# Initial Empirical Evaluation of Anytime Lifted Belief Propagation

**Richard G. Freedman, Rodrigo de Salvo Braz∗, Hung Bui∗, Sriraam Natarajan**
Wake Forest University, ∗ SRI International

## Abstract

Lifted first-order probabilistic inference, which manipulates first-order representations of graphical models directly, has been receiving increasing attention. Most lifted inference methods to date need to process the entire given model before they can provide information on a query's answer, even if most of it is determined by a relatively small, local portion of the model. Anytime Lifted Belief Propagation (ALBP) performs Lifted Belief Propagation but, instead of first building a supernode network based on the entire model, incrementally processes the model on an as-needed basis, keeping a guaranteed bound on the query's answer the entire time. This allows a user to either detect when the answer has been already determined, before actually processing the entire model, or to choose to stop when the bound is narrow enough for the application at hand. Moreover, the bounds can be made to converge to the exact solution when inference has processed the entire model. This paper shows some preliminary results of an implementation of ALBP, illustrating how bounds can sometimes be narrowed a lot sooner than it would take to get the exact answer.

## 1 Introduction

Recently, there has been a surge in interest in the area of Statistical Relational Learning (SRL) [1] and several different representations have been proposed. The central problem in all these models is that of efficient inference. Earlier inference methods were based on sampling methods or methods that completely instantiate all the objects in the domain and hence were prohibitively expensive. Lifted inference [2, 3], inference that manipulates and keeps the first-order structure, avoiding extensive propositionalization, has been receiving increasing attention recently. To date, all lifted inference methods require a model to be *shattered* against itself and evidence, before inference starts. Shattering means dividing the random variables of the model into clusters of exactly symmetric variables. Evidence is often provided at the level of random variables on specific individuals, typically causing all random variables involving them to form singleton clusters. For many problems this is very close to propositionalization, and the gains from lifted inference are greatly decreased.

The reason shattering is needed in advance is because the algorithms that have been lifted (belief propagation and variable elimination) do require the entire model in order to compute a query's belief. So in general the entire model needs to be used, requiring it to be entirely shattered. However recent work on *box propagation* [4] shows how to derive *bounds* on beliefs from using only a portion of a model. This allows us to *gradually* shatter the model while obtaining useful bounds on the query. Interestingly, this also corresponds to the intuition that reasoning only considers sub- or individual cases in an as-needed basis, as it is done in theorem proving where unification and resolution are gradually used.

The box propagation algorithm was later extended to the SRL case as Anytime Lifted BP (ALBP) [5]. The key idea in ALBP is to perform incremental shattering with the propagation of bounds. In this work, we extend the previous paper by presenting the pseudocode for the ALBP algorithm and by empirically evaluating the algorithm on a small domain. This is a work-in-progress and the results that we present are initial observations. Nonetheless, some of these results clearly show the advantage of box propagation when the bounds can be narrowed much sooner than converging to a point estimate. A similar approach has been taken in [6] where the messages are grouped after every few iterations without constructing the full lifted network.

## 2 Algorithmic Details

**Algorithm 1** Pseudocode explaining the iterative process of Anytime Lifted Belief Propagation.

```
for all Query q in queryList do
    rn ← createSupernode(lift(q));
    cn ← rn;                                            ▷ Start at query
    addSupernode(fg, rn);
    stillSplittingAndShattering ← true;
    ▷ Use rules and evidence to shatter and split
    while stillSplittingAndShattering do
        for all Rule r in ruleList do
            if (r contains cn.predicate) and
               (not splitBy(cn, r)) then
                performSplit(cn, r, fg);                ▷ Extends
            end if
        end for
        for all Evidence e in evidenceList do
            if (e contains cn.predicate) and
               (not shatteredBy(cn, e)) and
               (isSubset(e.constriants, cn.groundings))
               then
                performSplit(cn, e, fg);                ▷ Extends
                performShatter(cn, e, fg);              ▷ Breaks
            end if
        end for
        ▷ Ground the query on the first iteration
        if 1st iteration then
            performShatter(rn, q, fg);                  ▷ Breaks
        end if
        ▷ Box propagation gets marginal bound
        interval ← runBoxPropagation(rn, fg);
        if intervalinterval < δ then
            break;
        end if
        ▷ Select the next node to split and shatter
        cn ← getNextNode(fg, METHOD);
        if cn == null then
            stillSplittingAndShattering ← false;
        end if
    end while
end for
```

