7

Inference in Bayesian Networks: A Historical Perspective

ADNAN DARWICHE

1 Introduction

Judea Pearl introduced Bayesian networks as a representational device in the early 1980s, allowing one to systematically and locally assemble probabilistic beliefs into a coherent whole. While some of these beliefs could be read off directly from the Bayesian network, many were implied by this representation and required computational work to be made explicit. Computing and explicating such beliefs has been the subject of much research and became known as the problem of *inference* in Bayesian networks. This problem is critical to the practical utility of Bayesian networks as the computed beliefs form the basis of decision making, which typically dictates the need for Bayesian networks in the first place.

Over the last few decades, the interest in inference algorithms for Bayesian networks remained great and has witnessed a number of shifts in emphasis with regards to the adopted computational paradigms and the type of queries addressed. My goal in this paper is to provide a historical perspective on this line of work and the associated shifts, where we shall see the key role that Judea Pearl has played in initiating and inspiring many of the technical developments that have formed and continue to form the basis of work in this area.

2 Starting with trees

It all began with trees — and polytrees! These are network structures that permit only one undirected path between any two nodes in the network; see Figure 1. If each node has at most one parent, we have a tree. Otherwise, we have a polytree. Pearl's first inference algorithm — and the very first algorithm for Bayesian networks — was restricted to trees [Pearl 1982] and was immediately followed by a generalization that became known as the *polytree algorithm* [Kim and Pearl 1983; Pearl 1986b]. The goal here was to compute a probability distribution for each node in the network given some evidence, a task which is known as computing node marginals.

The polytree algorithm was based on a message-passing computational paradigm, where nodes in the network send messages to a neighboring node after they have received messages from all other neighbors. Each message can be viewed as summarizing results from one part of the network and passing them on to the rest of the network. Messages that communicated information from parents to their children were said to quantify the causal

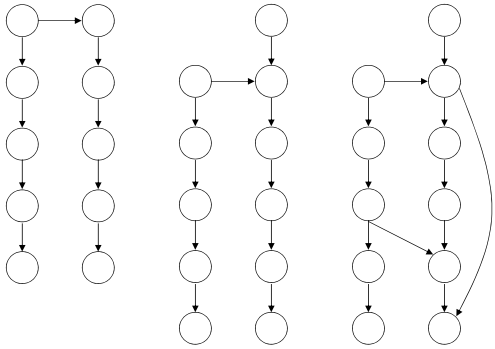

Figure 1. From left to right: a tree, a polytree, and a multiply–connected network.

support from parents to these children. On the other hand, messages that communicated information from children to their parents were said to quantify the diagnostic support from children to parents. The notions of causal and diagnostic supports were rooted in the causal interpretation of Bayesian network structures that Pearl insisted on, where parents are viewed as direct causes of their children. According to this interpretation, the distribution associated with a node in the Bayesian network is called the *belief* in that node, and is a function of the causal support it receives from its direct causes, the diagnostic support it receives from its direct effects, and the local information available about that node. This is why the algorithm is also known as the *belief propagation* algorithm, a name which is more common today.

The polytree algorithm has had considerable impact and is of major historical significance for a number of reasons. First, it was the very first *exact* inference algorithm for this class of Bayesian networks. Second, its time and space complexity were quite modest being linear in the size of the network. Third, the algorithm formed the basis for a number of other algorithms, both exact and approximate, that will be discussed later. In addition, the algorithm provided a first example of reading off independence information from a network structure, and then using it to decompose a complex computation into smaller and independent computations. It formally showed the importance of independence, as portrayed by a network structure, in driving computation and in reducing the complexity of inference.

One should also note that, according to Pearl, this algorithm was motivated by the work of [Rumelhart 1976] on reading comprehension, which provided compelling evidence that text comprehension must be a distributed process that combines both top-down and bottom-up inferences. This dual mode of inference, so characteristic of Bayesian analysis, did not match the capabilities of the ruling paradigms for uncertainty management in the 1970s. This led Pearl to develop the polytree algorithm [Pearl 1986b], which, as mentioned earlier, appeared first in [Pearl 1982] with a restriction to trees, and then in [Kim and Pearl 1983] for polytrees.

