A Deeper Empirical Analysis of CBP Algorithm: Grounding Is the Bottleneck

Shrutika Poyrekar and Sriraam Natarajan and Kristian Kersting*
School of Computing and Informatics, Indiana University, USA
* Department of Computer Science, Technische Universitat Dortmund, Germany

Abstract

In this work-in-progress, we consider a lifted inference algorithm and analyze its scaling properties. We compare two versions of this algorithm – the original implementation and a newer implementation built on a database. Our preliminary results show that constructing the factor graph from the relational model rather than the construction of the compressed model is the key bottleneck for the application of lifted inference in large domains.

1 Counting Belief Propagation

Recent years have seen a surge of interest in lifted probabilistic inference algorithms that exploit redundancies to speed up inference, ultimately avoiding explicit state enumeration by manipulating first-order state representations directly. Lifted inference exploits the observation that ground models obtained by instantiating a set of logical models exhibit a large degree of symmetry, which leads to repeating the same set of computations multiple times. Accordingly, lifted algorithms operate on predicate level so that the inference is performed at a higher level instead of at the instance level. There have been several different algorithms proposed recently, ranging from exact algorithms (Choi, de Salvo Braz, and Bui 2011; de Salvo Braz, Amir, and Roth 2005; den Broeck et al. 2011; Milch et al. 2008; Poole 2003; Sen, Deshpande, and Getoor 2008), deterministic approximation of exact algorithms (Gogate and Domingos 2013; Ahmadi, Kersting, and Sanner 2011; Ahmadi, Kersting, and Hadiji 2010; Kersting, Ahmadi, and Natarajan 2009; Nath and Domingos 2010; Singla and Domingos 2008), sampling algorithms (Milch and Russell 2006; Poon, Domingos, and Sumner 2008; Zettlemoyer, Pasula, and Kaelbling 2008), search-based algorithms (Gogate and Domingos 2011; Van den Broeck et al. 2011) and pre-processing (Shavlik and Natarajan 2009).

We had earlier developed *Counting Belief Propagation* based on message passing that is illustrated in Figure 1. The left-most graph in this figure shows a factor-graph representation of a graphical model. A factor graph is a bi-partite graph with random-variable nodes and factor nodes. For instance, in the first graph of the Figure 1, the circles with "A", "B", and "C" in them represent random variables, and the squares represent the factors, which are essentially functions that encode the probabilistic relationships between two random variables. The belief-propagation (BP) algorithm works by passing real-valued functions called "messages" along the edges between the nodes. For the factor graph in the figure, let us assume that the nodes A, B, and C all send the same message to factors f_1 and f_2, which send the message back to the nodes. As can be seen after first iteration, nodes A and C send the same message back while B sends two copies of the same message. This allows us to cluster the nodes A and C to a single node and the factors to a single factor. Hence, message passing can now take place on a smaller network. This is a simple example of a lifted inference algorithm from a message passing perspective.

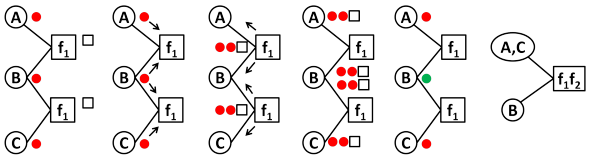

Figure 1: Counting Belief Propagation (CBP). From left to right, the steps of compressing a simple factor graph assuming no evidence. The shaded/colored small circles and squares denote the groups and signatures produced running CBP. On the right-hand side, the resulting compressed factor graph is shown.

This algorithm has now been widely applied to several tasks - from model counting and temporal inference (Kersting, Ahmadi, and Natarajan 2009) to computing personalized page ranks and Kalman filters (Ahmadi, Kersting, and Sanner 2011) to clustering on demand (Neumann, Ahmadi, and Kersting 2011) to solving linear programs (Mladenov, Ahmadi, and Kersting 2012). While this is successful in many real tasks, some important questions remain in understanding the scalability of the algorithm. Specifically, there is prevalent criticism that compression of the ground factor graph can be a bottleneck for this algorithm. In this paper, we explore this hypothesis in detail and investigate the relationship between construction of a ground factor graph and the compression of this factor graph in several domains. We show empirically that if there exists a robust, fast grounding technique, this is in fact a very practical algorithm. This

also opens up the potential research direction of constructing a fast (and possibly approximate) grounding algorithm. The results in this work are preliminary and help identify important future research directions.

We next present the experimental domains and the results of the evaluation in these domains. We then conclude by outlining some future research directions.

