Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)

Informed Lifting for Message-Passing

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

Abstract

Lifted inference, handling whole sets of indistinguishable objects together, is critical to the effective application of probabilistic relational models to realistic real world tasks. Recently, lifted belief propagation (LBP) has been proposed as an efficient approximate solution of this inference problem. It runs a modified BP on a lifted network where nodes have been grouped together if they have — roughly speaking — identical computation trees, the tree-structured unrolling of the underlying graph rooted at the nodes. In many situations, this purely syntactic criterion is too pessimistic: message errors decay along paths. Intuitively, for a long chain graph with weak edge potentials, distant nodes will send and receive identical messages yet their computation trees are quite different. To overcome this, we propose iLBP, a novel, easy-to-implement, informed LBP approach that interleaves lifting and modified BP iterations. In turn, we can efficiently monitor the true BP messages sent and received in each iteration and group nodes accordingly. As our experiments show, iLBP can yield significantly faster more lifted network while not degrading performance. Above all, we show that iLBP is faster than BP when solving the problem of distributing data to a large network, an important real-world application where BP is faster than uninformed LBP.

Introduction

Message passing algorithms, in particular belief propagation (BP), have been very successful in efficiently computing interesting properties of probability distributions. Moreover, they have also been proven to be successful in a number of important real-world AI tasks. Consider e.g. the problem of distributing data to a large network. A naive solution to this problem involves unicasting the data to individual nodes from the source node. This approach, however, does not scale well as packages are sent again and again. Another simple solution is server replication, a solution that might not be cost effective for all users. In contrast, peer-to-peer solutions divide the file into parts and the nodes exchange these parts until they re-assemble back to the complete file. Recently, Bickson *et al.* (2006) have shown how to use BP to solve this problem efficiently at realistic scale. More precisely, they have shown how to formalize the next-step problem — compute the next action that each peer in the content distribution network should take, where an action is the transfer of one file part from a neighboring node — as an undirected graphical model and ran BP on this model in order to find the next best action.

Many graphical models, however, produce inference problems with symmetries not reflected in the local graphical structure, and hence not exploited by BP. One of the most prominent examples are first-order and relational probabilistic models, see e.g. (Getoor and Taskar 2007). They are knowledge representations that combine aspects of first-order logic and probability, a long-standing goal of AI. Underlying all of them are standard graphical models. More precisely, creating all ground atoms and formulas induces a standard graphical model that typically has a lot of symmetries, in particular, factors shared across many groups of objects. Triggered by this insight, lifted BP approaches have been recently developed (Singla and Domingos 2008; Kersting, Ahmadi, and Natarajan 2009). They automatically group together nodes and factors into supernodes and superfactors if they have — roughly speaking — identical computation trees, the tree-structured unrolling of the graphical model rooted at the nodes. Then, they run a modified BP on the resulting lifted, i.e., clustered network. Being instances of BP, they can be extremely fast at computing approximate marginal probability distributions. So far, significant efficiency gains have been reported on entity recognition, link prediction, social network analysis, and Boolean model counting problems.

The first contribution of this paper is to show the power of lifted inference for another important AI task, namely, the content distribution problem, a problem that has not been studied in the statistical relational reasoning and learning community so far. This is somewhat surprising. YouTube like media portals have changed the way users access media content in the Internet. Every day, millions of people visit social media sites such as Flickr, YouTube, and Jumpcut, among others, to share their photos and videos, while others enjoy themselves by searching, watching, commenting, and rating the photos and videos; what your friends like will bear great significance for you. The vision of social search underlies the great success of all the recent so-

cial platforms. However, while many of these services are centralized, i.e., rely on server replication, the full flexibility of social information-sharing can often be better realized through direct sharing between peers. So, the question naturally arise "Does Bickson *et al.*'s graphical model for solving the next-step problem also produce inference problems with symmetries not reflected in the graphical structure?" An investigation of this question was the seed that grew into our main contribution: *informed* lifted BP.

