \Rightarrow$ `Asthma`(Bob).

Markov logic (Domingos and Lowd 2009) extends FOL by softening the hard constraints expressed by the formulas and is arguably the most popular modeling language for SRL. A soft formula or a weighted formula is a pair (f, w) where f is a formula in FOL and w is a real-number. A Markov logic network (MLN), denoted by $\mathcal{M}$, is a set of weighted formulas (f_i, w_i). Given a set of constants that represent objects in the domain, a Markov logic network defines a Markov network or a log-linear model. The Markov network is obtained by grounding the weighted first-order knowledge base and represents the following probability distribution.

$$P_{\mathcal{M}}(\omega) = \frac{1}{Z(\mathcal{M})} \exp\left(\sum_i w_i N(f_i, \omega)\right) \quad (1)$$

where ω is a world, $N(f_i, \omega)$ is the number of groundings of f_i that evaluate to `True` in the world ω and $Z(\mathcal{M})$ is a normalization constant or the partition function.

In this paper, we assume that the input MLN to our algorithm is in normal form (Jha et al. 2010). We require this for simplicity of exposition. A *normal* MLN is an MLN that satisfies the following two properties: (1) There are no constants in any formula, and (2) If two distinct atoms with the same predicate symbol have variables x and y in the same position then $\Delta x = \Delta y$. Note that in a normal MLN, we assume that the terms in each atom are ordered and therefore we can identify each term by its position in the order. Furthermore, we assume that the MLN is expressed as a set of weighted clauses.

Algorithm 1 WalkSAT(F, p, $maxFlips$)

$\omega =$ randomly chosen assignment to all variables in F
for $flip = 1$ **to** $maxFlips$
  if ω satisfies F then **return** ω
  $c =$ randomly selected unsatisfied clause in F
  with probability p flip a randomly selected variable in c
  else flip a variable that satisfies maximum number of clauses
return 'no solution found'

A common optimization inference task over MLNs is finding the most probable state of the world ω, that is finding a complete assignment to all ground atoms which maximizes the probability. This task is known as *Maximum a Posteriori* (MAP) inference in the Markov network literature, and *Most Probable Explanation* (MPE) inference in the Bayesian network literature. For Markov logic, this is formally defined as follows:

$$\begin{aligned} \arg\max_{\omega} P_{\mathcal{M}}(\omega) &= \arg\max_{\omega} \frac{1}{Z(\mathcal{M})} \exp\left(\sum_i w_i N(f_i, \omega)\right) \\ &= \arg\max_{\omega} \sum_i w_i N(f_i, \omega) \end{aligned} \quad (2)$$

From Eq. (2), we can see that the MAP problem in Markov logic reduces to finding the truth assignment that maximizes the sum of weights of satisfied clauses. Therefore, any weighted satisfiability solver can used to solve this problem. The problem is NP-hard in general, but effective solvers exist, both exact and approximate. The most commonly used approximate solver is MaxWalkSAT, a weighted variant of the WalkSAT local-search satisfiability solver (Selman, Kautz, and Cohen 1996) and is the algorithm used in the Alchemy system (Kok et al. 2006) for MAP inference.

The pseudo-code for WalkSAT is given in Algorithm 1. It takes as input a CNF formula F, a real number p and an integer $maxFlips$ which denotes the maximum number of iterations for which the algorithm will be run. The algorithm begins by randomly generating an assignment to all propositional variables in F. Then, at each iteration it randomly selects an unsatisfied clause and flips the value assigned to one of its literals as follows. With probability p, it flips the value assigned to a randomly selected literal and with probability $1 - p$, it flips a literal that maximizes the number of satisfied clauses (greedy hill-climbing step). WalkSAT is an incomplete algorithm, and as with other local search schemes may not yield the optimal solution. However, it performs remarkably well in practice, often outperforming other competing schemes by an order of magnitude. The optimization version of WalkSAT, i.e., MaxWalkSAT is similar to WalkSAT

except that at each step it tries to find an assignment that maximizes the total weight of the satisfied clauses.

Monadic WalkSAT

In this section, we will extend the WalkSAT solver to monadic first-order logic, a subset of first-order logic that contains only unary predicate symbols and has no function symbols. For example, the formula $\forall x\ \texttt{Smokes}(x) \Rightarrow \texttt{Asthma}(x)$ is in monadic first-order logic. For simplicity of exposition, we further restrict ourselves to a subset of monadic logic with no self-joins, i.e., a predicate appears only once in each formula.