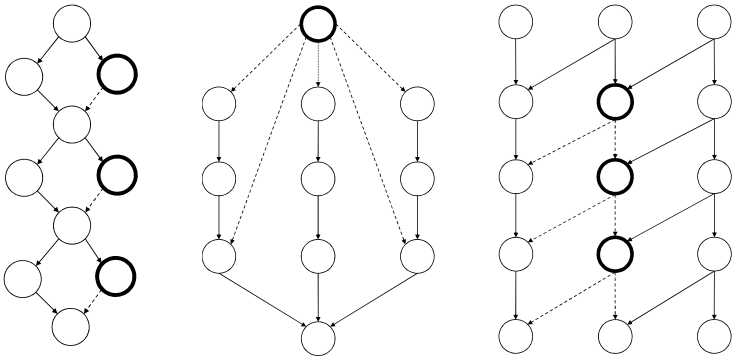

Figure 2. Networks and corresponding loop-cutsets (bold circles).

3 On to more general structures

Soon after the polytree algorithm was introduced, the search began for algorithms that can handle arbitrary network structures. Since polytrees were also referred to as singly–connected networks, arbitrary network structures were said to be multiply–connected; see Figure 1. One of the central ideas for handling these networks is based on the technique of conditioning. That is, one can set variable X to some value x and then solve the problem under that particular condition $X = x$. If this is repeated for all values of X, then one can recover the answer to the original problem by assembling the results obtained from the individual cases. The main value of this technique is that by conditioning variables on some values, one can simplify the problem. In Bayesian networks, one can effectively delete edges that are outgoing from a node once the value of that node is known, therefore, creating a simplified structure that can be as informative as the original structure in terms of answering queries.

Pearl used this observation to propose the algorithm of *loop-cutset conditioning* [Pearl 1986a; Pearl 1988], which worked by conditioning on enough network variables to render the network structure singly–connected. The set of variables that needed to be conditioned on is called a *loop–cutset*; see Figure 2. The loop–cutset conditioning algorithm amounted then to a number of invocations to the polytree algorithm, where this number is exponential in the size of the cutset — one invocation for each instantiation of the variables constituting the cutset. A key attraction of this algorithm is its modest space requirements, as it did not need much space beyond that used by the polytree algorithm. The problem with the algorithm, however, was in its time requirements when the size of the loop-cutset was large enough. The algorithm proved impractical in such a case and the search continued for alternative conditioning algorithms that could handle multiply–connected structures more efficiently.

The very first algorithm that found widespread use on multiply–connected networks was the *jointree* algorithm, also known as the *tree clustering* algorithm [Lauritzen and Spiegelhalter 1988]. This algorithm proved quite effective and remains practically influential un-

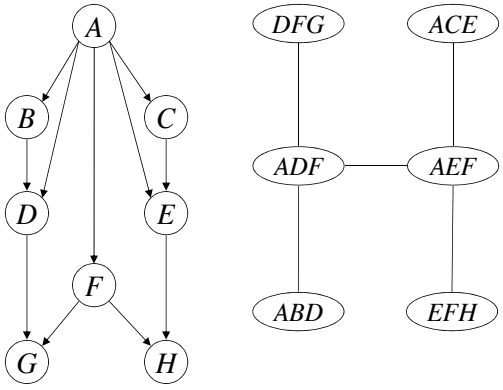

Figure 3. A Bayesian network structure and its corresponding jointree (tree of clusters).

til today — for example, it is the algorithm of choice in commercial implementations of Bayesian network inference. One way of understanding this algorithm is as a version of the polytree algorithm, invoked on a tree clustering of the multiply–connected network. For an example, consider Figure 3 which depicts a DAG and its corresponding tree of clusters — this is technically known as a *jointree* or a *tree decomposition* [Robertson and Seymour 1986]. One thing to notice here is that each cluster is a set of variables in the original network. The jointree algorithm works by passing messages across the tree of clusters, just as in the polytree algorithm. However, the size of these messages and the amount of work it takes to propagate them is now tied to the size of clusters.

The jointree is not an arbitrary tree of clusters as it must satisfy some conditions to legitimize the message passing algorithm. In particular, every node and its parents in the Bayesian network must belong to some tree cluster. Moreover, if a variable appears in two clusters, it must also appear in every cluster on the path connecting them. Ensuring these conditions may lead to clusters that are large. There is a graph–theoretic notion, known as *treewidth,* which puts a lower bound on the size of largest cluster [Robertson and Seymour 1986]. In particular, if the treewidth of the DAG is w, then any jointree of the DAG must have a cluster whose size is at least $w + 1$.[1] In some sense, the treewidth can be viewed as a measure of how similar a DAG structure is to a tree structure as it puts a lower bound on the width of any tree clustering (jointree) of the DAG.