In fact, simply applying lifted BP (LBP) to Bickson *et al.*'s graphical model did not succeed. Although there are symmetries exploitable for lifting, standard BP runs simply too quickly: it often converges within few iterations. In contrast, LBP's preprocessing step for lifting the network takes more iterations, does not produces any belief estimates, and may yield lifted networks that are too pessimistic. The situation is depicted in Fig. 1. It shows the error curves of running (L)BP on a local snapshot of a real-word file sharing network. As one can see, BP only takes 5 iterations, whereas lifted BP first takes 7 iterations of a color-message passing scheme to compute the lifted network and only then runs 5 highly efficient iterations simulating BP on the lifted network. This is slower than BP because:

- Computing the color-messages often takes essentially as much time as computing the BP messages, in particular for discrete nodes with few states and low degree.
- The lifting step is purely syntactic. Colors only denote the nodes and factors producing BP messages. Neither BP messages nor any belief estimates are actually computed.
- In turn, LBP cannot make use of the fact that BP message errors decay along paths (Ihler, Fisher III, and Willsky 2005) already at lifting time. It may spuriously assume some nodes send and receive different messages and, hence, produce pessimistic lifted network.

Informed lifted BP (iLBP) overcomes these problems. It is a novel, easy-to-implement, lifted BP approach that tightly interleaves lifting and modified BP iterations. This allows one to efficiently monitor the true BP messages sent and received in each iteration and to group nodes accordingly. As our experiments show, significant efficiency gains are obtainable: iLBP can yield significantly faster higher lifted network while not degrading performance. Above all, we show that iLBP is faster than BP when solving the problem of distributing data to a large network of fixed, known topology, an important real-world application where BP is faster than (uninformed) LBP.

We proceed as follows. After touching upon related work, we briefly review LBP. Then, we introduce iLBP and prove its correctness. Before concluding, we present our experimental results including an evaluation of iLBP for the content distribution problem in real-world networks.

Related Work

Recent years have witnessed a surge of interest in lifted probabilistic inference. Poole (2003), de Salvo Braz *et al.* (2005), and Milch *et al.* (2008) have developed lifted versions of the variable elimination algorithm. 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. Shavlik and Natarajan (2009) proposed a preprocessing algorithm that can reduce the effective size of Markov logic networks (Richardson and Domingos 2006) and, hence, can speed up greatly inference. All of these approaches, however, require a first-order logical specification of the model; iLBP does not.

Recently, Sen *et al.* (2008; 2009) presented a lifted variable elimination approach based on bisimulation. As iLBP, it essentially uses identical computation trees to group together indistinguishable objects. In contrast to the previous approaches, however, it does not require a first-order logical specification. The simplest and hence most efficient lifted inference approaches are the already mentioned lifted belief propagation approaches. They easily scale to realistic domains. Motivated by Jaimovich *et al.* (2007), Singla and Domingos (2008) developed the first lifted belief propagation variant requiring a Markov logic network as input. Subsequently, Kersting *et al.* (2009) generalized it to any factor graph over discrete, finite variables. Both of them perform lifting in an uninformed fashion and have not been used for solving the content distribution problem.

Probably the closest work to iLBP is de Salvo Braz *et al.*'s (2009) anytime LBP. It also performs lifting during message passing. In contrast to iLBP, however, it uses box (and not belief) propagation only on a subset of the model, but with supernodes, as in lifted BP. It also requires a first-order logical specification.