Lifted belief propagation [7] is based on the idea that symmetric variables (that is, variables with exactly the same set of dependencies) will receive and generate the same belief messages. It determines these sets (called *supernodes*) by shattering as a preprocessing step, and performs message passing between them. The main problem with this and the other lifted inference methods is that the nodes that are not even part of the evidence relevant to the query are shattered in advance. For example, consider the following two parfactors: $\forall Y \phi_1(funny(Y))$ and $\forall X, Y \phi_1(funny(Y), likes(X,Y))$. Now if the query is $P(funny(fred) \; | likes(Tom, Fred))$, and if it is observed that Tom is a friend of Fred, then by shattering, $likes(tom, Fred)$, $\langle likes(X, Fred), X \neq tom \rangle$ and $\langle likes(X,Y) \; X \neq tom \; Y \neq Fred \rangle$ will form the three clusters. Note that though there are no other evidence present, we still have to split $likes(X,Y)$ into two more clusters, one for $Y = Fred$ and one for $Y \neq Fred$. If we had another evidence, say that Mary is a friend of Fred, then we will have more clusters. As the amount of evidence increases, the number of shatterings also increases as the clusters may need to be shattered even if they are not part of the evidence.

ALBP on the other hand, does not consider shattering the model completely against the evidence and shatters only if required. Instead, ALBP extends box propagation to the relational setting. ALBP works by using only a subset of the model for box propagation, but with supernodes, as in Lifted BP. Shattering is performed only as needed for accommodating the parfactors considered at each step, thus minimizing the number of shatterings. Also, the propagation of the bounds can possibly allow for decision making even if the marginal has not converged.

Algorithm 1 presents the outer loop of the algorithm[1]. The outer loop explains the overall process of each iteration of the ALBP algorithm. Beginning with a single supernode that represents the query predicate with no constraints (root node $rn$), the lifted factor graph ($fg$) is further extended and broken on each iteration until the marginal probability bounds are satisfactory or $fg$ is completely shattered. In each iteration, a chosen node ($cn$) in $fg$ is *extended* by all rules that involve $cn$'s predicate; this mimics the unraveling process of theorem proving. Any evidence on a subset of the constraints of $cn$ also results in *breaking* $fg$ in order to separate the constraints with respect to the difference in observation. Since the query likely applies to a subset of $rn$'s constraints, we also break at $rn$ in the first iteration with respect to the query's constraints.

## 3 Initial Experiments

We have run our implementation of Anytime Lifted Belief Propagation on a variation of the popular smokers example with the following MLN [8]:

| | |
|---|---|
| 2.3 | $\neg cancer(x)$ |
| 1.4 | $\neg smokes(x)$ |
| 2.0 | $\neg friends(x, y)$ |
| −0.5 | $hospital(x)$ |
| 2.5 | $party(x)$ |
| 1.5 | $smokes(x) \rightarrow cancer(x)$ |
| 1.1 | $friends(x, y) \rightarrow (smokes(x) \leftrightarrow smokes(y))$ |
| 1.4 | $party(x) \wedge smokes(x)$ |
| 0.5 | $cancer(x) \rightarrow hospital(x)$ |
| −1.0 | $party(x) \rightarrow hospital(x)$ |

Our domain contained five persons, and we queried how likely each person had cancer given various observations. For each person, up to one randomly cho-

[1] An example and additional pseudocode can be found at http://tsi.wfubmc.edu/labs/strait/ALBP.pdf