2 Empirical Analysis

To empirically analyze the CBP algorithm, we used two implementations: first is the original implementation of Kersting et al. (Kersting, Ahmadi, and Natarajan 2009) in Python. We focus here on the naive $\mathcal{O}(n^2)$ version of color-passing using flooding. Doing so is a sensible idea since flooding is easy to parallelize. There indeed are quasi-linear $\mathcal{O}((m+n)\log n)$ algorithms for color passing on graphs with n vertices and m edges due to asynchronous color updates, see e.g. (Berkholz, Bonsma, and Grohe 2013; Grohe et al. 2013) and references in there for more details, however, asynchronous updates are far more difficult to parallelize and the overhead may actually spoil the benefits of distributed computations. The second implementation is a java-based algorithm that uses Tuffy (Niu et al. 2012), a database implementation of Markov Logic networks that employs PostgreSQL underneath. In this implementation, we used Tuffy for grounding the MLN to a factor graph and implemented the compression step using the data structures (tables) provided by Tuffy. The aim was to answer the following two questions:

1. Does the use of a database system improve the performance of CBP? Does it scale well compared to the original implementation?
2. Which is the bottleneck? Construction of the factor graph or its compression?

To answer these two questions, we considered four standard MLNs that are used widely by the StaRAI community. They are (i) Smokers (ii) Cora entity resolution (iii) UW-CSE and (iv) WebKB. These MLNs were obtained from the alchemy website [1]. We kept the evidence to about 10% of the total groundings (this evidence is significantly high for many lifted inference algorithms to handle). We increased the number of objects in the domain to better understand the scaling properties of the algorithms. The reported results are averaged over 5 runs. It must be mentioned that the number of MLN clauses in Smokers and WebKB domain are significantly smaller than the number of clauses in the other two domains. Thus the algorithms scale better in these two domains.

The results of the (initial) empirical analysis are presented in Figure 2. Both the methods converged to the exact same marginals for the queries indicating that there is no difference in the quality of the results. The second row of the results are the results in the 4 domains of the original CBP implementation (we will call this as CBPP to denote CBP in Python) and the first row is the Tuffy based implementation (we will call this as CBPT) in Java. It can be observed in **all** the domains of both the implementations that grounding of the MLN to a factor graph is the key bottleneck compared to that of the compression. This suggests that there is a necessity in improving the grounding of the network for efficient inference. Also, it can be observed that in some domains, CBPP was not able to compress the factor graph as it ran out of memory. Note that in all domains, the number of ground atoms is squared of the number of objects. Hence, in Cora domain for example, CBPP could not compress with 1M ground atoms while CBPT was able to handle 1B ground atoms. In WebKB, both methods were able to handle 1B ground atoms as the number of clauses was significantly small.

In both domains, CBPP is faster than CBPT due to its faster implementation in C++ and Python as the list operations inside Python are faster. But as mentioned earlier, the Tuffy based implementation is able to scale much better than CBPP (1B vs 10k in Cora and 100M vs 100 in UWCSE). This demonstrates that the use of a database underneath helps in scaling up the lifted inference algorithm by order of magnitude.

To summarize and answer the two questions (1) CBPT scales significantly better than CBPP algorithm although the latter is faster due to its implementation in Python and (2) Construction of the ground factor graph is the significant bottleneck in all the domains for both the algorithms.

3 Discussion

As far as we are aware, this is the first preliminary evaluation of the scalability of a lifted inference algorithm. Our goal was to evaluate whether the algorithms can handle a scale of millions of ground atoms and evidence that can be in millions of ground atoms. It is clear that with the use of underlying data base technology, BP algorithm can scale well.

The results while initially promising also raise several questions and challenges that need to be addressed before the algorithm can be applied on larger data sets:) (1) the full grounding must be avoided to form the factor graph. Approaches such as FROG (Shavlik and Natarajan 2009) and Tuffy's default grounding require knowledge of the query to avoid fully grounding the network. (2) It would be an interesting research direction to determine how to incrementally ground the data for compression of the factor graph. (3) There is a necessity to parallelize the grounding operation to scale up inference. There has been prior work on using MapReduce for the compression step (Ahmadi et al. 2013) but as we show here, the key bottleneck is still the grounding process which needs to be sped up. (4) Finally, combining incremental grounding with compression in the same step would be an interesting direction for future research.

4 Acknowledgement

We like to acknowledge the work performed under LongView International Inc. AFRL contract FA8750-14-C-0022 under AFOSR STTR Topic AF13-AT11. The opinions and conclusions do not reect the position of the Air Force, AFOSR or AFRL. The authors also gratefully acknowledge