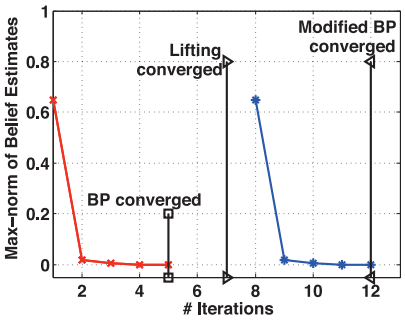

Figure 1: Max-norm of belief estimates vs. number of iterations of (L)BP on a factor graph representing the collaboration graph of a local snapshot of the Gnutella network (1374 nodes, 4546 edges). BP converges within 5 iterations. LBP first lifts the network using color-passing (7 iterations) without estimating any beliefs at all; then it runs the highly efficient modified BP on the lifted network. (Best viewed in color)

Lifted Belief Propagation

Let $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ be a set of n discrete-valued random variables each having d states, and let x_i represent the possible realizations of random variable X_i. Graphical models compactly represent a joint distribution over $\mathbf{X}$ as a product of factors (Pearl 1991), i.e., $P(\mathbf{X} = \mathbf{x}) =$

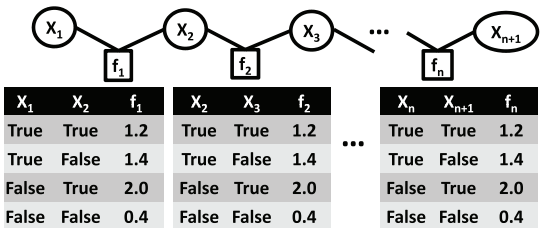

Figure 2: An example for a factor graph — a chain graph model with $n+1$ nodes — with associated potentials. Circles denote variables, squares denote factors.

$Z^{-1}\prod_k f_k(\mathbf{x}_k)$. Each factor f_k is a non-negative function of a subset of the variables $\mathbf{x}_k$, and Z is a normalization constant. If $P(\mathbf{X}=\mathbf{x})>0$ for all joint configurations $\mathbf{x}$, the distribution can be equivalently represented as a log-linear model: $P(\mathbf{X}=\mathbf{x}) = Z^{-1}\exp\left[\sum_i w_i \cdot g_i(\mathbf{x})\right]$, where the factors $g_i(x)$ are arbitrary functions of (a subset of) the configuration $\mathbf{x}$. Each graphical model can be represented as a factor graph. A factor graph, cf. Fig 2, is a bipartite graph that expresses the factorization structure of the joint distribution. It has a variable node (denoted as a circle) for each variable X_i, a factor node (denoted as a square) for each f_k, with an edge connecting variable node i to factor node k if and only if X_i is an argument of f_k. We assume one factor $f_i(\mathbf{x}) = \exp\left[w_i \cdot g_i(\mathbf{x})\right]$ per feature $g_i(\mathbf{x})$.

An important (#P-complete) inference task is to compute the conditional probability of variables given the values of some others, the evidence, by summing out the remaining variables. The belief propagation (BP) algorithm is an efficient way to solve this problem that is exact when the factor graph is a tree, but only approximate when the factor graph has cycles. Although this loopy BP has no guarantees of convergence or of giving the correct result, in practice it often does, and can be much more efficient than other methods (Murphy, Weiss, and Jordan 1999).

BP can elegantly be described in terms of operations on a factor graph. To do so, we first introduce messages between variable nodes and their neighboring factor nodes and vice versa. The message from a variable X to a factor f is

$$\mu_{X\to f}(x) = \prod\nolimits_{h\in \mathrm{nb}(X)\setminus\{f\}} \mu_{h\to X}(x)$$

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

$$\mu_{f\to X}(x) = \sum_{\neg\{X\}}\left(f(\mathbf{x}) \prod_{Y\in\mathrm{nb}(f)\setminus\{X\}} \mu_{Y\to f}(y)\right)$$

where $\mathrm{nb}(f)$ are the arguments of f, and the sum is over all of these except X, denoted as $\neg\{X\}$. The messages are usually initialized to 1, and the unnormalized belief of each variable X_i can be computed from the equation $b_i(x_i) = \prod_{f\in\mathrm{nb}(X_i)} \mu_{f\to X_i}(x_i)$. Evidence is incorporated by setting $f(\mathbf{x})=0$ for states $\mathbf{x}$ that are incompatible with it. Different schedules may be used for message-passing.

