Lifted Message Passing for Satisfiability

Fabian Hadiji and **Kristian Kersting** and **Babak Ahmadi**
Knowledge Discovery Department, Fraunhofer IAIS
53754 Sankt Augustin, Germany
{firstname.lastname}@iais.fraunhofer.de

Abstract

Unifying logical and probabilistic reasoning is a long-standing goal of AI. While recent work in lifted belief propagation, handling whole sets of indistinguishable objects together, are promising steps towards achieving this goal that even scale to realistic domains, they are not tailored towards solving combinatorial problems such as determining the satisfiability of Boolean formulas. Recent results, however, show that certain other message passing algorithms, namely, survey propagation, are remarkably successful at solving such problems. In this paper, we propose the first lifted variants of survey propagation and its simpler version warning propagation. Our initial experimental results indicate that they are faster than using lifted belief propagation to determine the satisfiability of Boolean formulas.

Introduction

Much has been achieved in the field of AI, yet much remains to be done if we are to reach the goals we all imagine. One of the key challenges in moving ahead is closing the gap between logical and statistical AI. Logical AI has mainly focused on complex representations, and statistical AI on uncertainty. Intelligent agents, however, must be able to handle both the complexity and the uncertainty of the real world.

Therefore, it is not surprising that recent years have witnessed a surge of interest in lifted probabilistic inference, handling whole sets of indistinguishable objects together. For instance, lifted versions of the variable elimination algorithm have been proposed, see e.g. (Milch et al. 2008) and references in there. These exact inference approaches are extremely complex, so far do not easily scale to realistic domains, and hence have only been applied to rather small artificial problems. Recently, Sen et al. (2008) presented a lifted variable elimination approach based on bisimulation. It essentially groups together random variables if they have identical computation trees, the tree-structured unrolling of the underlying graphical model rooted at the nodes. The simplest and hence most efficient lifted inference approaches are lifted belief propagation approaches. They easily scale to realistic domains. Singla and Domingos (2008) developed the first lifted belief propagation variant tailored towards Markov logic networks. Subsequently, Kersting et al. (2009) generalized Singla and Domingos' approach to any factor graph over finite random variables. Similar to Sen et al., they also group together random variables, if they have identical computation trees, but now run the more efficient belief propagation approximation.

The main drawback of all currently existing lifted inference approaches, however, is that they are not tailored towards combinatorial problems such as determining the satisfiability of Boolean formulas. In many real-world applications, however, the problem formulation does not neatly fall into the pure probabilistic inference case. The problem may very well have a component that can be well-modeled as a combinatorial problem, hence, taking it outside the scope of standard (lifted) belief propagation. Ideally, lifted inference should be efficient here, too, exploiting as much symmetries as possible. Driven by a similar question, namely how to reason with both probabilistic and deterministic dependencies, but for the non-lifted case, Poon and Domingos (2006) developed MC-SAT that combines ideas from MCMC and satisfiability. It still proceeds by first fully instantiating the first-order theory and then essentially staying at the propositional level. LazySAT (Singla and Domingos 2006) is a lazy version of WalkSAT taking advantage of relational sparsity. Recently, Poon et al.(2008) have shown that this idea of lazy inference goes beyond satisfiability and can be combined with other propositional inference algorithms such as belief propagation. So far, however, the link has not been explored on the lifted (message-passing) level.

In this paper, we therefore revisit lifted message-passing algorithms this time for combinatorial problems. Specifically, we focus on the Boolean satisfiability (SAT) problem consisting of a formula F representing a set of constraints over n Boolean variables, which must be set so as to satisfy all constraints. It is the most studied problem in Computer Science and AI. All other NP-complete problems can be reduced to it and, hence, "reduction to SAT" is a powerful paradigm for solving Computer Science and AI problems and has applications in several important areas such as automated deduction, verification, and planning, among others.

