Sound and Efficient Inference with Probabilistic and Deterministic Dependencies

Hoifung Poon **Pedro Domingos**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
{*hoifung, pedrod*}@*cs.washington.edu*

Abstract

Reasoning with both probabilistic and deterministic dependencies is important for many real-world problems, and in particular for the emerging field of statistical relational learning. However, probabilistic inference methods like MCMC or belief propagation tend to give poor results when deterministic or near-deterministic dependencies are present, and logical ones like satisfiability testing are inapplicable to probabilistic ones. In this paper we propose MC-SAT, an inference algorithm that combines ideas from MCMC and satisfiability. MC-SAT is based on Markov logic, which defines Markov networks using weighted clauses in first-order logic. From the point of view of MCMC, MC-SAT is a slice sampler with an auxiliary variable per clause, and with a satisfiability-based method for sampling the original variables given the auxiliary ones. From the point of view of satisfiability, MC-SAT wraps a procedure around the SampleSAT uniform sampler that enables it to sample from highly non-uniform distributions over satisfying assignments. Experiments on entity resolution and collective classification problems show that MC-SAT greatly outperforms Gibbs sampling and simulated tempering over a broad range of problem sizes and degrees of determinism.

Introduction

Many real-world problems require the use of both probabilistic and deterministic information. For example, entity resolution (the problem of determining which observations correspond to the same object) involves both probabilistic inferences (e.g., observations with similar properties are more likely to be the same object) and deterministic ones (e.g., transitive closure: if $x = y$ and $y = z$, then $x = z$) (McCallum & Wellner 2005). Unfortunately, while the state of the art in pure probabilistic and pure deterministic inference is quite advanced, combining them has received much less attention to date. At the boundary of the two, near-deterministic dependencies are particularly intractable, and responsible for the #P-completeness of probabilistic inference (Roth 1996). Problems of this type appear frequently in the new field of statistical relational learning, which combines statistical learning and inductive logic programming (Dietterich *et al.* 2004).

Deterministic dependencies break the support of a probability distribution into disconnected regions, making it difficult to design ergodic Markov chains for MCMC inference (Gilks *et al.* 1996). Gibbs sampling is trapped in a single region, and never converges to the correct answers. Running multiple chains with random starting points does not solve this problem, because it does not guarantee that different regions will be sampled with frequency proportional to their probability, and there may be a very large number of regions. Blocking is only viable for localized deterministic dependencies, but very large-scale ones often occur (e.g., transitive closure). Simply finding a starting point within the support of the distribution is an NP-hard problem. Near-deterministic dependencies preserve ergodicity, but lead to unacceptably long convergence times, even for methods like simulated tempering (Marinari & Parisi 1992). In belief propagation (Yedidia *et al.* 2001), deterministic or near-deterministic dependencies can lead to incorrect answers or failure to converge. A number of authors have taken advantage of deterministic dependencies to speed up exact inference in graphical models (e.g., (Allen & Darwiche 2003; Dechter & Mateescu 2004; Bartels & Bilmes 2004)), but their algorithms are unlikely to scale to problems with a large number of densely connected variables, like those found in statistical relational learning, and do not help in the case of near-deterministic dependencies.

Satisfying sets of interlocking deterministic constraints is an NP-complete problem, but in practice it is often tackled efficiently by current satisfiability solvers. For example, WalkSAT (Selman *et al.* 1996) can solve hard problems with hundreds of thousands of variables in minutes. Recently, it has been extended to sampling solutions near-uniformly (Wei *et al.* 2004). In this paper, we take advantage of this capability by developing MC-SAT, an MCMC algorithm that is able to handle deterministic and near-deterministic dependencies by using Wei et al.'s SampleSAT as a subroutine to efficiently jump between isolated or near-isolated regions of non-zero probability, while preserving detailed balance. MC-SAT accepts problems defined in Markov logic, a very general language that has both Markov networks and finite first-order logic as special cases (Richardson & Domingos 2006).

We begin by briefly reviewing the necessary background in MCMC, satisfiability and Markov logic. We then describe

MC-SAT and its application to entity resolution and collective classification problems, illustrating its ability to improve on algorithms like Gibbs sampling and simulated tempering.