The connection between the complexity of inference algorithms and treewidth is actually the central complexity result that we have today for exact inference [Dechter 1996]. In particular, given a jointree whose width is w, node marginals can be computed in time and space that is exponential only in w. Note that a network treewidth of w guarantees the existence of such a jointree, but finding it is generally known to be hard. Hence, much

[1] In graph theory, treewidth is typically defined for undirected graphs. The treewidth of a DAG as used here corresponds to the treewidth of its moralized graph: one which is obtained by connecting every pair of nodes that share a child in the DAG and then dropping the directionality of all edges.

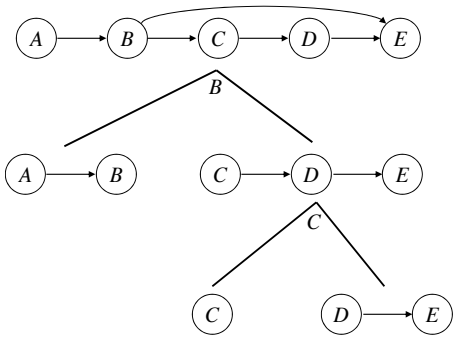

Figure 4. Decomposing a Bayesian network by conditioning on variable B and then on variable C.

work on this topic concerns the construction of jointrees with minimal width using both heuristics and complete search methods (see [Darwiche 2009] for a survey).

4 More computational paradigms

Since a typical implementation of the jointree algorithm will indeed use as much time and space as is suggested by the complexity analysis, we will not be able to rely on the jointree algorithm in the case where we do not find a jointree whose width is small enough. To overcome this treewidth barrier, research on inference algorithms continued in a number of directions.

With regards to work on conditioning algorithms, the main breakthrough in this regard was based on observing that one can employ conditioning in other and more effective ways than loop–cutset conditioning. For example, one can condition on enough variables to split the network into disconnected sub–networks, which can then be solved independently. These sub–networks need not be polytrees, as each one of them can be solved recursively using the same method, until sub–networks reduce to a single node each; see Figure 4. With appropriate caching schemes to avoid solving the same sub–network multiple times, this method of *recursive conditioning* can be applied with the same complexity as the jointree algorithm. In particular, one can guarantee that the space and time requirements of the algorithm are at most exponential in the treewidth of underlying network structure. This result assumes that one has access to a decomposition structure, known as a *dtree,* which is used to control the decomposition process at each level of the recursive process [Darwiche 2001]. Similar to a jointree, finding an optimal dtree (i.e., one that realizes the treewidth guarantee on complexity) is hard. Yet, one can easily construct such a dtree given an optimal jointree, and vice versa [Darwiche 2009].

Even though recursive conditioning and the jointree algorithm are equivalent from this complexity viewpoint, recursive conditioning provided some new contributions to inference. On the theoretical side, it showed that conditioning as an inference paradigm can indeed reach the same complexity as the jointree algorithm — a question that was open

for some time. Second, the algorithm provided a flexible paradigm for time-space tradeoffs: by simply controlling the degree of caching, the space requirements of the algorithm can be made to range from being only linear in the network size to being exponential in the network treewidth (given an appropriate dtree). Moreover, the algorithm provided a convenient framework for exploiting local structure as we shall discuss later.

On another front, and in the continued search of an alternative for the jointree algorithm, a sequence of efforts culminated into what is known today as the *variable elimination* algorithm [Zhang and Poole 1994; Dechter 1996]. According to this algorithm, one maintains the probability distribution of the Bayesian network as a set of factors (initially the set of CPTs) and then successively eliminates variables from this set one variable at a time.[2] The elimination of a variable can be implemented by simply combining all factors that mention that variable and then removing the variable from the combined factor. After eliminating a variable, the resulting factors represent a distribution over all remaining (un-eliminated) variables. Hence, by repeating this elimination process, one can obtain the marginal distribution over any subset of variables, including, for example, marginals over single variables.