Although already quite efficient, many graphical models produce factor graphs with symmetries not reflected in the graphical structure. Reconsider the factor graph in Fig. 2. The associated potentials are identical. Lifted BP (LBP) can make use of this fact. Essentially, as e.g. described in (Kersting, Ahmadi, and Natarajan 2009), it performs two steps: Given a factor graph G, it first computes a compressed factor graph $\mathfrak{G}$ and then runs a modified BP on $\mathfrak{G}$. (We use fraktur letters such as $\mathfrak{G}$, $\mathfrak{X}$, and $\mathfrak{f}$ to denote the lifted, i.e., compressed graphs, nodes, and factors).

(Step 1) Lifting by Color-Passing (CP): Initially, all variable nodes fall into the $d+1$ groups (one or more of these may be empty), namely known state $s_1,\ldots,$ known state s_d, and *unknown*. For ease of explanation, we will represent the groups by colors (respectively shades). All factor nodes with the same associated potentials also fall into one group represented by a color. Now, each variable node sends a message to its neighboring factor nodes saying "I am of color C". 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. This color signature is sent back to the neighboring variables nodes, essentially saying "You have communicated with these kinds of nodes". The variable nodes stack the incoming signatures together and, hence, form unique signatures of their one-step message history. Variable nodes with the same stacked signatures are grouped together, and a new color is assigned to each group. The factors are grouped in a similar fashion based on the incoming color signatures of neighboring nodes. This CP process is iterated until no new colors are created anymore. As the effect of the evidence propagates through the factor graph, more groups are created.

The final lifted graph $\mathfrak{G}$ is constructed by grouping all nodes (factors) with the same color (signatures) into *supernodes* (*superfactors*). Supernodes (superfactors) are sets of nodes (factors) that send and receive the same messages at each step of carrying out BP on G and form a partition of the nodes in G.

(Step 2) Modified BP (MBP) on the Lifted Graph[1]: The basic idea is to run a modified BP on $\mathfrak{G}$ that simulates BP on G. An edge from a superfactor $\mathfrak{f}$ to a supernode $\mathfrak{X}_i$ in $\mathfrak{G}$ essentially represents multiple edges in G. Let $c(\mathfrak{f},\mathfrak{X}_i)$ be the number of identical messages that would be sent from the factors in the superfactor $\mathfrak{f}$ to each node in the supernode $\mathfrak{X}_i$ if BP was carried out on G. The message from a supervariable $\mathfrak{X}$ to a superfactor $\mathfrak{f}$ is $\mu_{\mathfrak{X}\to\mathfrak{f}}(x) =$

$$\mu_{\mathfrak{f}\to\mathfrak{X}}(x)^{c(\mathfrak{f},\mathfrak{X})-1}\cdot\prod\nolimits_{\mathfrak{h}\in\mathrm{nb}(\mathfrak{X})\setminus\{\mathfrak{f}\}}\mu_{\mathfrak{h}\to\mathfrak{X}}(x)^{c(\mathfrak{h},\mathfrak{X})}$$

where $\mathrm{nb}(\mathfrak{X})$ now denotes the neighbor relation in the lifted graph $\mathfrak{G}$. The $c(\mathfrak{f},\mathfrak{X})-1$ exponent reflects the fact that a supervariable's message to a superfactor excludes the corresponding factor's message to the variable if BP was carried out on G. Finally, the unnormalized belief of $\mathfrak{X}_i$, i.e., of any node X in $\mathfrak{X}_i$ can be computed from the equation $b_i(x_i) = \prod_{\mathfrak{f}\in\mathrm{nb}(\mathfrak{X}_i)}\mu_{\mathfrak{f}\to\mathfrak{X}_i}(x_i)^{c(\mathfrak{f},\mathfrak{X})}$. Evidence is incorporated as in standard BP.

[1]For the sake of simplicity, we present simplified equations that neglect the positions a supernode may appear in a superfactor.