Probabilistic Inference

Graphical models compactly represent the joint distribution of a set of variables (or nodes) $X = (X_1, X_2, \ldots, X_n) \in \mathcal{X}$ as a product of non-negative *potential functions* (Pearl 1988): $P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$, where each potential ϕ_k is over a subset of the variables $x_{\{k\}}$, and Z is a normalization constant. Under appropriate restrictions, the model is a *Bayesian network* and $Z = 1$. A *Markov network* or *Markov random field* can have arbitrary potentials. As long as $P(X = x) > 0$ for all x, the distribution can be equivalently represented as a *log-linear model*: $P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(x)\right)$, where the $f_i(x)$ are arbitrary *feature functions*. In this paper, we will be concerned exclusively with Boolean variables and features. Roughly speaking, larger weights w_j correspond to stronger dependencies. Deterministic dependencies can be viewed as the limit $w_i \to \infty$ (and $P(X = x) \to 0$ for some x).

The fundamental inference problem in graphical models is computing conditional probabilities. Perhaps the most widely used method for this is Markov chain Monte Carlo (MCMC) (Gilks *et al.* 1996), and in particular Gibbs sampling, which proceeds by sampling each variable in turn given its *Markov blanket* (the variables it appears in some potential with). To generate samples from the correct distribution, it suffices that the Markov chain satisfy *ergodicity* and *detailed balance*. In essence, all states must be aperiodically reachable from each other, and for any two states x, y $P(x)Q(x \to y) = P(y)Q(y \to x)$, where Q is the chain's transition probability. When strong dependencies are present, changes to the state of a variable given its neighbors become very unlikely, and convergence of the probability estimates to the true values becomes very slow. In the limit of deterministic dependencies, ergodicity breaks down. One way to speed up Gibbs sampling is by *simulated tempering* (Marinari & Parisi 1992), where chains with reduced weights are run in parallel with the original one, and we periodically attempt to swap the states of two chains. However, when weights are very large swaps become very unlikely, and infinite weights still break ergodicity. Another widely used approach relies on *auxiliary variables* to capture the dependencies. For example, we can define $P(X = x, U = u) = (1/Z) \prod_k I_{[0, \phi_k(x_{\{k\}})]}(u_k)$, where ϕ_k is the kth potential function, u_k is the kth auxiliary variable, $I_{[a,b]}(u_k) = 1$ if $a \leq u_k \leq b$, and $I_{[a,b]}(u_k) = 0$ otherwise. The marginal distribution of X under this joint is $P(X = x)$, so to sample from the original distribution it suffices to sample from $P(x, u)$ and ignore the u values. $P(u_k|x)$ is uniform in $[0, \phi_k(x_{\{k\}})]$, and thus easy to sample from. $P(x|u)$ is uniform in the "slice" of $\mathcal{X}$ that satisfies $\phi_k(x_{\{k\}}) \geq u_k$ for all k. Identifying this region is the main difficulty in this technique, known as *slice sampling* (Damien *et al.* 1999).

Satisfiability

A *knowledge base (KB)* in propositional logic is a set of formulas over Boolean variables. Every KB can be converted to *conjunctive normal form (CNF)*: a conjunction of clauses, each clause being a disjunction of literals, each literal being a variable or its negation. *Satisfiability* is the problem of finding an assignment of truth values to the variables that satisfies all the clauses (i.e., makes them true) or determining that none exists. It is the prototypical NP-complete problem. The last decade and a half has seen tremendous progress in the development of highly efficient satisfiability solvers. One of the most efficient approaches is stochastic local search, exemplified by the WalkSAT solver (Selman *et al.* 1996). Starting from a random initial state, WalkSAT repeatedly flips (changes the truth value of) a variable in a random unsatisfied clause. With probability q, WalkSAT chooses the variable that maximizes the number of satisfied clauses, and with probability $1-q$ it chooses a random variable. WalkSAT keeps going even if it finds a local maximum, and after n flips restarts from a new random state. The whole procedure is repeated m times. WalkSAT can solve random problems with hundreds of thousands of variables in a fraction of a second, and hard ones in minutes.

The MaxWalkSAT algorithm (Kautz *et al.* 1997) extends WalkSAT to the weighted satisfiability problem, where each clause has a weight and the goal is to maximize the sum of the weights of satisfied clauses. Park (2002) showed how the problem of finding the most likely state of a Bayesian network given some evidence can be efficiently solved by reduction to weighted satisfiability. The same encoding can be used to compute probabilities in Bayesian networks using our MC-SAT algorithm.

