ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-1011

Conditioned Belief Propagation Revisited

Thomas Geier and **Felix Richter** and **Susanne Biundo**[1]

Abstract. Belief Propagation (BP) applied to cyclic problems is a well known approximate inference scheme for probabilistic graphical models. To improve the accuracy of BP, a divide-and-conquer approach termed Conditioned Belief Propagation (CBP) has been proposed in the literature. It recursively splits a problem by conditioning on variables, applies BP to subproblems, and merges the results to produce an answer to the original problem. In this essay, we propose a reformulated version of CBP that exhibits anytime behavior, and allows for more specific tuning by formalizing a further decision point that decides which subproblem is to be decomposed next. We propose some simple and easy to compute heuristics, and demonstrate their performance using an empirical evaluation on randomly generated problems.

1 Introduction

Belief Propagation (BP) [5] works by sending messages between variables along the edges of the problem graph. For acyclic problems the algorithm terminates after a number of message-passing steps that is linear in the size of the graph, and in these cases BP computes exact results for the marginal probabilities of all variables at once. But when the graph contains cycles, BP is no longer guaranteed to converge, and the results produced are no longer exact. With the knowledge that BP optimizes a certain variational approximation [6], it is also possible to extract an estimate of the partition function from the messages.

In this short essay[2] we describe a simple method of improving the approximation quality of BP. The basic idea was already formulated by Pearl [5], who proposed to condition on variables to break loops. But instead of aiming to break all loops, we apply BP again to a now slightly less cyclic problem. This very idea was picked up by Eaton and Ghahramani [2], who introduced the term Conditioned Belief Propagation (CBP) for it. They describe a very elaborate method of picking variables to condition on. In this paper, we formalize a simpler version of CBP with an additional choice point we call "leaf selection", together with simpler, well performing heuristics.

Related to CBP are collapsed sampling methods, which sample assignments to a subset of variables while solving the conditioned problem exactly (see for example Cycle-Cutset sampling [1]). In contrast to this, CBP systematically explores conditions, while solving the remaining problem approximately. Further, CBP can be regarded as a mixture model. See for example Jaakkola and Jordan's work about using mixtures of Mean field approximations for probabilistic inference [4]. The difference to this approach is mainly that CBP "ties" the components to certain conditions, and the mixture components in CBP are mutually exclusive.

[1] Institute of Artificial Intelligence, Ulm University, Germany, email: forename.surname@uni-ulm.de

[2] There exists an extended version of this essay [3].

2 CBP

CBP is an inference algorithm for Markov networks that yields an approximation to the partition function. With slight modifications it can also produce marginal probabilities for all variables. We denote sets of variables using bold letters ($\mathbf{X}, \mathbf{Y}, \dots$), and single variables using normal style ($X, Y, \dots$). The set of all variables is $\mathcal{X}$. For a set of variables $\mathbf{X}$, let $\mathrm{V}(\mathbf{X})$ be the set of all its *assignments*, which are functions mapping a variable $X \in \mathbf{X}$ to one of its finitely many values $\mathrm{D}(X)$, and let $\widetilde{\mathrm{V}}(\mathbf{X})$ be the set of partial assignments to $\mathbf{X}$. We denote (partial) assignments with lower case (Greek) letters. The set of all partial assignments is $\mathcal{A}$. A *factor* $\phi : \mathrm{V}(\mathbf{X}_\phi) \rightarrow \mathbb{R}^+$ over a finite set of variables $\mathbf{X}_\phi$ maps an assignment to its variables to the non-negative reals. A *Markov network* (or *problem*) Φ over variables $\mathbf{X}_\Phi$ is the product over a finite number of factors: $\Phi(\mathbf{x}) = \prod_{\phi \in \Phi} \phi(\mathbf{x}_\phi)$, mapping assignments $\mathbf{x} \in \mathrm{V}(\mathbf{X}_\Phi)$ to the non-negative reals. The set of all Markov networks is $\mathcal{P}$. Let Z_Φ be the *partition function* (or normalizing constant) of a Markov network Φ, defined by $Z_\Phi = \sum_{\mathbf{x}} \Phi(\mathbf{x})$.