The main attraction of this computational paradigm is its simplicity — at least as compared to the initial formulations of the jointree algorithm. Variable elimination, however, turned out to be no more efficient than the jointree algorithm in the worst case. In particular, the ideal time and space complexities of the algorithm also depend on the treewidth — in particular, they are exponential in treewidth when computing the marginal over a single variable. To achieve this complexity, however, one needs to use an optimal order for eliminating variables [Bertele and Brioschi 1972]. Again, constructing an optimal elimination order that realizes the treewidth complexity is hard in general. Yet, one can easily construct such an optimal order from an optimal jointree or dtree, and vice versa.

Even though variable elimination proved to have the same treewidth complexity as the jointree algorithm, it better explained the semantics of the jointree algorithm, which can now be understood as a sophisticated form of variable elimination. In particular, one can interpret the jointree algorithm as a refinement on variable elimination in which: (1) multiple variables can be eliminated simultaneously instead of one variable at a time; (2) a tree structure is used to control the elimination process and to save the results of intermediate elimination steps. In particular, each message passed by the jointree algorithm can be interpreted as the result of an elimination process, which is saved for re-use when computing marginals over different sets of variables [Darwiche 2009]. As a result of this refinement, the jointree algorithm is able to perform successive invocations of the variable elimination algorithm, for computing multiple marginals, while incurring the cost of only one invocation, due mainly to the re-use of results across multiple invocations.

Given our current understanding of the variable elimination and jointree algorithms, one now speaks of only two main computational paradigms for exact probabilistic inference: conditioning algorithms (including loop-cutset conditioning and recursive conditioning) and elimination algorithms (including variable elimination and the jointree algorithm).

[2] A factor is a function that maps the instantiations of some set of variables into numbers; see Figure 5. In this sense, each probability distribution is a factor and so is the marginal of such a distribution on any set of variables.

X	Y	Z	$f(.)$
F	F	F	0.9
F	F	T	0.1
F	T	F	0.9
F	T	T	0.1
T	F	F	0.1
T	F	T	0.9
T	T	F	0.5
T	T	T	0.5

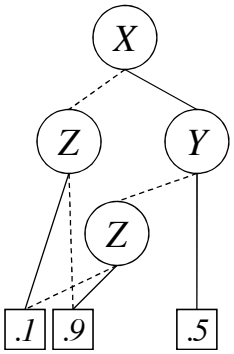

Figure 5. A factor over binary variables X, Y, Z with a tabular representation (left) and an ADD representation (right).

5 Beating the treewidth barrier with local structure

Assuming that we ignore the probabilities that quantify a Bayesian network, the treewidth guarantee is the best we have today on the complexity of exact inference. Moreover, the treewidth determines the best-case performance we can expect from the standard algorithms based on conditioning and elimination.

It has long been believed though that exploiting the local structure of a Bayesian network can speed up inference to the point of beating the treewidth barrier, where local structure refers to the specific properties attained by the probabilities quantifying the network. One of the main intuitions here is that local structure can imply independence that is not visible at the structural level and this independence may be utilized computationally [Boutilier et al. 1996]. Another insight is that determinism in the form of $0/1$ probabilities can also be computationally useful as it allows one to prune possibilities from consideration [Jensen and Andersen 1990]. There are many realizations of these principles today. For elimination algorithms — which rely heavily on factors and their operations — local structure permits one to have more compact representations of these factors than representations based on tables [Zhang and Poole 1996], leading to a more efficient implementation of the elimination process. One example of this would be the use of Algebraic Decision Diagrams [R.I. Bahar et al. 1993] and associated operations to represent and manipulate factors; see Figure 5. For conditioning algorithms, local structure reduces the number of cases one needs to consider during inference and the number of sub-computations one needs to cache. As an example of the first, suppose that we have an and-gate whose output and one of its inputs belong to a loop cutset. When conditioning the output on 1, both inputs must be 1 as well. Hence, there is no need to consider multiple values for the input in this case during the

conditioning process [Allen and Darwiche 2003]. This would no longer be true, however, if we had an or-gate. Moreover, the difference between the two cases is only visible if we exploit the local structure of corresponding Bayesian networks.

Another effective technique for exploiting local structure, which proved to be a turning point in speeding up inference, is based on encoding Bayesian networks using logical constraints and then applying logical inference techniques to the resulting knowledge base [Darwiche 2002]. One can indeed efficiently encode the network structure and some of its local structure, including determinism, using knowledge bases in conjunctive normal form (CNF). One can then either compile the CNF to produce a circuit representation of the Bayesian network (see below), or apply model counting techniques and use the results to recover answers to probabilistic queries [Sang, Beame, and Kautz 2005].