Most recently, Wei et al.(2004) extended WalkSAT to sample satisfying solutions near-uniformly by combining it with simulated annealing. At near-zero temperature, simulated annealing samples solutions uniformly, but will generally take too long to find them. WalkSAT finds solutions very fast, but samples them highly non-uniformly. Wei et al.'s SampleSAT algorithm samples solutions near-uniformly and highly efficiently by, at each iteration, performing a WalkSAT step with probability p and a simulated annealing step with probability $1 - p$. The parameter p is used to trade off uniformity and computational cost.

Markov Logic

Many domains contain complex relational structure, and first-order logic allows us to compactly represent it. For example, in a domain with n objects, representing the transitivity of a binary relation requires n^3 formulas in propositional logic, but only one in first-order logic: $\forall x \forall y \forall z \; R(x, y) \wedge R(y, z) \Rightarrow R(x, z)$. Just as Markov networks can be viewed as a probabilistic extension of propositional logic, Markov logic is a probabilistic extension of finite first-order logic (Richardson & Domingos 2006). A *Markov logic network (MLN)* is a set of weighted first-order clauses. Together with a set of constants, it defines a Markov network with one node per ground atom and one feature per ground clause. The weight of a feature is the weight of the first-order clause that originated it. The probability of a state x in such a network is given by $P(x) = (1/Z)\exp(\sum_i w_i f_i(x))$, where Z is a normalization constant, w_i is the weight of the ith clause, $f_i = 1$ if the ith clause is true, and $f_i = 0$ otherwise.

Markov logic makes it possible to compactly specify probability distributions over complex relational domains. Deterministic dependencies are represented by formulas with infinite weight. In this paper we will focus on MLNs whose formulas are function-free clauses and assume domain closure, ensuring that the Markov networks generated are finite (Richardson & Domingos 2006). Given the constants in the domain, inference in Markov logic reduces to inference over the resulting Markov network. Richardson and Domingos (2006) used Gibbs sampling for this purpose, but it is too slow for near-deterministic dependencies, and unsound for deterministic ones. MC-SAT addresses these problems.

Inference is often a subroutine when learning statistical models of relational domains (Dietterich *et al.* 2004). These models often contain hundreds of thousands of variables or more, making efficient inference crucial to their learnability. Many representations used in this field can be compactly translated into Markov logic (Richardson & Domingos 2006), and thus MC-SAT is also applicable to them.

The MC-SAT Algorithm

MC-SAT applies slice sampling to Markov logic, using SampleSAT to sample a new state given the auxiliary variables. In the Markov network obtained by applying an MLN to a set of constants, each ground clause c_k corresponds to the potential function $\phi_k(x) = \exp(w_k f_k(x))$, which has value e^{w_k} if c_k is satisfied, and 1 otherwise. We introduce an auxiliary variable u_k for each c_k. Assume for the moment that all weights are non-negative. On the ith iteration of MC-SAT, if c_k is not satisfied by the current state $x^{(i)}$, u_k is drawn uniformly from $[0,1]$; therefore $u_k \leq 1$ and $u_k \leq e^{w_i}$, and there is no requirement that it be satisfied in the next state. If c_k is satisfied, u_k is drawn uniformly from $[0, e^{w_i}]$, and with probability $1 - e^{-w_i}$ it will be greater than 1, in which case the next state must satisfy c_k. Thus, sampling all the auxiliary variables determines a random subset M of the currently satisfied clauses that must also be satisfied in the next state. We then take as the next state a uniform sample from the set of states $SAT(M)$ that satisfy M. (Notice that $SAT(M)$ is never empty, because it always contains at least the current state.) The initial state is found by applying a satisfiability solver to the set of all hard clauses in the network (i.e., all clauses with infinite weight). If this set is unsatisfiable, the output of MC-SAT is undefined.

Algorithm 1 gives pseudo-code for MC-SAT. $\mathcal{U}_S$ is the uniform distribution over set S. At each step, all hard clauses are selected with probability 1, and thus all sampled states satisfy them. For brevity, the code ignores the case of negative weights. These are handled by noting that a clause with weight $w < 0$ is equivalent to its negation with weight $-w$, and a clause's negation is the conjunction of the negations of all of its literals. Thus, instead of checking whether the clause is satisfied, we check whether its negation is satisfied; if it is, with probability $1 - e^{w}$ we select all of its negated literals, and with probability e^{w} we select none.