The CBP algorithm decomposes a given problem Φ step by step. This forms a tree of partial assignments with the empty assignment as the root. For each inner node $\xi \in \widetilde{\mathrm{V}}(\mathbf{X}_\Phi)$, its children are the assignments obtained by extending ξ by all assignments to some variable X. Because at each stage of the algorithm only the leaf nodes of this tree are relevant, we capture the state of the computation by a set of partial assignments $Q \subseteq \widetilde{\mathrm{V}}(\mathbf{X}_\Phi)$. Q always implies a partition of all assignments $\mathrm{V}(\mathbf{X}_\Phi)$. The function $\mathtt{ref}_{L,V} : \mathcal{P} \times 2^{\mathcal{A}} \rightarrow 2^{\mathcal{A}}$ applies one refinement step to a set of leaves Q, using leaf selection heuristic $L : \mathcal{P} \times 2^{\mathcal{A}} \rightarrow \mathcal{A}$ and variable selection heuristic $V : \mathcal{P} \times \mathcal{A} \rightarrow \mathcal{X}$. Letting $\xi = L(\Phi, Q)$, and $X = V(\Phi, \xi)$,

$$\mathtt{ref}_{L,V}(\Phi, Q) = (Q \setminus \{\xi\}) \cup \{\{\xi \cup \{X \mapsto x_i\}\} \mid x_i \in \mathrm{D}(X)\}. \tag{1}$$

To obtain an estimate of the partition function, we sum over the estimates $Z^{BP}_{\Phi[\xi]}$ obtained from applying BP to the problem Φ conditioned on partial assignment ξ. We define the function $\mathtt{sum} : \mathcal{P} \times 2^{\mathcal{A}} \rightarrow \mathbb{R}^+$ as $\mathtt{sum}(\Phi, Q) = \sum_{\xi \in Q} Z^{\mathrm{BP}}_{\Phi[\xi]}$. Then we can define the estimate of the partition function obtained from n steps of $\mathtt{CBP}_{L,V} : \mathcal{P} \times \mathbb{N}^+ \rightarrow \mathbb{R}^+$ as

$$\mathtt{CBP}_{L,V}(\Phi, n) = \mathtt{sum}(\mathtt{ref}^n_{L,V}(\Phi, \{\emptyset\})). \tag{2}$$

Here, $\mathtt{ref}^n_{L,V}$ means the n-fold recursive application of $\mathtt{ref}_{L,V}$ in its second argument: $\mathtt{ref}^n(\Phi, Q) = \mathtt{ref}(\Phi, \mathtt{ref}^{n-1}(\Phi, Q))$ and $\mathtt{ref}^0(\Phi, Q) = \mathtt{ref}(\Phi, Q)$. Turning this function into an imperative algorithm reveals the anytime nature of CBP.

Since BP yields exact results on tree-structured problems, one can stop the decomposition of a leaf once it contains no loops, or use other exact methods to solve the leaf earlier. But anyway CBP converges to the exact solution, since it becomes equivalent to summing over all assignments once all variables are conditioned in all leaves.

Note that this formalization is agnostic to the used inference algorithm, and every other way of calculating an approximate partition function can be used.

3 Heuristics

We discuss two simple leaf selection heuristics. The first is minimum depth (MIN DEPTH) which grows the tree in a breadth-first like manner, and resembles CBP as described in [2]. The second is maximum weight (MAX Z) which expands the leaf with the highest estimated partition function. The reasoning behind MAX Z is to focus effort on the most important regions of the probability space, and to find leaves early where BP overestimates the true partition function. On the downside it fails to expand leaves that are undervalued.

As discussed, the reasons for BP yielding wrong results are convergence to local optimum (or total failure to converge), and error contributions from cycles. Since cycles are also the cause of bad convergence, it is reasonable to aim at using conditioning to break these influential cycles. We propose a variable selection heuristic called *Time To Converge* (TTC). It selects a variable that is adjacent to the edge for which the maximum of convergence time over the messages in both directions is the highest. The idea is to identify parts of the problem where BP has trouble converging. For a more detailed discussion, and more heuristics see the extended version [3].

4 Evaluation

We evaluated CBP with the proposed heuristics on randomly generated problems. An excerpt of the results is given in Figure 1. Concerning the leaf selection heuristics, we found that MAX Z dominates MIN DEPTH in all cases. We assume that MAX Z can still be improved by incorporating some estimate of the approximation error. Looking at the variable selection heuristics, we found that TTC performs very well — in particular on grid-structured problems. Its performance appears to be similar to the BBP heuristic from [2] on grid problems, and it exceeds the performance of BBP on randomly structured problems. This is the case despite TTC being much simpler and faster to compute than BBP. For problems with random structure, graph-oriented variable selection heuristics like MAX DEGREE perform well (not shown). We conclude that it is important to choose the heuristic with regard to the characteristics of the problem, and simple heuristics can practically perform well.