In fact, we develop a lifted version of an exciting new algorithm for solving combinatorial problems, namely survey propagation (SP). It was introduced by Mezard et al. (2002) and can easily solve random SAT problems with one million variables in a few minutes on a desktop computer. Braunstein and Zecchina (2004) have shown that SP can be viewed as a form of belief propagation, hence, somehow suggesting that lifting SP is possible. As noted by Kroc et al. (2007; 2009), SP's remarkable effectiveness — the performance is clearly beyond the reach of mainstream SAT solvers — has created the impression that solving hard combinatorial instances requires survey propagation and belief propagation would have little success. Kroc et al., however, have shown that "the gap between BP and SP narrows" as the number of variables per clause increases. This is also supported by (Montanari, Ricci-Tersenghi, and Semerjian 2007) showing good performance of a combination of belief propagation and warning propagation, a popular message passing algorithm that is simpler than SP. Therefore we also show how to lift warning propagation and, hence, provide a complete picture of lifted message-passing algorithms for SAT, which is our main contribution.

We proceed as follows. We first provide background on Boolean formulas and factor graphs, and discuss lifted message passing. Specifically, we review **Step 1** of LBP, which consists of **lifting the factor graph**, and show how a simplified version can be used for lifting warning and survey propagation. Then we introduce **Step 2** for both warning and survey propagation, i.e., the **modified message-passing formulas** to be run on the lifted graph. Again we argue that they are simpler than the ones for LBP. Before concluding, we present the results of our initial experimental evaluation.

CNFs and Factor Graphs

We assume that a Boolean formula is represented in Conjunctive Normal Form (CNF). A *CNF* is a conjunction of disjunctions of Boolean literals. A literal is either a negated or unnegated propositional variable. Specifically, a CNF consists of n variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ with $x_i \in \{0, 1\}$ and m clauses $\mathbf{C} = (C_1, \ldots, C_m)$ constraining the variables. A clause C_i is defined on a subset of the variables $\mathbf{X}_i = \{X_{i,1}, \ldots, X_{i,k}\}$ and — this will turn out to be convenient for formulating the message-passing update formulas — a corresponding list of their signs which is denoted as $\mathbf{S}_i = \{s_{i,1}, \ldots, s_{i,k}\}$ with $s_{i,j} = -1$ if $X_{i,j}$ occurs *unnegated*, otherwise $s_{i,j} = 1$ if $X_{i,j}$ occurs *negated*. A solution to a CNF is an assignment to all variables in $\mathbf{X}$ satisfying all constraints in $\mathbf{C}$. As an example, consider the following CNF, consisting of three variables, three clauses, and seven literals: $(X_1 \vee \neg X_2) \wedge (\neg X_1 \vee X_2) \wedge (X_1 \vee X_2 \vee X_3)$.

As e.g. shown in (Kschischang, Frey, and Loeliger 2001), every CNF can be represented as a factor graph. A factor graph is a bipartite graph $G = (V, E)$, where V is the set of nodes, which contains a variable node (denoted as a circle) for each variable X_i and a factor node (denoted as a square) for each clause C_i. There is an edge $e \in E$ between a variable node and a factor node if and only if the variable appears in the clause. Since G is bipartite, there are no edges connecting two variable nodes or two factors. The factor graph representing the example CNF from above is shown in Fig. 1 **(left)**. We use a dashed line between a variable and a clause whenever the variable appears negated in a clause, i.e. $s_{i,j} = 1$, otherwise a full line.

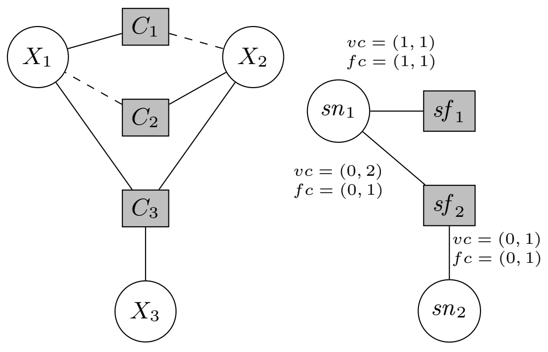

Figure 1: **(Left)** $(X_1 \vee \neg X_2) \wedge (\neg X_1 \vee X_2) \wedge (X_1 \vee X_2 \vee X_3)$ represented as a factor graph. Circles denote variable nodes and squares denote factors. A dashed line indicates that the variable appears negated in a clause; otherwise we use a full line. **(Right)** The resulting lifted CNF factor graph after running CP as illustrated in Fig. 2. For details, see main text.