Algorithm 1 MC-SAT(*clauses, weights, num_samples*)

```
$x^{(0)} \leftarrow$ Satisfy(hard clauses)
for $i \leftarrow 1$ to num_samples do
    $M \leftarrow \emptyset$
    for all $c_k \in$ clauses satisfied by $x^{(i-1)}$ do
        With probability $1 - e^{-w_k}$ add $c_k$ to $M$
    end for
    Sample $x^{(i)} \sim \mathcal{U}_{SAT(M)}$
end for
```

Theorem 1 *The Markov chain generated by MC-SAT satisfies ergodicity and detailed balance.*

Proof. All states in the support of the true distribution $P(x)$ must satisfy all the hard clauses. At any step, there is non-zero probability that M will contain only the hard clauses, and therefore that the new state will be an arbitrary one in the support of $P(x)$. Thus all states in the support of $P(x)$ are reachable from each other in an arbitrary number of steps, and the Markov chain is ergodic. Let $Q(x \overset{M}{\rightarrow} y)$ be the probability of an MC-SAT step transitioning from state x to state y via clause set M. We show that MC-SAT satisfies detailed balance by proving the stronger claim that $P(x)Q(x \overset{M}{\rightarrow} y) = P(y)Q(y \overset{M}{\rightarrow} x)$. Given the assignment x, the probability of M being the selected clause set is $\prod_{i:\, f_i(x)=1 \,\wedge\, c_i \notin M} e^{-w_i} \cdot \prod_{j:\, c_j \in M}(1 - e^{-w_j})$. If $c_i \in M$, c_i is satisfied by both x and y. Given a uniform sampler, x and y have the same probability of being generated from M: $Q_M(x) = Q_M(y)$. Therefore

$$\begin{aligned}
&P(x)Q(x \overset{M}{\rightarrow} y) \\
&\quad = \tfrac{1}{Z}\exp\left(\textstyle\sum_i w_i f_i(x)\right) \cdot \textstyle\prod_{i:\, f_i(x)=1 \,\wedge\, c_i \notin M} e^{-w_i} \\
&\qquad \cdot \textstyle\prod_{c_i \in M} (1 - e^{-w_i}) \cdot Q_M(y) \\
&\quad = \tfrac{1}{Z} \textstyle\prod_{f_i(x)=1} e^{w_i} \cdot \prod_{i:\, f_i(x)=1 \,\wedge\, c_i \notin M} e^{-w_i} \\
&\qquad \cdot \textstyle\prod_{c_i \in M}(1 - e^{-w_i}) \cdot Q_M(y) \\
&\quad = \tfrac{1}{Z} \textstyle\prod_{c_i \in M} e^{w_i} \cdot \prod_{c_i \in M}(1 - e^{-w_i}) \cdot Q_M(y) \\
&\quad = \tfrac{1}{Z} \textstyle\prod_{c_i \in M} e^{w_i} \cdot \prod_{c_i \in M}(1 - e^{-w_i}) \cdot Q_M(x) \\
&\quad = P(y)Q(y \overset{M}{\rightarrow} x). \qquad \Box
\end{aligned}$$

Theorem 1 illustrates a fundamental difference between MC-SAT and other MCMC methods like Gibbs sampling and simulated tempering: MC-SAT is guaranteed to be sound, even in the presence of deterministic dependencies, while these other methods are not. In practice, perfectly uniform samples are too costly to obtain, and MC-SAT uses SampleSAT to obtain nearly uniform ones. SampleSAT's p parameter allows us to easily trade off speed and uniformity. We further speed it up by performing unit propagation during and after clause selection, and by using WalkSAT's tabu heuristic. SampleSAT is also imperfect in that it may fail to find a satisfying solution, when in fact there is always one (the current state). When this happens, we assume the current state is indeed the only solution, and use it as the next sample.

MC-SAT may at first appear to be impractical, because it requires a complete run of a SAT solver at each step. However, it is well known that most SAT runs are extremely short, and it is unlikely that the clause set generated at any particular step will be in the critical "hard" region. Indeed, in our experiments we found that SampleSAT ran on average in a fraction of a second, and this time was in fact dominated by the time required to generate M. It is also possible to mix SampleSAT steps with ordinary Gibbs steps, on the basis that SampleSAT is needed for jumping between modes, but not for exploring a mode.