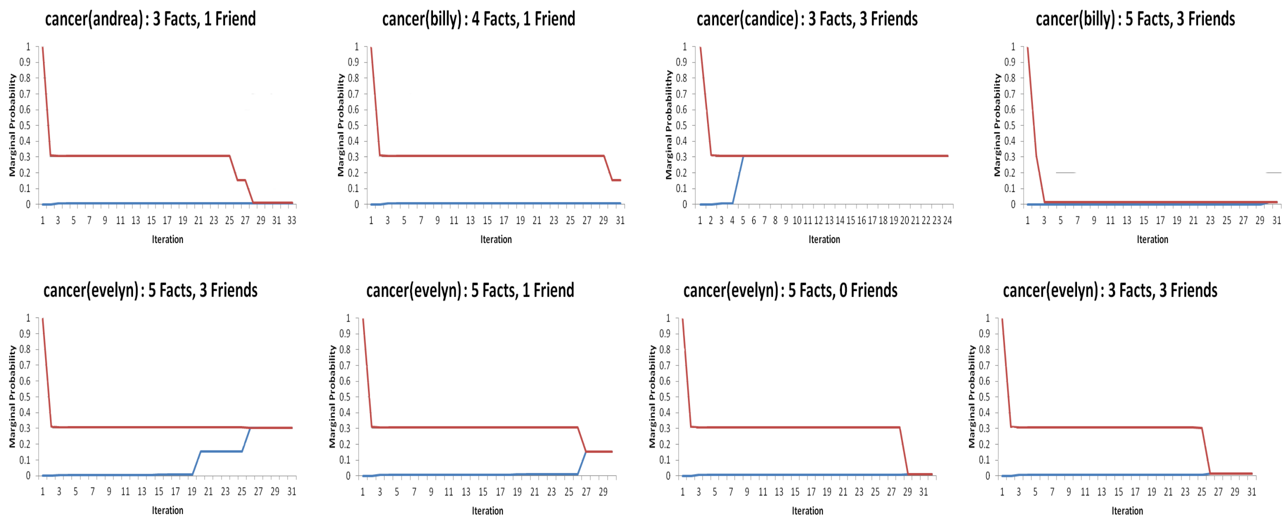


Figure 1: Plots of the marginal probability bounds during ALBP for various queries on the dataset. The red line represents the upper bound and the blue line represents the lower bound.

sen attribute was observed. Either zero, one or three friends relation were observed for each individual. The effects the varying observations had on the bounds of a single query (*cancer(X)*) are presented in Figure 1.

We observe three effects – (1) If there is immediate evidence about the query then ALBP converges quickly where the bound becomes zero rapidly (right two cases in the top figure). (2) If there is evidence close to the query but is not very strong, the bound becomes narrow quickly but convergence happens later. (3) Finally, if multiple evidences appear as the network is expanded, the bounds progressively decrease to zero. Currently, the choice of the next parfactor to be shattered is made at random. A better heuristic can lead to faster convergence of the bound in many cases. Our initial results are still promising as in most cases, the bounds become quite narrow before ALBP converges.

## 4 Conclusion

We presented the ALBP algorithm that performs lifted inference without shattering the entire model in advance. This allows for faster useful inference when bounds are enough (such as for decision-making). Importantly, it allows us to treat classes of random variables as a group even when they are not exactly symmetric; in other words, it allows us to reason in general about objects that are only *approximately* symmetric, where the notion of approximation becomes more restricted the more precise an answer is required. Our initial experiments are promising as the bounds become narrower before the algorithm converges.

An interesting extension in that direction would be making the algorithm consider the cost of obtaining evidence or sharper bounds on certain variables. Similarly, exploring the different methods of identifying the next parfactor to shatter would be an important future work. Finally, the algorithm should be evaluated on large real domains.

**Acknowledgements** The authors gratefully acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

## References

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

[2] D. Poole. First-Order Probabilistic Inference. In *IJCAI*, pages 985–991, 2003.

[3] R. de Salvo Braz, E. Amir, and D. Roth. Lifted First Order Probabilistic Inference. In *IJCAI*, 2005.

[4] J. M. Mooij and H. J. Kappen. Bounds on marginal probability distributions. In *NIPS*, 2008.