Step 1: Lifting the Factor Graph

One of the best known message passing approaches is belief propagation (BP). Given a probability distribution $P(x_1, \ldots, x_n) = \frac{1}{Z} \prod_k f_k(\mathbf{x}_k)$ over some random variables, it computes the marginal probabilities of the variables. Here, each factor f_k is a non-negative function of a subset $\mathbf{X}_k$ of the random variables, and Z is a normalization constant. Here, we use it to illustrate the basic idea of lifted message-passing.

BP works by sending messages between variable nodes and their neighboring factor nodes. The message from a variable i to a factor a is

$$n_{i \to a}(x_i) = \prod\nolimits_{b \in \mathrm{nb}(i) \setminus \{a\}} m_{b \to i}(x_i)$$

where $\mathrm{nb}(i)$ is the set of factors i appears in. The message from a factor to a variable is

$$m_{a \to i}(x_i) = \sum\nolimits_{\neg\{i\}} \left(f_a(\mathbf{x}_a) \prod_{j \in \mathrm{nb}(a) \setminus \{i\}} n_{j \to a}(x_j) \right)$$

where $\mathrm{nb}(a)$ are the arguments of factor a, and the sum is over all of these except i, denoted as $\neg\{i\}$. The messages are usually initialized to 1.

Although BP is already quite efficient, many graphical models produce inference problems with a lot of symmetries not reflected in the graphical structure and hence not exploitable by BP. To overcome this, lifted versions of BP (LBP) have recently been proposed (Singla and Domingos 2008; Kersting, Ahmadi, and Natarajan 2009). LBP runs in two steps:

1. It groups together nodes and factors that send and receive the same messages. This is sometimes called lifting.
2. Then, it runs a modified BP on the resulting lifted, i.e., clustered network.

As shown empirically by Singla and Domingos (Singla and Domingos 2008) as well as by Kersting et al. (Kersting, Ahmadi, and Natarajan 2009), this can significantly speed up

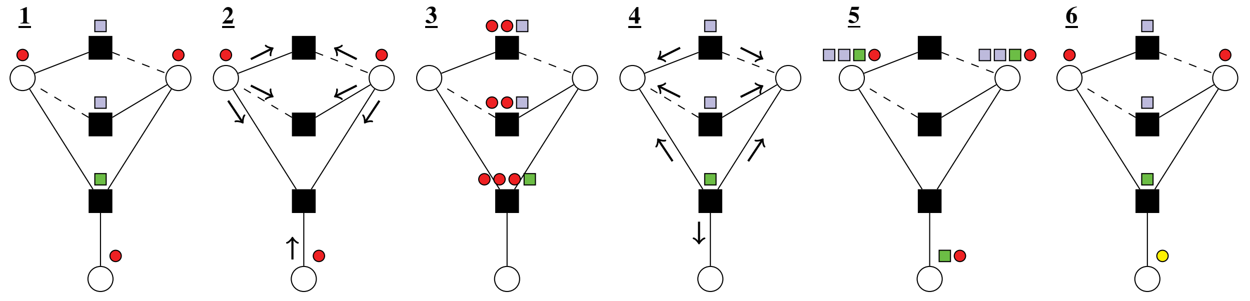

Figure 2: Lifting a factor graph running a color passing (CP) approach. From left to right, the steps of CP running on the factor graph in Fig. 1 **(left)** (assuming no evidence). The colored small circles and squares denote the groups and signatures produced running CFG. (Best viewed in color.)

inference. More importantly, Kersting et al. have shown that the lifting step can be viewed as a color-passing (CP) approach. This view abstracts from the type of messages sent and, hence, highlights one of the main insights underlying the present paper: *CP can be used to lift other message-passing approaches.*

More importantly, as we will show, it can actually be simplified in the SAT context. But first, let us illustrate CP. To do so, we use fraktur letters such as $\mathfrak{G}$, $\mathfrak{X}$, and $\mathfrak{f}$ to denote the lifted graphs, nodes, and factors. Due to space restrictions, we will not review the lifted BP message update formulas but instead refer to (Singla and Domingos 2008; Kersting, Ahmadi, and Natarajan 2009).