Realizations of the above techniques became practically viable long after the initial observations about local structure, but have allowed one to reason efficiently with some networks whose treewidth can be quite large (e.g., [Chavira, Darwiche, and Jaeger 2006]). Although there is some understanding of the kind of networks that tend to lend themselves to these techniques, we still do not have strong theoretical results that characterize these classes of networks and the savings that one may expect from exploiting their local structure. Moreover, not enough work exists on complexity measures that are sensitive to both network structure and parameters (the treewidth is only sensitive to structure).

One step in this direction has been the use of arithmetic circuits to compactly represent the probability distributions of Bayesian networks [Darwiche 2003]. This representation is sensitive to both network topology and local structure, therefore, allowing for compact circuit representations in some cases where the treewidth of the network can be quite large; see Figure 6. Given a circuit representation, inference can be performed quite efficiently through simple circuit evaluation and differentiation techniques. Hence, the size of a circuit representation can be viewed as an indicator of the complexity of inference with respect to the given network. Again, however, we do not have enough theoretical results to broadly predict the size of these circuit representations or bound the complexity of constructing them.[3]

6 More queries for Bayesian networks

Pearl introduced another computational problem for Bayesian networks, known as the MPE for Most Probable Explanations. The goal here is to find the most likely instantiation of the network variables, given that some of these variables are fixed to some given value. Pearl actually proposed the first algorithm for this purpose, which was a variation on the polytree algorithm [Pearl 1987a].

A more general problem is MAP which stands for Maximum a Posteriori hypothesis. This problem searches for an instantiation of a subset of the network variables that is most probable. Interestingly, MAP and MPE are complete for two different complexity classes, which are also distinct from the class to which node marginals is complete for. In particular,

[3]Note, however, that an arithmetic circuit can always be constructed in time which is exponential only in the treewidth, given a jointree of corresponding width.

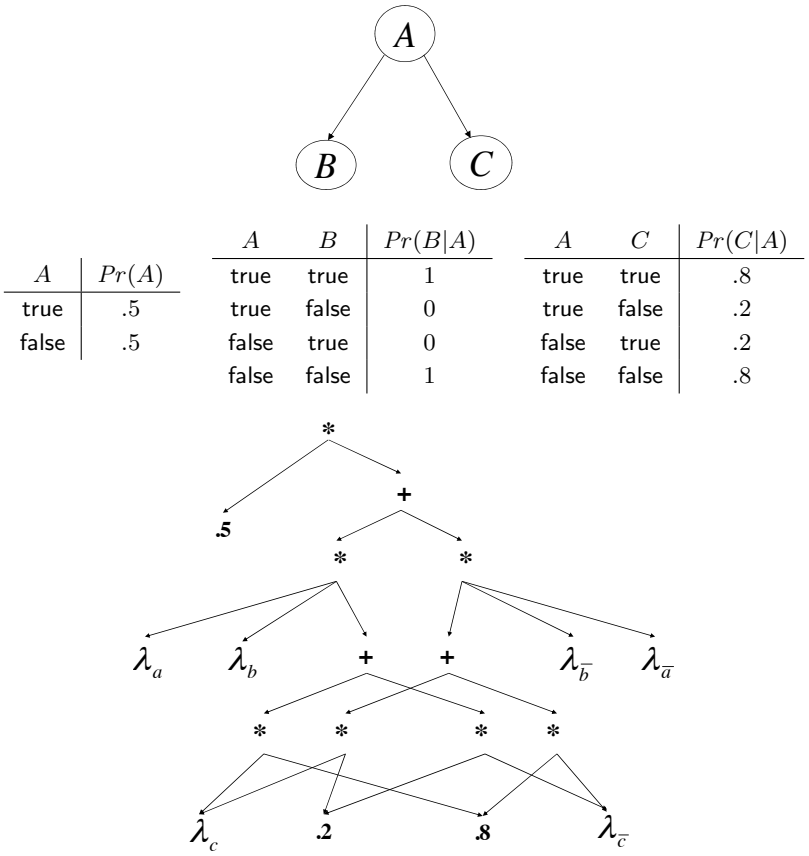

Figure 6. A Bayesian network and a corresponding arithmetic circuit.

given the standard assumptions of complexity theory, MPE is the easiest and MAP is the most difficult, with node marginals in the middle.[4]

The standard techniques based on variable elimination and conditioning can solve MPE and MAP as well [Dechter 1999]. MPE can be solved with the standard treewidth guarantee. MAP, however, has a worse complexity in terms of what is known as *constrained treewidth,* which depends on both the network topology and MAP variables (that is, variables for which we are trying to find a most likely instantiation of) [Park and Darwiche 2004]. The constrained treewidth can be much larger than treewidth, depending on the set of MAP variables.