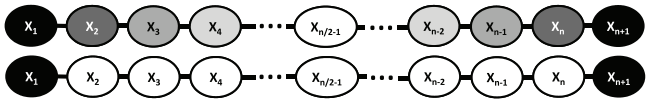

Figure 3: Supernodes — indicated by the colors of the original nodes — produced by uninformed **(top)** and informed **(bottom)** lifting on a chain graph model with $n+1$ nodes and identical, weak edge potentials. Factors have been omitted. Uninformed lifting produces $n/2-1$ many supernodes whereas informed lifting takes advantage of decaying message errors and produces only 2 supernodes: one for the end nodes of the chain and one for the other nodes.

Informed Lifting

Let us analyze how LBP behaves on the simple chain model from Fig. 2, now assuming weak (but still identical) edges potentials. The lifted graph produced by LBP is shown in Fig. 3 **(top)**. Due to the purely syntactic "lifting" criterion used by CP — *group nodes according to identical (colored) computation trees*, the tree-structured unrolling of the underlying graph rooted at the nodes — nodes with the same distance to one of the ends of the chain are grouped together. Consequently, $n/2+1$ many supernodes are produced. Can we do any better?

It was shown by Ihler *et al.* (2005) that message errors decay along paths. Intuitively, for long chain graphs — they are also a subproblem in both acyclic and cyclic graphical models — with weak edge potentials, distant nodes are approximately independent. So, the question naturally arises: "How can we make use of decaying errors with the goal to produce higher lifted graphs?"

Intuitively, making use of it would be easy, if we knew the true BP messages sent and received in each iteration of BP. We run one iteration of BP, color nodes and factors according to the true BP messages they sent and received, and iterate until the colors are not changing anymore. Because CP takes only finitely many iterations as proven in (Singla and Domingos 2008; Kersting, Ahmadi, and Natarajan 2009), it follows that this informed coloring indeed converges after a finite number of iterations. The question is when? In the worth case, we may run BP until convergence on the original graph before switching to the highly efficient MBP on the lifted graph. Hence, this naive solution cancels the benefits of lifted inference

Consequently, we propose an adaptive approach, called *informed* LBP, that interleaves CP and MBP iterations as summarized in Algorithm 1. Lines 1-4 simulate BP's first iteration. We cluster the nodes with respect to the evidence and run one iteration of CP. On the resulting lifted factor graph $\mathfrak{G}$, we run one iteration of MBP. This is simulating the initial BP iteration on the original factor graph G. Hence, we can make use of BP's 1-iteration messages $m_i(x_i)$ and belief estimates $b_i(x_i)$. Consequently, we can safely group together nodes according to the BP messages (line 5). In the while-loop (line 6), we repeat this process until the belief estimates converge. That is, we compute the (possibly refined) lifted network and run another CP-iteration (line 7). By induction, running MBP on the resulting lifted graph $\mathfrak{G}'$ (line 9) simulates the second BP iteration on the original graph G. Hence, we can re-colorize nodes according to the true BP messages (lines 10-12), and so on. It follows that iLBP produces the same results as BP.

Algorithm 1: iLBP – informed Lifted BP. We use $b_i(x_i)$ resp. $m_i(x_i)$ to denote the unnormalized beliefs resp. messages of both variable node X_i and variable nodes covered by supernodes $\mathfrak{X}_i$.