Essentially, CP simulates message passing, keeping track of which nodes and factors send the same messages, and groups nodes and factors together correspondingly. Specifically, let G be a given factor graph with variable and factor nodes. Initially, all variable nodes fall into the same group as we do not deal with evidence here. For ease of explanation, we will represent the groups by colored circles, say red. All factor nodes with the same associated potentials also fall into one group represented by colored squares. For the factor graph in Fig. 1 **(left)** the situation is depicted in Fig. 2. As shown on the left-hand side, assuming no evidence, all variable nodes are unknown, i.e., red. Now, each variable node sends a message to its neighboring factor nodes saying “I am of color red” (step 2). A factor node sorts the incoming colors into a vector according to the order the variables appear in its arguments. The last entry of the vector is the factor node's own color, represented as light blue, respectively green, squares in Fig. 2 (step 3). Based on these signatures, the colors of the factors are newly determined and sent back to the neighboring variables nodes (step 4), essentially saying “I have communicated with these nodes”. The variable nodes stack the incoming signatures together and, hence, form unique signatures of their one-step message history (step 5). Variable nodes with the same stacked signatures, i.e., message history can be grouped together. To indicate this, we assign a new color to each group (step 6). In our running example, only variable node X_3 changes its color from red to yellow. Finally, this process is iterated. The process stops when no new colors are created anymore.

The final lifted factor graph $\mathfrak{G}$ is constructed by grouping all nodes with the same color into so-called *supernodes* and all factors with the same color signatures into so-called *superfactors*. Supernodes (resp. superfactors) are sets of nodes (resp. factors) that send and receive the same messages at each step of carrying out message-passing on G. It is clear that they form a partition of the nodes in G. In our case, variable nodes X_1, X_2 and factor nodes C_1, C_2 are grouped together into supernode $sn_1 = \{X_1, X_2\}$ and superfactor $sf_1 = \{C_1, C_2\}$, the others are singletons (sn_2 and sf_2) as shown in Fig. 1 **(right)**.

Now, we are essentially ready to run Step 2, i.e., the modified warning respectively survey propagation on the lifted factor graph. Before doing so, we show that CP and its outcome, the lifted factor graph, are simpler for the CNF case than the ones for the BP case.

First, we can employ a more efficient color signature coding scheme. The initial grouping of the factors solely depends on their zero state. Additionally, the factor nodes do not have to sort the incoming colors according to the positions of the variables, instead only the sign of a variable matters, i.e. only two positions exist.

Second, we can employ simplified counts. As in LBP, the lifted update formulas essentially simulate ground warning resp. survey propagation on the lifted network. To do so, we store the counts of how often messages are sent along edges. Specifically, there are two types of counts. First, there is the *factor count* fc. It represents the number of equal ground factors that send messages to the variable at a given position. Because there are only two positions — a variable may occur at any position in *negated* and *unnegated* form, we store two values: counts for the negated position and for the unnegated position. Reconsider Fig. 1 **(right)**. Here, the count associated with the edge between sn_1 and sf_1 is $(1, 1)$ because sn_1 represents ground variables that appeared positive and negative in ground factors represented by sf_1. Second, there is the *variable count* vc. It corresponds to the number of ground variables that send messages to a factor at each position. In the ground network, the clause C_3 is connected to three positive variables, but two of them are represented by a single supernode in the lifted network. Hence, we have $vc = (0, 2)$ for the edge between sn_1 and sf_2. So, for lifted CNF factor graphs, all counts are of dimension 2 only.

Step 2: Lifted Warning Propagation

Clauses consisting of a single literal only, i.e. factors with an edge to exactly one variable X only, are called unit clauses. A unit clause essentially fixes the truth value of X. The process of fixing all variables appearing in unit clauses and simplifying the CNF correspondingly is called *Unit Propagation* (UP). Casted into the message passing framework (Braunstein, Mézard, and Zecchina 2005; Montanari, Ricci-Tersenghi, and Semerjian 2007), it is known under the name of *Warning Propagation (WP)*.