An alternate way of arriving at MC-SAT is to start from SampleSAT and ask: how can we use it for probabilistic inference? How can we turn a uniform sampler into a sampler for highly non-uniform distributions? Slice sampling accomplishes exactly this, and hence its use in MC-SAT.

Experiments

Domains

Entity resolution is the problem of determining which observations correspond to the same entity. For example, when merging databases we need to determine which records are duplicates. This problem is of crucial importance to many large scientific projects, businesses, and government agencies, and has received increasing attention in the AI community in recent years. We carried out experiments using the BibServ.org database of bibliographic records, which combines CiteSeer, DBLP, and user-donated databases. The goal is to compute, for every pair of citations (x, y), the marginal probability that $x = y$, and similarly for pairs of fields (authors, titles and venues). We used a Markov logic network similar to that of Singla and Domingos (2005), containing formulas like: if two fields have high TF-IDF similarity, they are likely to be the same; if two fields are the same, their records are likely to be the same; if two records are the same, their fields are the same; etc. This last rule is deterministic, while the previous ones are not. Crucially, we added the transitivity rule: $\forall x, y, z\; x = y \wedge y = z \Rightarrow x = z$. This rule is used in an *ad hoc* way in most entity resolution systems, and greatly complicates inference. We also added the near-deterministic rule "If the titles and venues are both the same, the papers are the same," which captures the non-linear effect of these two pieces of evidence. We used the CiteSeer and user-donated subset of BibServ (88,035 citations), applied the canopy method (McCallum *et al.* 2000) to extract subclusters of plausible matches, and used the largest canopies to form data sets of varying size. For evaluation, we hand-labeled the data.

Collective classification is the problem of simultaneously classifying a set of related objects, and has many important instances: Web page classification, image segmentation, social network analysis, word-of-mouth marketing, spin glasses, hidden Markov models, etc. We manually created an MLN that captures the essential aspects of many different collective classification problems. Its two key rules are: $\forall x, y, u\; C(x, u) \wedge R(x, y) \Rightarrow C(y, u)$ (if an object x is of class u and is linked to object y, y is likely to also be of class u) and $\forall x, y, u\; C(x, u) \wedge C(y, u) \Rightarrow R(x, y)$ (objects of the same class are likely to be linked). Depending on the domain, the degree of hardness of these rules can vary widely. We also included rules of the form $\forall x, v, u A(x, v) \wedge E(v, u) \Rightarrow C(x, u)$ to represent the more traditional predictive relation between an object's attributes and its class. ($E(v, u)$ means that value v is evidence of class u. In our experiments, some values were indicative of a single class, and some were indicative of more than one class.) Finally, we added unit clauses to capture the default frequencies of classes and relations. We randomly generated data sets with varying numbers of objects and categories. The goal is to compute the marginal probabilities of the groundings of $C(x, u)$ given the groundings of $R(x, y)$, $A(x, v)$ and $E(v, u)$ as evidence.

Systems

We implemented MC-SAT and simulated tempering as extensions of the Alchemy system (Kok *et al.* 2005), and used Alchemy's implementation of Gibbs sampling, with ten chains. We found the performance of MC-SAT to be fairly insensitive to the SampleSAT settings. We report the results for $p = 0.5$, a temperature of 0.5, and continuing the run for 10 steps after reaching a solution. For simulated tempering, we averaged the probabilities of m runs of n swapping chains each, and tried various combinations of m and n. The best results were obtained with three runs of ten swapping chains each, and this is what we report. We used evenly spaced weights for the chains (e.g., for ten chains: $w, 0.9w, 0.8w, \ldots, 0.1w$, where w is the true weight). However, in experiments with very large weights even $0.1w$ was too large to allow any mode jumping, and we also tried $w, w/k, w/k^2, \ldots, w/(k^{n-1})$, with k equal to the $(n-1)$th root of the largest weight. This ensured that the highest-temperature chain had all weights of 1.0 or less. However, it did not noticeably improve the results, and with either scheme there was very little chain swapping for the largest weights, making simulated tempering not much better than Gibbs (since the samples from all but the true chain are discarded). All chains of all algorithms were initialized with random solutions to all the hard clauses. (By default, Alchemy initializes MCMC with a mode found using MaxWalkSAT, but some of our data sets were too large for MaxWalkSAT.) We used add-one smoothing on all probabilities.