[5] R. de Salvo Braz, S. Natarajan, H. Bui, J. Shavlik, and S. Russell. Anytime lifted belief propagation. In *Statistical Relational Learning Workshop*, 2009.

[6] K. Kersting, Y. El Massaoudi, B. Ahmadi, and F. Hadiji. Informed lifting for message–passing. In *AAAI*, 2010.

[7] P. Singla and P. Domingos. Lifted first-order belief propagation. In *AAAI*, pages 1094–1099, 2008.

[8] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for AI*. Morgan & Claypool, 2009.

# Initial Empirical Evaluation of Anytime Lifted Belief Propagation - Supplement

Richard G. Freedman, Rodrigo de Salvo Braz, Hung Bui, Sriraam Natarajan

## 1 Additional Pseudocode

**Algorithm 1** ALBP Pseudocode for Extending

```
function PERFORMSPLIT(Supernode splitAt,
Rule r, FactorGraph fg)
    ▷ A new parfactor represents the rule
    newP ← createParfactor(r);
    addParfactor(fg, newP);
    ▷ Adjacent supernodes will be created
    for all Predicate p in r.predicates do
        newS ← createSupernode(p);
        ▷ If it already exists, then replace new one
        if fg.supernodes contains newS then
            newS ← retrieve(fg.supernodes, newS);
        else
            addSupernode(newS);
        end if
        setAdjacent(newS, newP);
        appendToBack(newS, newSList);
    end for
    ▷ Synchronize adjacent supernode constraints
    unifyArguments(newSList, r, splitAt);
    ▷ Prior shattering may duplicate groundings
    for all Supernode s_n in newSList do
        for all Supernode s_f in fg.supernodes do
            if (s_f.constraints ⊂ s_n.constraints)
              and (s_f ≠ s_n) then
                ▷ Match through breaking
                performShatter(s_n, s_f, fg);
            end if
        end for
    end for
end function
```

## 2 ALBP Example

Consider the following knowledge base:

```
pf1 : φ1(hasGoodOffer(Person),
offer(Job,Person),goodFor(Person,Job))
pf2 : φ2(goodFor(Person,Job),cityPerson(Person),
inCity(Job))
pf3 : φ3(goodFor(Person,Job),goodEmployer(Job))
...
0.9:  offer(mary,Job), Job in {a,b,c}.
1.0:  not offer(mary,Job), Job not in {a,b,c}.
0.8:  goodEmployer(Job), Job in {a,c}.
1.0:  inCity(c).
...
```

$\phi_i$ is the potential of $i^{th}$ parfactor (denoted as $pf_i$). The higher the potential, the higher is the probability of the ground instance being true. Let the goal be to predict if *Mary* has a good job offer. Figure 1 outlines how the Anytime Lifted BP algorithm proceeds to answer this query. First, the algorithm begins with the query variable *hasGoodOffer(Mary)*. Now, it considers the parfactor *hasGoodOffer(Person), offer(Job,Person), goodFor(Person,Job)*. Just considering the blanket factor from this node (since there is no evidence associated with these nodes), reduces the bounds to lie between $[0.1, 1.0]$. Note that the bound already decreases without having to perform any shattering. In $(iii)$, it can be observed that the factor graph is now shattered into two distinct regions corresponding to the instantiations of $\langle P = Mary\rangle$ and $\langle P \neq Mary\rangle$. Since, we are not currently interested in the non-Mary case, we will ignore the bottom half of $(iii)$. Note that this will not affect the bound on the query since this is disconnected from the model that predicts about *Mary*. This is very similar to Prolog queries with unbound variables that get bounded to possibly many values. Also, the network is further shattered based on whether or not the job $J$ is in $\{a, b, c\}$.