Intuitively, a message $m_{a\to i} = 1$ sent from a factor to a variable says: "Warning! The variable i must take on the value satisfying the clause represented by the factor a." As e.g. shown by Braunstein et al. (2005), this can be expressed mathematically as follows:

$$m_{a\to i} = \prod\nolimits_{j\in \mathrm{nb}(a)\setminus\{i\}} \theta(n_{j\to a} s_{a,j})$$

where $\mathrm{nb}(a)$ is the set of variables that a constrains and $\theta(x) = 0$ if $x \le 0$ and $\theta(x) = 1$ if $x > 0$. $s_{a,j}$ is the sign of the edges and defined as above. The message from a variable to a factor is:

$$n_{i\to a} = \left(\sum_{b\in \mathrm{nb}_+(i)\setminus\{a\}} m_{b\to i}\right) - \left(\sum_{b\in \mathrm{nb}_-(i)\setminus\{a\}} m_{b\to i}\right)$$

where $\mathrm{nb}_+(i)$ resp. $\mathrm{nb}_-(i)$ is the set of factors in which i appears unnegated resp. negated.

The lifted WP equations take the following form[1]:

$$m_{a\to i,p} = \prod_{j\in\mathrm{nb}(a)} \prod_{p'\in P(a,j)} \theta(n_{j\to a,p'} s_{p'})^{\mathrm{vc}(a,j,p') - \delta_{ij}\delta_{pp'}}$$

with $s_{p'} = 1$ if $p' = 0$ and $s_{p'} = -1$ if $p' = 1$. Here, $P(a,j)$ only runs over positions with a variable count greater than zero. Reconsidering the factor graph in Fig. 1 **(right)**, P contains both positions for the edge $sn_1 - sf_1$, while for $sn_2 - sf_2$ it only contains the unnegated position. The Kronecker deltas take care of the "$\setminus\{i\}$" in the product's range of the ground formula. They reduce counts when a message is sent to the node itself that means it prevents double counting of messages. Intuitively, this already shows that the lifted formula indeed simulates WP on the ground network. It even becomes more clear when we run it on the ground network shown in 1 **(left)**. In this case, there is exactly one position with a variable count greater than zero for each neighbor j of factor a. In other words, $P(a,j)$ is a singleton. In turn, the second product can be dropped, and both formulas coincide.

Similarly, one can intuitively check that the following formula is correct for the lifted case:

$$\begin{aligned} n_{i\to a,p} =& \Big(\sum_{\substack{b\in\mathrm{nb}(i),\\ \mathrm{fc}(b,i,1)>0}} m_{b\to i}(\mathrm{fc}(b,i,1) - \delta_{ab}\delta_{p1})\Big) - \\ & \Big(\sum_{\substack{b\in\mathrm{nb}(i),\\ \mathrm{fc}(b,i,0)>0}} m_{b\to i}(\mathrm{fc}(b,i,0) - \delta_{ab}\delta_{p0})\Big)\end{aligned}$$

[1]We define $0^0 = 1$ in cases where $\theta(x) = 0$ and $\mathrm{vc}(a,j,p') - \delta_{ij,pp'} = 0$.

Again the δs take care of the "$\setminus\{a\}$", the "$\mathrm{fc}(b,j,1/0) > 0$" of "$\mathrm{nb}_{+/-}(j)$". Using this correspondence, the proof that applying the lifted formulas to the lifted graph gives the same results as WP applied to the ground network follows essentially from the one for LPB (Singla and Domingos 2008).

Step 2: Lifted Survey Propagation

In survey propagation, the intuition of a message $m_{a\to i}$ sent from a factor to a variable is that of a "survey" which represents the probability that a warning is sent from a to i. Essentially, the SP equations can be lifted in the very same way as for WP. That is, we apply rewriting rules such as "$\setminus\{i\}$"$\mapsto$"$\delta_{ij}\delta_{pp'}$" to the ground SP equations. This yields the lifted SP equations as listed in Table 1.

Indeed, we have to be a little bit more careful. More precisely, for messages from factors to variables we use exactly the same rules. For the messages from variables to factors, however, we use slightly different rules. The set "$b \in \mathrm{nb}_a^{\mathbf{u/s}}(i)$" maps to "$b \in \mathrm{nb}(i), p' \in P(b,i), p' \neq p$" resp. to "$\ldots, p' = p$" for the message sent at position p. This ensures that we consider exact those neighbors of i that tend to make it **u**nsatisfy resp. **s**atisfy the clause a.