Methodology

We assigned a weight of 1000 to deterministic clauses. (A state that violates a clause with this weight becomes 2×10^{434} times less probable, and effectively never occurs in runs of MCMC.) To observe the behavior of the algorithms with varying degreees of determinism, we also varied the weights of the hard clauses logarithmically from 1 to 32. To evaluate scalability, we varied the number of objects from 50 to 150. In the BibServ domain, 100 objects yielded 14,450 query atoms and 1.25 million ground clauses; in the collective classification domain, 2,000 and 105,140, respectively.

For BibServ, we trained the MLN using our hand-labeled data and the algorithm of Singla and Domingos (2005), as implemented in Alchemy. The algorithm cannot learn hard clauses, so we learned only the weights of the soft ones. Since the data matches the hard clauses perfectly, adding them does not affect the fit. For inference, we modified Alchemy to minimize the time and memory required to create the relevant ground network given the evidence. This network was then passed to all three algorithms.

Generating data from an MLN is a difficult problem (indeed, the problem MC-SAT addresses). Thus, for the collective classification domain, we instead followed the approach of first generating the data using a heuristic procedure, and then learning the MLN weights from this data. While this does not guarantee that the data is a perfect sample from the network, it is unlikely to bias results in favor of one inference algorithm.

A natural way to evaluate the algorithms would be to let them run until convergence and compare the running times. However, this is not feasible because some of the algorithms may never converge, and diagnosing convergence is extremely difficult, particularly in near-deterministic domains. Instead, we gave all algorithms the same running time and compared their accuracy, which more closely parallels the way MCMC is used in practice. A standard measure of accuracy is the K-L divergence of the actual and computed distributions. Since computing it exactly is infeasible for realistic-sized domains, we used the negative log-likelihood of the data according to the inference algorithms as a proxy. This is appropriate because the negative log-likelihood is a sampling approximation of the K-L divergence to the data-generating distribution, shifted by its entropy, which is indepedent of the inference algorithm.

Results

The results are shown in Figures 1 and 2, and are averages of ten runs. For the time graph, we used 1000 as the hard clause weight, 100 objects, and no burn-in. We begin the plots at the end of the first complete iteration for each algorithm. MC-SAT converges very rapidly, and dominates the other two algorithms by a large margin. Each Gibbs chain never escapes the mode it started in. Simulated tempering improves very slowly over time as swapping takes effect (note the log scale). For the weight graph, we used 100 objects, 10 minutes of burn-in, and 100 minutes for the entire inference. In both domains, MC-SAT dominates for weights beyond 4, by a wide margin. Most remarkably, the performance of MC-SAT is nearly constant throughout the entire weight range. For the object number graph, we used a weight of 32 for the hard clauses, and allowed 10 minutes for burn-in and 100 minutes for the entire inference. In both domains, MC-SAT dominates throughout the entire range, by a wide margin. We also conducted experiments on the more difficult task of simultaneous collective classification and link prediction (i.e., predicting both $C(x, u)$ and $R(x, y)$ given $A(x, v)$ and $E(v, u)$), and MC-SAT outperformed the other algorithms by an even wider margin.

In summary, MC-SAT greatly outperforms Gibbs sampling and simulated tempering, particularly when deterministic dependencies are present. This is attributable to its ability to rapidly jump between modes while maintaining detailed balance.