MPE and MAP problems have search components which lend themselves to branch-and-bound techniques [Kask and Dechter 2001]. Over the years, many sophisticated MPE and MAP bounds have been introduced, allowing branch-and-bound solvers to prune the search space more effectively. Consequently, this allows one to solve some MPE and MAP

[4]The decision problems for MPE, node marginals, and MAP are NP–complete, PP–complete, and NP^{PP}–complete, respectively.

problems efficiently, even when the network treewidth or constrained treewidth are relatively high. In fact, only relatively recently did practical MAP algorithms surface, due to some innovative bounds that were employed in branch-and-bound algorithms [Park and Darwiche 2003].

MPE algorithms have traditionally received more attention than MAP algorithms. Recently, techniques based on LP relaxations, in addition to reductions to the MAXSAT problem, have been employed successfully for solving MPE. LP relaxations are based on the observation that MPE has a straightforward formulation in terms of integer programming, which is known to be hard [Wainwright, Jaakkola, and Willsky 2005; Yanover, Meltzer, and Weiss 2006]. By relaxing the integral constraints, the problem becomes a linear program, which is tractable but provides only a bound for MPE. Work in this area has been focused on techniques that compensate partially for the lost integral constraints using larger linear programs, and on developing refined algorithms for handling the resulting "specialized" linear programs.[5] The MAXSAT problem has also been receiving a lot of attention in the logic community [Bonet, Levy, and Manyà 2007; Larrosa, Heras, and de Givry 2008], which developed effective techniques for this purpose. In fact, reductions of certain MPE problems (those with excessive logical constraints) to MAXSAT seem to be the state of the art for some problems in this category.

7 Approximations may be good enough

In addition to work on exact inference algorithms for Bayesian networks, much work has also been dedicated to approximate inference algorithms which are generally more efficient but settle for less than accurate answers. Interestingly enough, the two major paradigms for approximate inference as practiced today were also initiated by Judea Pearl.

In particular, immediately after proposing the polytree algorithm, Pearl also proposed the use of Gibbs sampling as a method for approximate inference in Bayesian networks [Pearl 1987b]. This paper started a tradition in applying MCMC techniques for solving Bayesian networks and is considered as the founding paper in this direction. Further stochastic simulation methods were also proposed after realizing that sampling from Bayesian networks can be done easily by simply traversing the network structure [Henrion 1988].

In his seminal book on Bayesian networks [Pearl 1988], Pearl also proposed applying the belief propagation (polytree) algorithm to networks that have an arbitrary structure (in Exercise 4.7). This proposal required some initialization of network messages and entailed that a node may have to keep sending messages to each of its neighbors until convergence is reached (i.e., the messages are no longer changing); see Figure 7. Interestingly enough, such an algorithm, which is now known as *loopy belief propagation (LBP),* tends to converge, yielding good approximations to a variety of problems. In fact, this particular algorithm was found to correspond to a state–of–the–art algorithm used in the channel coding community and today is widely viewed as a key method of choice for approximate inference [Frey and MacKay 1997].

[5] In the community working on LP relaxations and related methods, "MAP" is used to mean "MPE" as we have discussed it in this article.

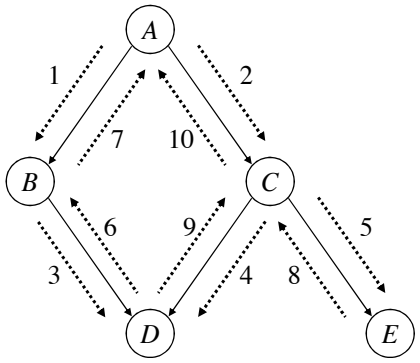

Figure 7. A Bayesian network annotated with an ordering of LBP messages (leading to a sequential message passing schedule).