The top portion of part $(iii)$ is presented in part $(iv)$. As can be observed, currently the network has 2 distinct components based on the value for the job $J$. In $(v)$, the algorithm propagates further and requests for bounds from the parfactor $pf_3$ that corresponds to *(goodFor(Person,Job),goodEmployer(Job))* when instantiated with $Mary$ for $P$ and with $\langle a, b, c\rangle$ for $J$. Note that this corresponds to shattering the parfactor $pf_3$ based on the instantiations of Job. Finally in $(vi)$, given the parfactor $0.8$ : *goodEmployer(Job), Job in* $\{a,c\}$, the bounds decreases to between $[0.82, 1.0]$. If the bound is satisfactory the algorithm terminates. Else, it proceeds to consider the other parfactors and shatters them based on the constraints and the other instantiations. It should be noted that the bound has been shrunk to a width of just $0.18$ with the network being shattered just twice. If the model proceeds with shattering further, the bounds will be further reduced. It is worth noting that in (vi), considering the parfactor $pf_3$ splits $b$away from the group of jobs. More importantly, $a$ and $c$ are grouped together even though they are not indistinguishable given the entire model. All the previous lifted inference algorithms would have separated them. This clearly shows that our algorithm avoids shattering unless it is absolutely necessary.

The key thing to note is that the shattering takes place in a lazy manner in that the

**Algorithm 2** ALBP Pseudocode for Breaking

```
▷ Shattering on queries and supernodes is similar
function performShatter(Supernode
shatterAt, Evidence e, FactorGraph fg, Set <Node> shatteredNodes = {},
Map <Parfactor, Parfactor> shatteredParfactors = {})
    ▷ Shattering duplicates the supernode
    s1 ← createSupernode(lift(shatterAt.predicate));
    s2 ← createSupernode(lift(shatterAt.predicate));
    ▷ The constraints complement each other
    s1.constraints ←
      shatterAt.groundings ∩ e.constraints;
    s2.constraints ←
      shatterAt.groundings − e.constraints;
    ▷ If already exists, then replace new one
    if fg.supernodes contains s1 then
        s1 ← retrieve(fg.supernodes, s1);
    else
        addSupernode(fg, s1);
    end if
    if fg.supernodes contains s2 then
        s2 ← retrieve(fg.supernodes, s2);
    else
        addSupernode(fg, s2);
    end if
    putInSet(shatteredNodes, shatterAt);
    ▷ Shatter the parfactors adjacent to shatterAt
    for all Parfactor p in
      shatterAt.adjacentParfactors do
        if shatteredParfactors.keys contains p
          then
            newP ← retrieveMapOf(
              shatteredParfactors, p);
        else
            if shatteredParfactors.elements
              contains p then
                newP ← retrieveKeyOf(
                  shatteredParfactors, p);
            else
                newP ← hardCopyParfactor(p);
                addParfactor(fg, newP);
                putInMap(shatteredParfactors, p,
                  newP);
            end if
        end if
        ▷ Recursively shatter adjacent supernodes
        if not shatteredNodes contains p then
            performShatter(p, e, fg, shatteredNodes,
              shatteredParfactors);
        end if
        ▷ Swap adjacencies to isolate shatterAt
        replaceAdjacent(shatterAt, s1, p);
        replaceAdjacent(shatterAt, s2, newP);
    end for
    removeSupernode(fg, shatterAt);
end function
```

**Algorithm 3** ALBP Pseudocode for Breaking

```
function performShatter(Parfactor
shatterRelay, Evidence e, FactorGraph fg,
Set<Node> shatteredNodes, Map
<Parfactor, Parfactor> shatteredParfactors)
    putInSet(shatteredNodes, shatterRelay);
    for all Supernode s in
      shatterRelay.adjacentSupernodes do
        if not shatteredNodes contains s then
            performShatter(s, e, fg, shatteredNodes,
               shatteredParfactors);
        end if
    end for
end function
```

model is shattered as and when it is needed. In this model, we first considered the instantiation of *Person* and then that of *Job*. The messages are passed back to the query node from the node that has been shattered. The messages **always** decrease the bound. When the bound has been decreased to the desired level, the algorithm can be allowed to terminate. Note that in many cases, we need a rough estimate of probability and not the exact value. This is especially true while decision-making as we might be interested in knowing if the probability of one event is significantly higher than the other and not on the exact values of the probabilities.