[1] http://alchemy.cs.washington.edu/mlns/

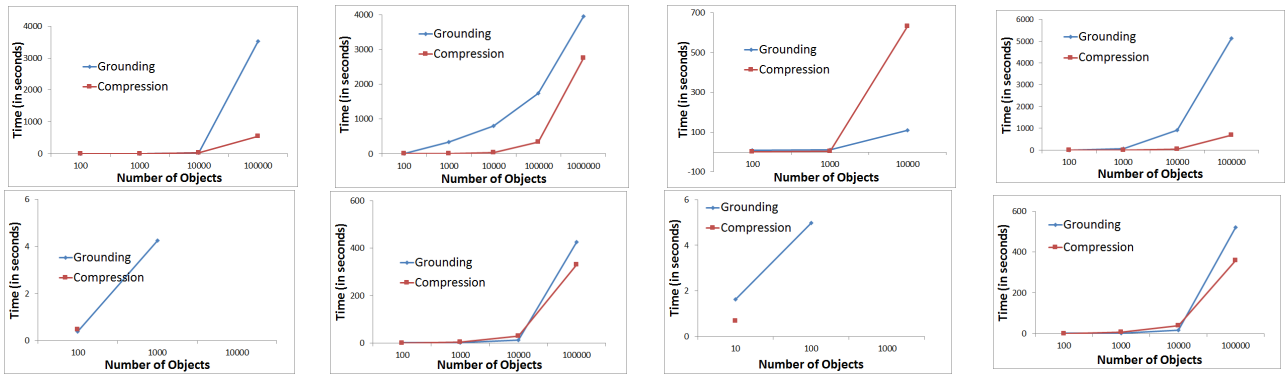

Fig. 2: Results of our evaluation. The top row presents the results of the CBPT algorithm while the bottom row presents the results of CBPP algorithm. Columnwise, the first column is the Cora data set, the second is the Smokers data set, third is UW-CSE and the last column is the WebKB data set.

discussions with the AFRL COTR James Nagy for his constructive feedback and support.

References

Ahmadi, B.; Kersting, K.; Mladenov, M.; and Natarajan, S. 2013. Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training. *Machine Learning*.

Ahmadi, B.; Kersting, K.; and Hadiji, F. 2010. Lifted belief propagation: Pairwise marginals and beyond. In P. Myllymaeki, T. Roos, T. J., ed., *Proceedings of the 5th European Workshop on Probabilistic Graphical Models (PGM–10)*.

Ahmadi, B.; Kersting, K.; and Sanner, S. 2011. Multi-evidence lifted message passing with application to pagerank and the kalman filter. In *IJCAI*.

Berkholz, C.; Bonsma, P.; and Grohe, M. 2013. Tight lower and upper bounds for the complexity of canonical colour refinement. In *21st Annual European Symposium on Algorithms (ESA)*, 145–156.

Choi, J.; de Salvo Braz, R.; and Bui, H. 2011. Efficient methods for lifted inference with aggregate factors. In *AAAI 2011*.

de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted First Order Probabilistic Inference. In *IJCAI*, 1319–1325.

den Broeck, G. V.; Taghipour, N.; Meert, W.; Davis, J.; and Raedt, L. D. 2011. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, 2178–2185.

Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *Proc. 27th Conf. Uncertainty in AI*.

Gogate, V., and Domingos, P. 2013. Structured message passing. In *UAI13*.

Grohe, M.; Kersting, K.; Mladenov, M.; and Selman, E. 2013. Dimension reduction via colour refinement. In *arXiv:1307.5697*.

Kersting, K.; Ahmadi, B.; and Natarajan, S. 2009. Counting Belief Propagation. In *UAI*.

Milch, B., and Russell, S. 2006. General-purpose mcmc inference over relational structures. In *Uncertainty in Artificial Intelligence*, 349–358. AUAI Press.

Milch, B.; Zettlemoyer, L.; Kersting, K.; Haimes, M.; and Pack Kaelbling, L. 2008. Lifted Probabilistic Inference with Counting Formulas. In *AAAI*.

Mladenov, M.; Ahmadi, B.; and Kersting, K. 2012. Lifted linear programming. In *AISTATS*, 788–797.

Nath, A., and Domingos, P. 2010. Efficient lifting for online probabilistic inference. In *AAAI*.

Neumann, M.; Ahmadi, B.; and Kersting, K. 2011. Markov logic sets: Towards lifted information retrieval using pagerank and label propagation. In *AAAI*.

Niu, F.; Zhang, C.; Re, C.; and Shavlik, J. 2012. Scaling inference for markov logic via dual decomposition. In *ICDM*, 1032–1037.

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

Poon, H.; Domingos, P.; and Sumner, M. 2008. A general method for reducing the complexity of relational inference and its application to mcmc. In *AAAI*, 1075–1080.

Sen, P.; Deshpande, A.; and Getoor, L. 2008. Exploiting shared correlations in probabilistic databases. *Proc. VLDB Endow.* 1:809–820.

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 *IJCAI*.

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

Van den Broeck, G.; Taghipour, N.; Meert, W.; Davis, J.; and De Raedt, L. 2011. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*.

Zettlemoyer, L. S.; Pasula, H. M.; and Kaelbling, L. P. 2008. Logical particle filtering. In *Probabilistic, Logical and Relational Learning - A Further Synthesis*, number 07161 in Dagstuhl Seminar Proceedings.