Now, we can safely apply the rewrite rules to the ground survey propagation equations as they can e.g. be found in Braunstein et al. (2005). They look as follows:

$$m_{a\to i} = \prod_{j\in\mathrm{nb}(a)\setminus\{i\}} \left[\frac{n^u_{j\to a}}{n^u_{j\to a} + n^s_{j\to a} + n^0_{j\to a}}\right]$$

$$n^u_{i\to a} = \left[1 - \prod_{b\in\mathrm{nb}^u_a(i)} (1 - m_{b\to i})\right] \prod_{b\in\mathrm{nb}^s_a(i)} (1 - m_{b\to i})$$

$$n^s_{i\to a} = \left[1 - \prod_{b\in\mathrm{nb}^s_a(i)} (1 - m_{b\to i})\right] \prod_{b\in\mathrm{nb}^u_a(i)} (1 - m_{b\to i})$$

$$n^0_{i\to a} = \prod_{b\in\mathrm{nb}(i)\setminus\{a\}} (1 - m_{b\to i})$$

where $\mathrm{nb}^u_a(i) = \mathrm{nb}_+(i)$ and $\mathrm{nb}^s_a(i) = \mathrm{nb}_-(i)\setminus\{a\}$ if $s_{a,i} = 1$, while $\mathrm{nb}^u_a(i) = \mathrm{nb}_-(i)$ and $\mathrm{nb}^s_a(i) = \mathrm{nb}_+(i) \setminus \{a\}$ if $s_{a,i} = -1$. In the following we will investigate different SAT solving scenarios using message passing algorithms.

Experimental Evaluation

Our intention here is to empirically investigate the correctness of the lifted message updates for WP and SP. To this aim, we implemented lifted SP and its variants in Python using the LIBDAI library (Mooij 2009). More precisely, we implemented decimation approaches. Decimation is the process of assigning a truth value to one variable (or a few variables) of F and simplifying F, obtaining a smaller formula on $n - 1$ variables. Now, we repeatedly decimate the formula in this manner, until all variables have been fixed. As a proof of concept, we compared the performances of lifted message passing approaches with the corresponding ground

(1) Message sent from factor a to a variable i at position p:

$$m_{a \to i,p} = \prod_{\substack{j \in \mathrm{nb}(a),\\ p' \in P(a,j)}} \left[\frac{n^u_{j \to a,p'}}{n^u_{j \to a,p'} + n^s_{j \to a,p'} + n^0_{j \to a,p'}} \right]^{\mathrm{vc}(a,j,p') - \delta_{ij}\delta_{pp'}}$$

(2) Messages sent from variable j to factor a at position p':

$$\begin{aligned}
n^u_{i \to a,p} &= \left[1 - \prod_{\substack{b \in \mathrm{nb}(i),\\ p' \in P(b,i), p' \neq p}} (1 - m_{b \to i})^{\mathrm{fc}(b,i,p') - \delta_{ab}}\right] \prod_{\substack{b \in \mathrm{nb}(i),\\ p' \in P(b,i), p' = p}} (1 - m_{b \to i})^{\mathrm{fc}(b,i,p') - \delta_{ab}}\\
n^s_{i \to a,p} &= \left[1 - \prod_{\substack{b \in \mathrm{nb}(i),\\ p' \in P(b,i), p' = p}} (1 - m_{b \to i})^{\mathrm{fc}(b,i,p') - \delta_{ab}}\right] \prod_{\substack{b \in \mathrm{nb}(i),\\ p' \in P(b,i), p' \neq p}} (1 - m_{b \to i})^{\mathrm{fc}(b,i,p') - \delta_{ab}}\\
n^0_{i \to a,p} &= \prod_{\substack{b \in \mathrm{nb}(i)\\ p' \in P(b,i)}} (1 - m_{b \to i})^{\mathrm{fc}(b,i,p') - \delta_{ab}\delta pp'}
\end{aligned}$$

Table 1: The lifted update formulas for survey propagation equations. For details, we refer to the main text.

versions on standard CNF benchmarks. We use decimation to measure the efficiency of the algorithms.

To assess performance, we report running times. Running time is measured by the number of messages sent. For the typical message sizes, e.g., for binary random variables with low degree, computing color messages is essentially as expensive as computing the actual messages. Therefore, we view the running time in terms of the number of both color and (modified) messages computed, treating individual message updates as atomic unit time operations. For BP, we used the “flooding” message protocol, the most widely used and generally best-performing method for BP. Here, messages are passed from each variable to all corresponding factors and back at each step. Messages were damped by 0.4, and the convergence threshold was 10^{-3}. For WP we used a sequential random update schedule and initialized all message to zero. SP used a sequential random update as well, but the messages were randomly initialized.