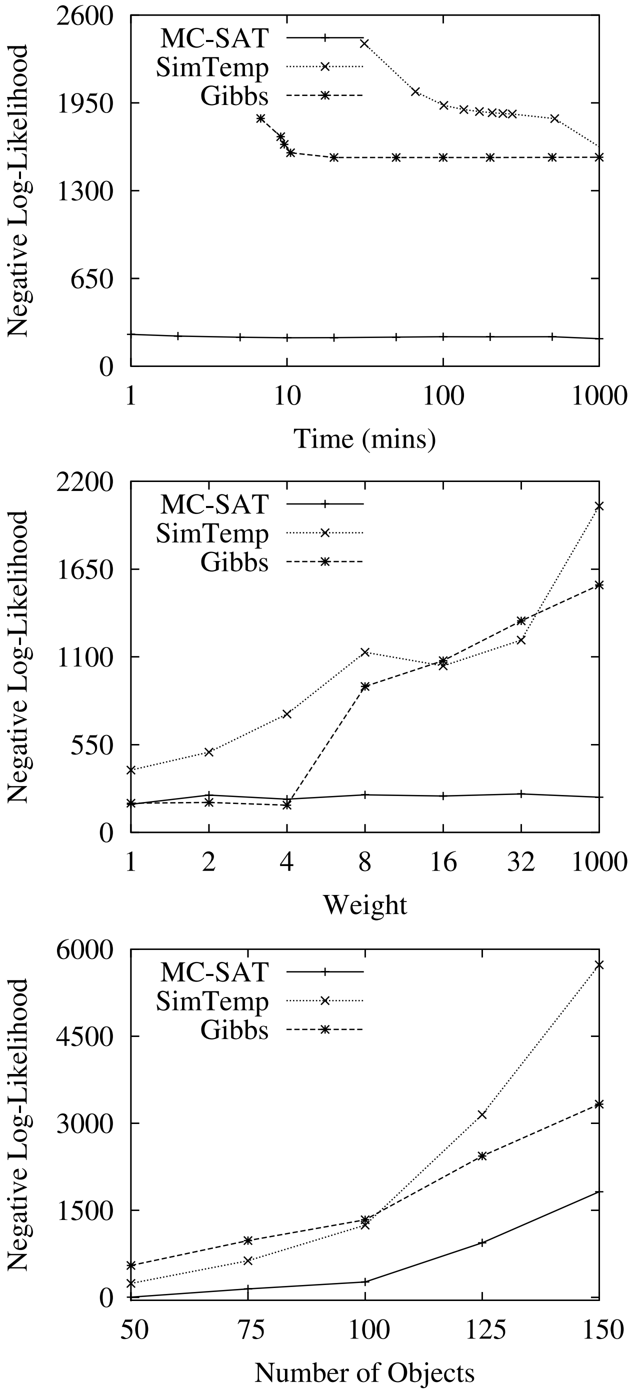
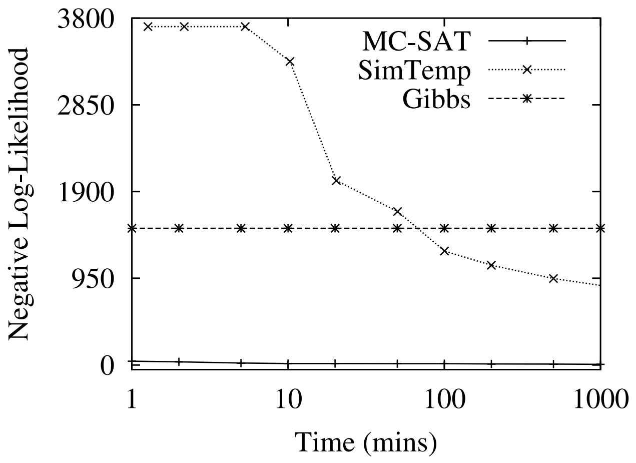
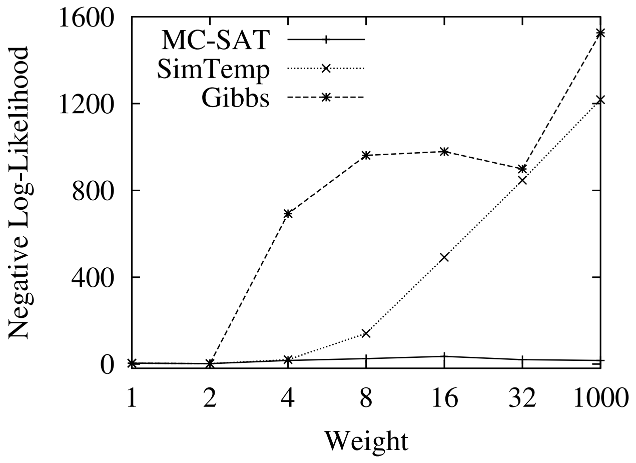


Figure 1: Experimental results for entity resolution: negative log-likelihood as a function of time (top graph), hard clause weight (middle), and number of objects (bottom).