This connection and the viability of LBP as an approximation algorithm came to light around the mid 1990s, almost a decade after Pearl first suggested the algorithm. Work on LBP and related methods has been dominating the field of approximate inference for more than a decade now. One of the central questions was: if LBP converges, what is it converging to? This question was answered in a number of ways [Minka 2001; Wainwright, Jaakkola, and Willsky 2003; Choi and Darwiche 2006], but the first characterization was put forth in [Yedidia, Freeman, and Weiss 2000]. According to this characterization, one can understand LBP as approximating the distribution of a Bayesian network by a distribution that has a polytree structure [Yedidia, Freeman, and Weiss 2003]. The iterations of the algorithm can then be interpreted as searching for the node marginals of that approximate distribution, while minimizing the KL–divergence between the original and approximate distributions.

LBP has actually two built-in components. The first corresponds to a particular approximation that it seeks, which is formally characterized as discussed before. The second component is a particular method for seeking the approximation, through a process of message passing. One can try to seek the same approximation using other optimization methods, which has also been the subject of much research. Even the message passing scheme leaves a lot of room for variation, which is captured formally using the notion of a message passing schedule — for example, messages can be passed sequentially, in parallel, or combinations therefore. One therefore talks about the "convergence" properties of such algorithms, where the goal is to seek methods that have better convergence properties.

LBP turns out to be an example of a more general class of approximation algorithms that poses the approximate inference problem as a constrained optimization problem. These methods, which are sometimes known as *variational algorithms,* assume a tractable class of distributions, and seeks to find an instance in this class that best fits the original distribution [Jordan et al. 1999; Jaakkola 2001]. For example, we may want to assume an approximating Bayesian network that is fully-disconnected, and that the distribution it in-

duces should have as small a KL–divergence as possible, when compared to the distribution being approximated. The goal of the constrained optimization problem is then to find the CPT parameters of the approximate network that minimizes the KL–divergence between it and the original network (subject to the appropriate normalization constraints). Work in this area typically varies across two dimensions: proposing forms for the approximating distribution, and devising methods for solving the corresponding optimization problem. Moreover, by varying these two dimensions, we are given access to a spectrum of approximations, where we are able to trade the quality of an approximation with the complexity of computing it.

8 Closing Remarks

During the first decade or two after Pearl's introduction of Bayesian networks, inference research was very focused on exact algorithms. The efforts on these algorithms slowed down towards the mid to late 1990s, to pick up again early in the century. The slowdown was mostly due to the treewidth barrier, at a time where large enough networks were being constructed to make standard algorithms impractical at that time. The main developments leading to the revival of exact inference algorithms has been the extended reach of conditioning methods, the deeper understanding of elimination methods, and the more effective exploitation of local structure. Even though these developments have increased the reach of exact algorithms considerably, we still do not understand the extent to which this reach can be pushed further. In particular, the main hope appears to be in further utilization of local structure to speed up inference, but we clearly need better theories for providing guarantees on such speedups and a better characterization of the networks that lend themselves to such techniques.

On the approximate inference side, stochastic simulation methods witnessed a surge after the initial work on this subject, with continued interest throughout, yet not to the level enjoyed recently by methods based on belief propagation and related methods. This class of algorithms remains dominant, with many questions begging for answers. On the theoretical side, we do not seem to know enough on when approximations tend to give good answers, especially that this seems to be tied not only to the given network but also to the posed query. On the practical side, we have yet to translate some of the theoretical results on generalizations of belief propagation — which provides a spectrum that tradeoffs approximation quality with computational resources — into tools that are used routinely by practitioners.

There has been a lot of progress on inference in Bayesian networks since Pearl first made this computational problem relevant. There is clearly a lot more to be done as we seem to always exceed the ability of existing algorithms by building more complex networks. In my opinion, however, what is greatly missed since Pearl's initial work on this subject is his insistence on semantics, where he spared no effort in establishing connections to cognition, and in grounding the most intricate mathematical manipulations in human intuition. The derivation of the polytree algorithm stands as a great example of this research methodology, as it provided high level and cognitive interpretations of almost all intermediate computa-

tions performed by the algorithm. It is no wonder then that the polytree algorithm not only started the area of inference in Bayesian networks a few decades ago, but it also remains a basis for some of the latest developments and inspirations in this area of research.

Acknowledgments: I wish to thank Arthur Choi for many valuable discussions while writing this article.