We evaluated (lifted) WP+BP decimation on the circuit synthesis problem `2bitmax_6`. The formula has 192 variables and 766 clauses. The resulting factor graph has 192 variable nodes, 766 factor nodes, and 1800 edges. The statistics of the runs are shown in Fig. 3 **(left)**. As one can see, lifting yields significant improvement in efficiency especially in the first iterations: LIFTED WP + BP reduces the messages sent by 13% compared to GROUND WP + BP when always fixing the most magnetized (largest difference between negated and unnegated marginals) variable.

Then, we investigated the Latin square construction problem `ls8-norm`. The formula has 301 variables and 1601 clauses, resulting in a factor graph with 3409 edges. The statistics of running lifted message-passing are shown in Figure 3 **(middle)**. Again, the lifting yields improvement in efficiency: LIFTED WP + BP saved 16% of the messages sent by the ground version.

Finally[2], we also applied the lifted message passing algorithms to a random 3-CNF, namely `wff.3.150.525`. The CNF contains 150 variables, 525 clauses and 1575 edges. As expected, no lifting was possible as shown in 3 **(right)**.

We also ran ground and lifted SP on these three problem instances. Due to the properties of the CNFs, SP always converged to a paramagnetic state, i.e. the trivial solution, with a single iteration. In such cases the problem is usually passed to a SAT-solver such as WalkSAT. Note, however, that the lifted SP saved roughly 45% of the messages the ground version sent on the `2bitmax_6` problem. On the latin square, lifted SP sent only 44% of the number of messages ground SP sent. As expected, the lifting produced a small overhaed on the random CNF for the lifted SP.

To summarize, the results clearly show that a unified lifted message-passing framework that includes BP, WP, and SP is indeed useful for SAT. Not only are significant efficiency gains obtainable by lifting, but lifted WP-guided BP clearly outperforms unguided (lifted) BP.

Conclusions

We have presented the first lifted message-passing algorithms for determining the satisfiability of Boolean formulas. Triggered by the success of recent lifted belief propagation approaches, our algorithms construct a network of supernodes and superfactors, corresponding to sets of nodes and factors that are sending and receiving the same messages, and apply warning resp. survey propagation to this network. Our initial experimental results validate the correctness of the resulting lifted approaches. Significant efficiency gains are obtainable when employing lifted instead of standard warning respectively survey propagation. In particular, lifted survey and warning propagation approaches are faster than ”just” using LBP to determine the satisfiability of Boolean formulas.

Indeed, much remains to be done. Next to running more experiments and developing lifted decimation and back-

[2]Note that we also conducted initial experiments on CNF produced from the ”Friends-and-Smokers” Markov logic network, see e.g. (Singla and Domingos 2008), for the case of deterministic clauses. Setting rather few unit clauses, however, either produced contradictions or solutions quickly. Therefore, we do not report results here. We are currently running a more detailed investigation.