Data: A factor graph G with variable nodes X and factors f, Evidence E
Result: Unnormalized marginals $b_i(x_i)$ for all supernodes and, hence, for all variable nodes

1 Colorize X and f w.r.t. E;
2 $\mathfrak{G} \leftarrow$ one iteration CP;
3 Initialize messages for $\mathfrak{G}$;
4 $(b_i(x_i), m_i(x_i)) \leftarrow$ one iteration MBP on $\mathfrak{G}$;
5 Colorize all X_is according to $m_i(x_i)$;
6 **while** $b_i(x_i)$*s have not converged* **do**
7 $\quad \mathfrak{G}' \leftarrow$ one iteration CP (based on new colors);
8 $\quad$ Initialize novel supernodes using $b_i(x_i)$ and $m_i(x_i)$;
9 $\quad (b_i(x_i), m(x_i)) \leftarrow$ one iteration of MBP on $\mathfrak{G}'$;
10 $\quad$ **foreach** *supernode* $\mathfrak{X}$ *in* $\mathfrak{G}$ **do**
11 $\quad\quad$ **if** *the* $m(x_i)$*s of the* X_i*s in* $\mathfrak{X}$ *differ* **then**
12 $\quad\quad\quad$ Colorize all X_i in $\mathfrak{X}$ according to $m(x_i)$
13 $\quad\quad$
14 $\quad$
15 Return $b_i(x_i)$ for all supernodes

Theorem 1. *Given a factor graph G and some evidence E, iLBP produces the same results as running BP applied to G. In turn, it produces the same results as running LBP.*

Note that, to reduce the $\mathcal{O}(n^2)$ effort for re-coloring in each iteration, line 10 iterates over all supernodes and checks whether any re-colorizing is required at all. This takes $\mathcal{O}(n)$ time. If a change has been detected, only the variable nodes corresponding to the supernode at hand are re-colorized. We also note that in models over discrete variables with few states and low degree as often produced by Markov logic networks, iLBP is very efficient: computing color messages is essentially as expensive as computing BP messages.

The lifted graph produced by iLBP on our chain model example is shown in Fig. 3 **(bottom)**. Independently of the domain size $n+1$, only 2 supernodes are produced, an order of magnitude less than LBP produces. As our experimental results suggest, iLBP seems generally to produce highly lifted graphs extremely quickly. Moreover, in contrast to LBP, belief estimates are computed (1) from the very first iteration on (2) using MBP, i.e., really lifted inference.

Experimental Evaluation

Our intention here is to investigate the following questions: Can iLBP be faster than LBP, also when BP is faster than LBP? Can it yield more compressed lifted networks than LBP?

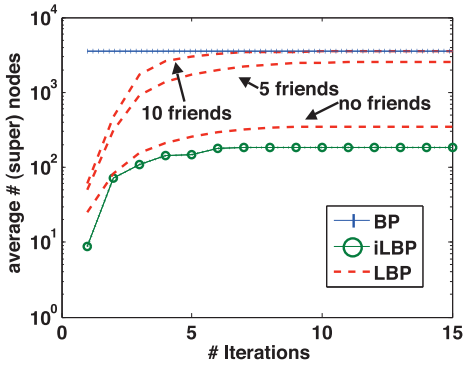

Figure 4: Number of super(nodes) (in log-scale) vs. iterations (including CP iterations for LBP) averaged over 5 reruns for the social network, dynamic MLN. The number of (super)nodes roughly corresponds to the efficiency, i.e., the number of messages sent. As one can see, LBP's performance degrades with more evidence. In contrast, iLBP remains uneffect. In particular, iLBP produces less many supernodes. (Best viewed in color)

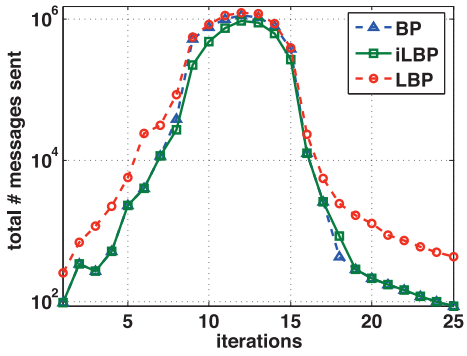

Figure 5: Total sum of messages (in log-scale) vs. content distribution decimation iterations on a Gnutella snapshot (10876 nodes, 39994 edges). The number of messages essentially corresponds to the efficiency. As one can see iLBP is the most efficient one. (Best viewed in color)

To this aim, we implemented ((i)L))BP in Python using the LIBDAI library (Mooij 2009) and evaluated their performances on the dynamic "Friends-and-Smokers Markov logic network (MLN) as well as for solving the content distribution problem on snapshots of real-world file-sharing networks. In all experiments, we used the "flooding" message protocol, the most widely used and generally best-performing method for static networks. Here, messages are passed from each variable to each corresponding factor and back at each step. Messages were not damped, and the convergence threshold was 10^{-8}.