To lift WalkSAT, we will lift the flip operation. The flip operation for propositional WalkSAT alters the truth value of a propositional variable. Analogously, the lifted flip operation alters truth assignments to all groundings of a predicate. To lift the flip operation, we make use of the fact that in monadic logic, several assignments will have the same probability. Therefore, if we group them together, the local search algorithm can directly jump from one group to the next group without making any "within group" moves. Formally, using the generalized binomial rule (Jha et al. 2010), we can prove that:

Proposition 1 *If two worlds ω_1 and ω_2 of a monadic MLN $\mathcal{M}$ are such that: (a) they differ only on truth assignments to the groundings of predicate $\texttt{S}$; and (b) the number of true groundings of $\texttt{S}$ is the same in both ω_1 and ω_2, then* $\Pr_{\mathcal{M}}(\omega_1) = \Pr_{\mathcal{M}}(\omega_2)$.

Proposition 1 enables us to group assignments to all groundings of a first-order atom by the number of true (or false) groundings. Therefore, we can compactly represent these equi-probable assignments of a predicate $\texttt{S}$ using the tuple $(\texttt{S}, k)$ where k is the number of true groundings of $\texttt{S}$. We will refer to $(\texttt{S}, k)$ as a **lifted assignment**.

Algorithm 2 presents the lifted WalkSAT algorithm. The algorithm begins with a random lifted assignment to all predicates $\texttt{S}$ in F. Then at each iteration $flip$, it heuristically selects an unsatisfied clause c of F and changes the lifted assignment to one of the atoms of c either randomly or greedily.

On monadic MLNs, lifted WalkSAT is clearly more efficient than propositional WalkSAT. The search space over which WalkSAT operates is bounded by $O(2^{nd})$ where n is the number of atoms in the MLN and d is the maximum domain size (possible groundings) of a logical variable. On the other hand, the search space of Lifted WalkSAT is bounded by $O((d+1)^n)$. The space complexity of lifted WalkSAT is $O(n)$ while that of WalkSAT is $O(nd)$. When d is large (e.g., in a social networking application, d can be as large as a few billion), this represents a significant improvement.

Algorithm 2 Monadic WalkSAT$(F, p, maxFlips)$

```
(S_i, k_i)_{i=1}^n = a random lifted assignment to all predicates in F
for flip = 1 to maxFlips
  if (S_i, k_i)_{i=1}^n satisfies F then return (S_i, k_i)_{i=1}^n
  c = heuristically selected unsatisfied clause of F
  with probability p select at random a predicate S in c
    and assign a random lifted assignment to S
  else find a tuple (S, k) such that S is in c
      and (S, k) satisfies the maximum number of clauses in F
      Assign k to S
return 'no solution found'
```

WalkSAT for arbitrary polyadic MLNs

To use Algorithm 2 in arbitrary (polyadic) MLNs, we propose the following approach: convert the polyadic MLN to an equivalent monadic MLN by grounding a subset of its logical variables. For example,

Example 1 *Consider the MLN $R(x,y) \vee S(y,z)$ with $\Delta x = \Delta y = \Delta z = \{A, B, C\}$. We can convert this MLN to an equivalent monadic MLN, by grounding y, i.e., by substituting the logical variable y with $\{A, B, C\}$. This yields the following MLN with three clauses: $R(x,A) \vee S(A,z)$, $R(x,B) \vee S(B,z)$ and $R(x,C) \vee S(C,z)$. Renaming these partially ground predicates yields the monadic MLN: $R_A(x) \vee S_A(z)$, $R_B(x) \vee S_B(z)$ and $R_C(x) \vee S_C(z)$.*

It turns out however that finding the smallest subset of logical variables which when grounded will yield a monadic MLN is a NP-hard problem. We are interested in such a subset because we want to ground as fewer variables as possible. Formally,

Theorem 1 *Given a MLN $\mathcal{M}$, finding the smallest subset of logical variables in $\mathcal{M}$ which when grounded yields an equivalent monadic MLN is NP-hard.*

Thus, unless $P = NP$, there does not exist a polynomial time algorithm that can solve this minimum subset problem. Therefore, we propose to solve it approximately by formulating (encoding) it as a weighted MAX-SAT problem and using a propositional MAX-SAT solver (e.g., MaxWalkSAT) to obtain an approximate solution.