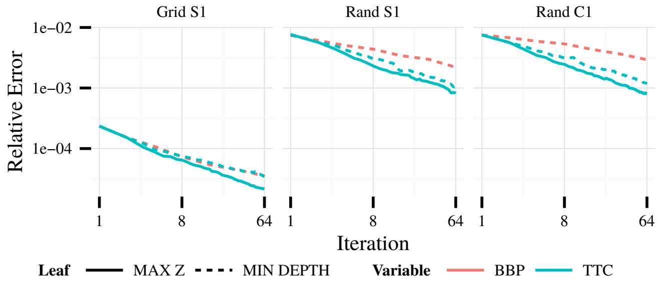

Figure 1. These plots show the median of the relative error over 500 random problems (grids are 8×8, and random problems have 25 variables and 50 factors; factor values S1 are drawn from exponentiated normal distribution $\exp(\mathcal{N})$, C1 factors are single features). Color encodes the variable selection heuristic, while line type represents leaf selection heuristic. The variable selection heuristic BBP shows results obtained from the heuristic in [2], whose implementation is fixed to MIN DEPTH. On the x-axis we print the number of iterations of CBP; iteration 1 corresponds to ordinary BP. The y-axis shows the relative error in $\log Z$.

5 Discussion

CBP offers a simple means to improve the accuracy of BP. Our formulation can be cast as an anytime algorithm, and allows to trade in time and space for improved accuracy. Since CBP solves partially conditioned problems, it is also able to reveal and exploit context-specific independence. Further, it can exploit deterministic dependencies when those become inconsistent with the current condition. Then it is possible to evaluate the current leaf to zero. In this way CBP is an algorithm that has facilities to solve both high entropy parts of problems (BP), as well as low entropy parts (conditioning). This is a perfect combination, as BP is weak on low entropy problems (i.e. problems with very strong dependencies), and conditioning fails under the presence of many equal choices.

Despite the apparent benefits of CBP, we would also like to point out a major short-coming that has to be solved before CBP can be used as a true general-purpose inference algorithm. If one takes a close look at the plots in Figure 1, one will notice that the accuracy improves only with the logarithm of the number of iterations. This is intuitive, since with the progression of CBP the error contribution of each leaf decreases with its weight, and thus each further decomposition step has a lesser effect. In addition, the relative improvement per step will be much smaller for problems with more variables, as the absolute improvement that can be gained by conditioning on one variable stays the same. This means that the computational cost of CBP required to achieve the same relative improvement grows exponentially with the problem size. This is clearly impractical. Thus, to make CBP a viable choice, we have to develop a way to exploit the independence between the conditioning effects of variables that are largely unrelated to each other. In summary, CBP appears to offer many desirable features, but it falls short of becoming practically interesting unless a way is found to make it scale to large problem instances.

Acknowledgements

This work is done within the Transregional Collaborative Research Centre SFB/ TRR 62 “Companion-Technology for Cognitive Technical Systems” funded by the German Research Foundation (DFG).

REFERENCES

[1] Bozhena Bidyuk and Rina Dechter, ‘Cycle-cutset sampling for Bayesian networks’, in *Advances in Artificial Intelligence*, 297–312, Springer, (2003).

[2] Frederik Eaton and Zoubin Ghahramani, ‘Choosing a variable to clamp: Approximate inference using conditioned belief propagation’, in *Proceedings of the 12th Int. Conf. on AI and Statistics*, volume 5, pp. 145–152, (2009).

[3] Thomas Geier, Felix Richter, and Susanne Biundo, ‘Conditioned Belief Propagation Revisited: Extended Version’, Technical Report UIB 2014-03, Ulm University, (2014).

[4] Tommi Jaakkola and Michael Jordan, ‘Improving the mean field approximation via the use of mixture distributions’, in *Learning in graphical models*, 163–173, Springer, (1998).

[5] Judea Pearl, ‘Fusion, propagation, and structuring in belief networks’, *Artificial intelligence*, **29**(3), 241–288, (1986).

[6] Jonathan Yedidia, William Freeman, and Yair Weiss, ‘Bethe free energy, Kikuchi approximations, and belief propagation algorithms’, *Advances in neural information processing systems*, **13**, (2001).