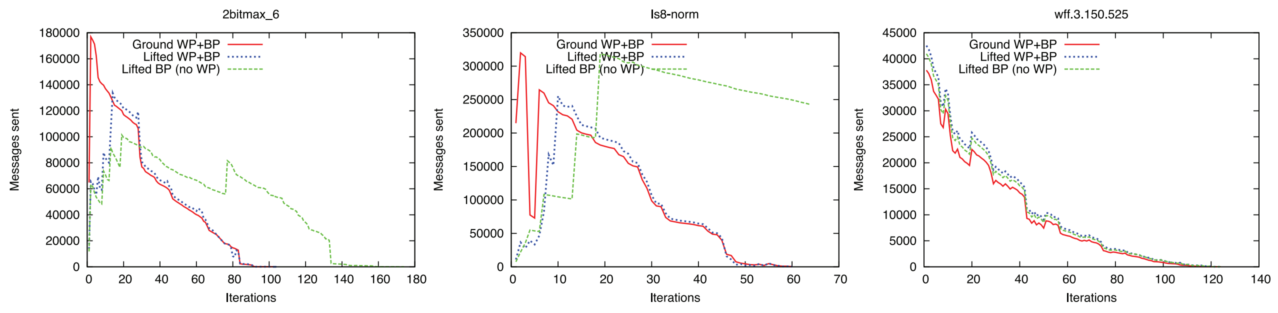

Figure 3: Total number of messages sent in each iteration of decimation for (lifted) WP+BP and lifted BP on several CNF benchmarks. **(Left)** `2bitmax_6`: Lifted WP+BP saves 13% of the messages ground WP+BP sent. **(Middle)** `ls8-norm`: Lifted WP+BP saves 16% of the messages ground WP+BP sent. **(Right)** `wff.3.150.525`. A small lifting overhead occurs.

tracking approaches, the most interesting avenue for future work is the tight integration of lifted SAT and lifted probabilistic inference. In many real-world applications, the problem formulation does not fall neatly into one of them. The problem may have a component that can be well-modeled as a SAT problem. Our work suggests to partition a problem into corresponding subnetworks, run the corresponding type of lifted message passing algorithm on each sub-network, and to combine the information from the different sub-networks. For the non-lifted case, this is akin to Duchi et al.'s (2006) COMPOSE approach that has been proven to be successful for problems involving matching problems as sub-problems. It is also very interesting to explore lifted inference for (stochastic) planning. Last but not least, we should start investigating lifted satisfiability for other classical AI tasks and, in turn, start exploring of what might be called Statistical Relational AI.

Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. This work is partially supported by the Fraunhofer ATTRACT scholarship STREAM and by the European Commission under contract number FP7-248258-First-MM.

References

Braunstein, A., and Zecchina, R. 2004. Survey propagation as local equilibrium equations. *J. Stat. Mech.* P06007.

Braunstein, A.; Mézard, M.; and Zecchina, R. 2005. Survey propagation: An algorithm for satisfiability. *Random Structures and Algorithms* 27(2):201–226.

Duchi, J.; Tarlow, D.; Elidan, G.; and Koller, D. 2006. Using combinatorial optimization within max-product belief propagation. In *Proc. of NIPS-06*, 369–376.

Kersting, K.; Ahmadi, B.; and Natarajan, S. 2009. Counting belief propagation. In A. Ng, J. B., ed., *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI–09)*.

Kroc, L.; Sabharwal, A.; and Selman, B. 2007. Survey propagation revisited. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI-07)*, 217–226.

Kroc, L.; Sabharwal, A.; and Selman, B. 2009. Messagepassing and local heuristics as decimation strategies for satisfiability. In *Proc. of the ACM Symposium on Applied Computing (SAC-09)*, 1408–1414.

Kschischang, F. R.; Frey, B. J.; and Loeliger, H.-A. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47.

Mézard, M.; Parisi, G.; and Zecchina, R. 2002. Analytic and algorithmic solution of random satisfiability problems. *Science* 297:812–815.

Milch, B.; Zettlemoyer, L.; Kersting, K.; Haimes, M.; and Pack Kaelbling, L. 2008. Lifted Probabilistic Inference with Counting Formulas. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI-08)*.

Montanari, A.; Ricci-Tersenghi, F.; and Semerjian, G. 2007. Solving constraint satisfaction problems through belief propagation-guided decimation. In *Proc. of the 45th Allerton Conference on Communications, Control and Computing*.

Mooij, J. M. 2009. libDAI 0.2.3: A free/open source C++ library for Discrete Approximate Inference. http://www.libdai.org/.

Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI-2006)*.

Poon, H.; Domingos, P.; and Summer, M. 2008. A general method for reducing the complexity of relational inference and its application to mcmc. In *Proc. of the 23rd National Conference on Artificial Intelligence (AAAI-08)*, 1075–1080.

Sen, P.; Deshpande, A.; and Getoor, L. 2008. Exploiting Shared Correlations in Probabilistic Databases. In *Proc. of the Intern. Conf. on Very Large Data Bases (VLDB-08)*.

Singla, P., and Domingos, P. 2006. Memory-efficient inference in relational domains. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI-2006)*.

Singla, P., and Domingos, P. 2008. Lifted First-Order Belief Propagation. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI-08)*, 1094–1099.