Weighted MAX-SAT Encoding: We associate a Boolean variable $X_{i,j}$ with each logical variable in the j-th position of each predicate $\texttt{S}_i$ in the MLN. A true assignment to this variable indicates that the variable will be grounded and a false assignment indicates otherwise. We introduce two types of hard clauses: *selection clauses* and *link clauses*. Selection clauses model the constraint that among the n terms of the predicate at least $(n-1)$ should be grounded (note that at least $n-1$ terms should be grounded to convert a n-ary predicate to a unary predicate). This can be done by adding clauses of the form $X_{i,j} \vee X_{i,k}$ for all $j \neq k$. Link clauses model the constraint that if a term at position j in a predicate $\texttt{S}_i$ shares a logical variable with a term at position l in the predicate $\texttt{S}_j$ in a formula F in the MLN, then either both terms should be grounded or not at all. In other words, $X_{i,j} \Leftrightarrow X_{k,l}$. For each variable $X_{i,j}$, we also introduce a soft clause of the form $\neg X_{i,j}, v_{i,j}$ where $v_{i,j}$ equals the log of the product of the domain size of the term in the j-th position of predicate $\texttt{S}_i$ and the number of formulas in which $\texttt{S}_i$ is involved in. The soft clauses model that it is better to ground terms having smaller domain size as well as the ones which are involved in as fewer formulas.

Algorithm 3 describes the pseudo-code for converting the optimization problem of finding the minimum subset to a

Algorithm 3 Weighted-MAX-SAT-Encoding(F)

define variables $X_{i,j}$ for j-th term of i-th predicate
define G as set of constraints to be solved by weighted MAXSAT solver
Hard Clauses:
  foreach formula in F
    if predicates, $\mathtt{S}_i$, $\mathtt{S}_k$, share a term on position j, l
      add $X_{i,j} \Leftrightarrow X_{k,l}$ to G
  foreach i, j, k such that $j \neq k$
    add $X_{i,j} \vee X_{i,k}$ to G
Soft Clauses:
  forall variables $X_{i,j}$ **add** $(\neg X_{i,j}, v_{i,j})$ to G
return G

weighted MAX-SAT problem.

Example 2 *Next, we illustrate the weighted MAX-SAT encoding of the optimization problem on our running example:* $R(x, y) \vee S(y, z)$ *with* $\Delta x = \Delta y = \Delta z = \{A, B, C\}$. *We have two predicates, and each of them has two terms. Therefore, we have to define* 4 *variables, say* R_1, R_2, S_1, S_2. *The set of hard clauses are as follows: (i)* $R_2 \Leftrightarrow S_1$*; (ii)* $R_1 \vee R_2$*; and (iii)* $S_1 \vee S_2$. *The set of soft clauses are: (i)* $\neg R_1 3$*; (ii)* $\neg R_2 3$*; (iii)* $\neg S_1 3$*; and (iv)* $\neg S_2$.

Summary and Future work

In this paper we proposed lifted MaxWalkSAT, an algorithm for MAP inference in MLNs. Our approach is based on the observation that if the MLN is monadic, then MaxWalkSAT is completely liftable. We used this observation in a straight-forward manner: convert the MLN to a monadic MLN and then apply lifted MaxWalkSAT over it. We proposed a method for converting the given (polyadic) MLN to a monadic MLN. This method first encodes the problem of finding a minimum subset of logical variables which when grounded will yield a monadic MLN to a weighted MAX-SAT problem and then uses a MAX-SAT solver (e.g., MaxWalkSAT) to solve the problem.

Directions for future work include: combining our approach with other lifted inference rules such as the power rule (Jha et al. 2010; Gogate and Domingos 2011); performing a detailed experimental evaluation of our proposed algorithm; and applying lifted MaxWalkSAT to real world applications.

Acknowledgements This research was partly funded by the ARO MURI grant W911NF-08-1-0242. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO or the U.S. Government.

References

Domingos, P., and Lowd, D. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. San Rafael, CA: Morgan & Claypool.

Getoor, L., and Taskar, B., eds. 2007. *Introduction to Statistical Relational Learning*. MIT Press.

Gogate, V., and Domingos, P. 2011. Probabilistic Theorem Proving. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 256–265. AUAI Press.

Hoos, H. H., and Stützle, T. 2004. *Stochastic local search: Foundations & applications*. Morgan Kaufmann.

Jha, A.; Gogate, V.; Meliou, A.; and Suciu, D. 2010. Lifted Inference from the Other Side: The tractable Features. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS)*, 973–981.

Kautz, H.; Selman, B.; and Jiang, Y. 1997. A General Stochastic Approach to Solving Problems with Hard and Soft Constraints. In Gu, D.; Du, J.; and Pardalos, P., eds., *The Satisfiability Problem: Theory and Applications*. New York, NY: American Mathematical Society. 573–586.

Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Poon, H.; and Domingos, P. 2006. The Alchemy System for Statistical Relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://alchemy.cs.washington.edu.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Poole, D. 2003. First-Order Probabilistic Inference. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 985–991. Acapulco, Mexico: Morgan Kaufmann.

Selman, B.; Kautz, H.; and Cohen, B. 1996. Local Search Strategies for Satisfiability Testing. In Johnson, D. S., and Trick, M. A., eds., *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. Washington, DC: American Mathematical Society. 521–532.