Conclusion

Many real-world applications require reasoning with a combination of probabilistic and deterministic dependencies. The MC-SAT algorithm accomplishes this by combining slice sampling with satisfiability testing. Experiments on entity resolution and collective classification problems show





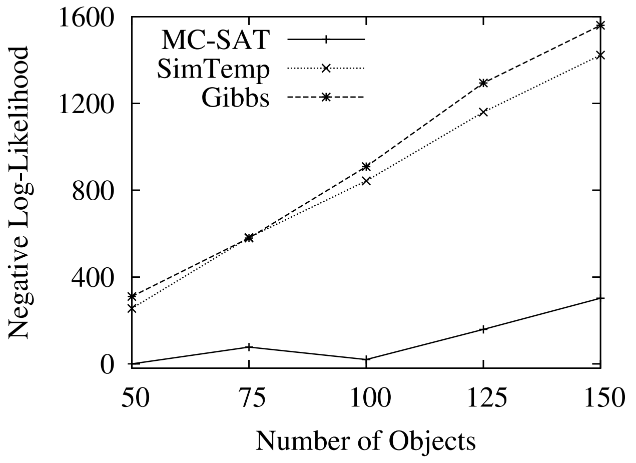


Figure 2: Results for collective classification: negative log-likelihood as a function of time (top graph), hard clause weight (middle), and number of objects (bottom).

that it greatly outperforms Gibbs sampling and simulated tempering. Directions for future work include using MC-SAT for learning, reducing its memory requirements in relational domains by exploiting sparseness, and applying it to other domains.

Acknowledgements

We are grateful to Bart Selman for helpful discussions, and to Parag Singla for help with the BibServ dataset. This research was partly supported by DARPA grant FA8750-05-2-0283 (managed by AFRL), DARPA contract NBCH-D030010, NSF grant IIS-0534881, and ONR grant N00014-05-1-0313. 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 DARPA, NSF, ONR, or the United States Government.

References

Allen, D. & Darwiche, A. 2003. New advances in inference by recursive conditioning. In *UAI-03*.

Bartels, C. & Bilmes, J. 2004. Elimination is not enough: Non-minimal triangulations for graphical models. Technical report UWEETR-2004-00010, Univ. of Washington.

Damien, P.; Wakefield, J.; Walker, S. 1999. Gibbs sampling for Bayesian non-conjugate and hierarchical models by auxiliary variables. *Journal of the Royal Statistical Society* B, 61:2.

Dechter, R. & Mateescu, R. 2004. Mixtures of deterministic-probabilistic nets and their search space. In *UAI-04*.

Dietterich, T.; Getoor, L.; Murphy, K., eds. 2004. *Proc. ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*. IMLS.

Gilks, W. R.; Richardson, S.; Spiegelhalter, D. J., eds. 1996. *Markov Chain Monte Carlo in Practice*. Chapman and Hall.

Kautz, H.; Selman, B.; Jiang, Y. 1997. A general stochastic approach to solving problems with hard and soft constraints. In *The Satisfiability Problem: Theory and Applications*. AMS.

Kok, S.; Singla, P.; Richardson, M.; Domingos, P. 2005. The Alchemy system for statistical relational AI. http://www.cs.washington.edu/ai/alchemy/.

Marinari, E., and Parisi, G. 1992. Simulated tempering: A new Monte Carlo scheme. *Europhysics Letters, 19, 451-458*.

McCallum, A.; Nigam, K.; Ungar, L. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD-00*.

McCallum, A. & Wellner, B. 2005. Conditional models of identity uncertainty with application to noun coreference. In *NIPS-04*.

Park, J. 2002. Using weighted MAX-SAT engines to solve MPE. In *AAAI-02*.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Richardson, M. & Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Roth, D. 1996. On the hardness of approximate reasoning. *Artificial Intelligence* 82:273–302.

Selman, B.; Kautz, H.; Cohen, B. 1996. Local search strategies for satisfiability testing. In *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. AMS.

Singla, P. & Domingos, P. 2005. Discriminative training of Markov logic networks. In *AAAI-05*.

Yedidia, J. S.; Freeman, W. T.; Weiss, Y. 2001. Generalized belief propagation. In *NIPS-01*.

Wei, W.; Erenrich, J.; Selman, B. 2004. Towards efficient sampling: Exploiting random walk strategies. In *AAAI-04*.