Social Networks: In our first experiment, we considered the dynamic "*Friends & Smokers*" MLN as introduced in (Kersting, Ahmadi, and Natarajan 2009) for 20 people and 10 time steps. This created a ground factor graph consisting of 4400 nodes and 8420 factors. For 5 randomly selected people, we randomly choose whether they smoke or not and who $0, 5$ resp. 10 of their friends are, and randomly assigned a time step to the information. Other friendship relations are still assumed to be unknown. Whether a person has cancer or not is unknown for all persons and all time points.

The goal is to see whether iLBP can produces more lifted networks than LBP. The number of (super)nodes vs iterations (averaged over 5 reruns) are summarized in Fig. 4. As one case see, iLBP can indeed produce less many supernodes and, hence, be faster than LBP. Most importantly, LBP cannot deal as good with more evidence as LBP: it is rather uneffected. Finally, the max-norm of iLBP was always below 10^{-8}.

Content Distribution Problem (CDP): Finally, we investigated the CDP. Bickson *et al.* assume that at the beginning of the download there is a source peer that contains all the file parts and that the other peers have no parts of the file. Starting from an initial graphical model, they run a decimation approach for solving the CDP: (1) solve the next step problem running the BP max-product inference algorithm, (2) use the MAP assignment result as the solution, (3) simplify the graphical model according to it, (4) repeat. In this paper, we use ((i)L)BP in step (1).

Bickson *et al.* construct the graphical model essentially as follows (for more details, we refer to (Bickson, Dolev, and Weiss 2006). The peers participating in the download form the nodes of the underlying graphical model. Each peer/node has only local knowledge: the parts, which it presently holds, and the list of its direct neighbors and the parts they hold. From this, all actions a peer can take follow such as "*download part i from neighbor j*". The initial probability of taking an action is encoded in the node potentials. There are several heuristic for initializing them. Here, we focused on the *uniform* policy that assigns equal probability to all parts. Similar results have been achieved using the *rarest part first* policy that assigns the highest probability to the part that has the lowest frequency in the network. Additionally, there is an edge potential between two peers/nodes that can take a part from a common neighbor. The edge potentials coordinate the actions among the peers. That is, we assign a very small value to pairs of actions that involves taking a part from the same third node: *at any time, only one peer can download a part from another peer*. The other values are set according to the node potentials, see (Bickson, Dolev, and Weiss 2006) for more details.

We simulated to distribute a single file(part) through a snapshot of the Gnutella network[2]. The max number of ((i)L)BP iterations per decimation round was set to 40. The results are summarized in Fig. 5. As one can see, iLBP can indeed be faster than BP in cases where LBP is slower than BP. In total, BP sent 5761952 messages, iLBP only 4272164 messages, and LBP 6381516 messages (including color messages). On a similar run on a different Gnutella network[3] consisting of 6301 nodes and 20777

[2]http://snap.stanford.edu/data/p2p-Gnutella04.html
[3]http://snap.stanford.edu/data/p2p-Gnutella08.html

the approaches behaved similar. In total, BP sent 5761952 messages, iLBP only 1972662 messages, and LBP 2962311 messages (including color messages). Again, the max-norm of iLBP was always below 10^{-8}. Finally, quantitatively similar results have been achieved when distributing multiple file parts.

To summarize, our questions can be answered affirmatively: iLBP can produce much higher lifted networks in less time than LBP while not degrading accuracy. Beliefs are already estimated at lifting time and they converge to the same estimates as (L)BP produce.