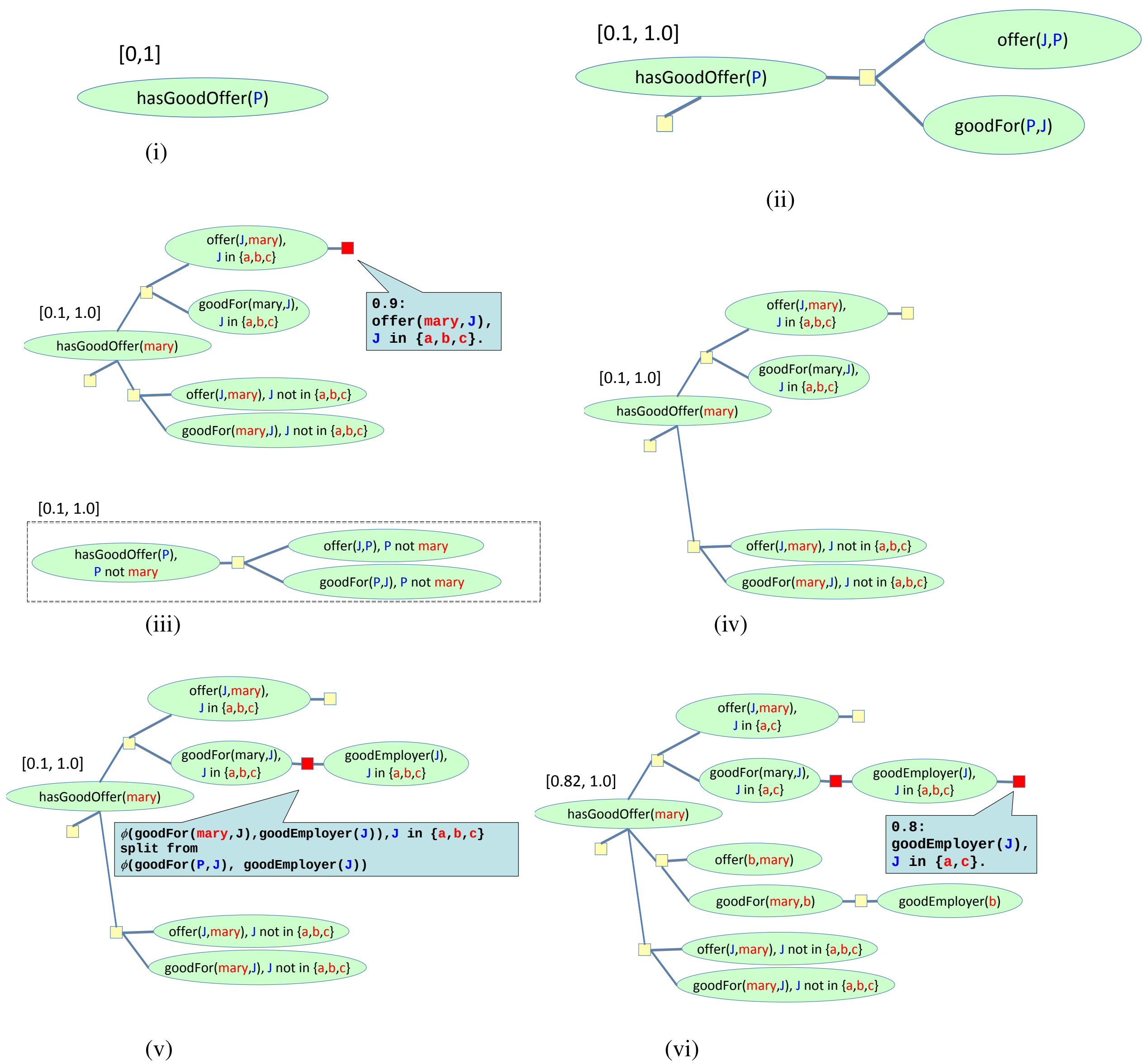


Figure 1: An illustrative example of Anytime Lifted Belief Propagation.