Conclusions

We have introduced *informed lifted BP*, a novel, generally applicable BP algorithm for lifted inference. By interleaving lifting and modified BP iterations, it can group together nodes resp. factors if their actual BP messages are identical. Intuitively, it adaptively explores the subgraph around each node resp. factor and groups them on an as-needed basis making use of decaying error messages. Several experiments have shown that in turn significant efficiency gains are obtainable compared to both BP and lifted BP, often orders of magnitude. Most importantly, informed lifting allows one to solve the important AI task of distributing content to large networks more efficiently than standard BP; a problem where lifting did not pay of so far.

There are several attractive avenues of future work. Formalizing relational content distribution rules along the lines of (Brafman 2006) would allow one to compactly represent rich communication networks. Lifting the adaptive case, when nodes and edges may enter and leave the network over time, is not only important for the content distribution problem but also e.g. for fraud detection in online auctions (Pandit et al. 2007) or computing data-oriented partitions in large overlay networks (Aberer et al. 2006). Finally, informed lifting paves the way to easy-to-implement lifted versions of other message passing algorithms such as generalized BP and convergent BP variants as well as survey propagation.

Acknowledgements: The work was supported by the Fraunhofer ATTRACT fellowship STREAM.

References

Aberer, K.; Bickson, D.; Dolev, D.; Hauswirth, M.; and Weiss, Y. 2006. Indexing Data-Oriented Overlay Networks using Belief Propagation. In *Proc. of the 7th Workshop of Distributed Algorithms and Data Atructures (WDAS-06)*.

Bickson, D.; Dolev, D.; and Weiss, Y. 2006. Efficient Large Scale Content Distribution. Technical Report Leibniz Center TR-2006-07, School of Computer Science and Engineering, The Hebrew University.

Brafman, R. 2006. Relational Preference Rules for Control. In *Proc. the 11th Intern. Conf. on Principles of Knowledge Representation and Reasoning (KR-08)*, 552–559.

de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted First Order Probabilistic Inference. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 1319–1325.

de Salvo Braz, R.; Natarajan, S.; Bui, H.; ; Shavlik, J.; and Russell, S. 2009. Anytime lifted belief propagation. In *Working Notes of the International Workshop on Statistical Relational Learning (SRL-09)*.

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

Ihler, A.; Fisher III, J.; and Willsky, A. 2005. Loopy Belief Propagation: Convergence and Effects of Message Errors. *Journal of Machine Learning Research* 6:905–936.

Jaimovich, A.; Meshi, O.; and Friedman, N. 2007. Template-based Inference in Symmetric Relational Markov Random Fields. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI-07)*, 191–199.

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)*.

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)*.

Mooij, J. M. 2009. libDAI 0.2.3: A Free/Open Source C++ Library for Discrete Approximate Inference. http://www.libdai.org/.

Murphy, K.; Weiss, Y.; and Jordan, M. 1999. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI-99)*, 467–475.

Pandit, S.; Chau, D.; Wang, S.; and Faloutsos, C. 2007. Netprobe: A Fast and Scalable System for Fraud Detection in Online Auction Networks. In *Proc. of the 16th International Conference on World Wide Web (WWW-07)*, 301–210.

Pearl, J. 1991. *Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2. edition.

Poole, D. 2003. First-Order Probabilistic Inference. In *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 985–991.

Richardson, M., and Domingos, P. 2006. Markov Logic Networks. *Machine Learning* 62:107–136.

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)*.

Sen, P.; Deshpande, A.; and Getoor, L. 2009. Bisimulation-based Approximate Lifted Inference. In A. Ng, J. B., ed., *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI–09)*.

Shavlik, J., and Natarajan, S. 2009. Speeding Up Inference in Markov Logic Networks by Preprocessing to Reduce the Size of the Resulting Grounded Network. In *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 1951–1956.

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.