References

Allen, D. and A. Darwiche (2003). New advances in inference by recursive conditioning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 2–10.

Bertele, U. and F. Brioschi (1972). *Nonserial Dynamic Programming*. Academic Press.

Bonet, M. L., J. Levy, and F. Manyà (2007). Resolution for max-sat. *Artif. Intell. 171*(8-9), 606–618.

Boutilier, C., N. Friedman, M. Goldszmidt, and D. Koller (1996). Context-specific independence in Bayesian networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Twelfth Conference (UAI-96)*, San Francisco, pp. 115–123. Morgan Kaufmann Publishers.

Chavira, M., A. Darwiche, and M. Jaeger (May 2006). Compiling relational Bayesian networks for exact inference. *International Journal of Approximate Reasoning 42*(1–2), 4–20.

Choi, A. and A. Darwiche (2006). An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pp. 1107–1114.

Darwiche, A. (2001). Recursive conditioning. *Artificial Intelligence 126*(1-2), 5–41.

Darwiche, A. (2002). A logical approach to factoring belief networks. In *Proceedings of KR*, pp. 409–420.

Darwiche, A. (2003). A differential approach to inference in Bayesian networks. *Journal of the ACM 50*(3), 280–305.

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Dechter, R. (1996). Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 211–219.

Dechter, R. (1999). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence 113*, 41–85.

Frey, B. J. and D. J. C. MacKay (1997). A revolution: Belief propagation in graphs with cycles. In *NIPS*, pp. 479–485.

Henrion, M. (1988). Propagating uncertainty in Bayesian networks by probalistic logic sampling. In *Uncertainty in Artificial Intelligence 2*, New York, N.Y., pp. 149–163. Elsevier Science Publishing Company, Inc.

Jaakkola, T. (2001). Tutorial on variational approximation methods. In D. Saad and M. Opper (Eds.), *Advanced Mean Field Methods*, Chapter 10, pp. 129–160. MIT Press.

Jensen, F. and S. K. Andersen (1990, July). Approximations in Bayesian belief universes for knowledge based systems. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence (UAI)*, Cambridge, MA, pp. 162–169.

Jordan, M. I., Z. Ghahramani, T. Jaakkola, and L. K. Saul (1999). An introduction to variational methods for graphical models. *Machine Learning 37*(2), 183–233.

Kask, K. and R. Dechter (2001). A general scheme for automatic generation of search heuristics from specification dependencies. *Artificial Intelligence 129*, 91–131.

Kim, J. and J. Pearl (1983). A computational model for combined causal and diagnostic reasoning in inference systems. In *Proceedings IJCAI-83*, Karlsruhe, Germany, pp. 190–193.

Larrosa, J., F. Heras, and S. de Givry (2008). A logical approach to efficient max-sat solving. *Artif. Intell. 172*(2-3), 204–233.

Lauritzen, S. L. and D. J. Spiegelhalter (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of Royal Statistics Society, Series B 50*(2), 157–224.

Minka, T. P. (2001). *A family of algorithms for approximate Bayesian inference*. Ph.D. thesis, MIT.

Park, J. and A. Darwiche (2004). Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research 21*, 101–133.

Park, J. D. and A. Darwiche (2003). Solving MAP exactly using systematic search. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI–03)*, Morgan Kaufmann Publishers San Francisco, California, pp. 459–468.

Pearl, J. (1982). Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings American Association of Artificial Intelligence National Conference on AI*, Pittsburgh, PA, pp. 133–136.

Pearl, J. (1986a). A constraint-propagation approach to probabilistic reasoning. In L. Kanal and J. Lemmer (Eds.), *Uncertainty in Artificial Intelligence*, pp. 357–369. Amsterdam, North Holland.

Pearl, J. (1986b). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence 29*, 241–288.

Pearl, J. (1987a). Distributed revision of composite beliefs. *Artificial Intelligence 33*(2), 173–215.

Pearl, J. (1987b). Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence 32*, 245–257.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California.

R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi (1993). Algebraic Decision Diagrams and Their Applications. In *IEEE /ACM International Conference on CAD*, Santa Clara, California, pp. 188–191. IEEE Computer Society Press.

Robertson, N. and P. D. Seymour (1986). Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms 7*, 309–322.

Rumelhart, D. (1976). Toward an interactive model of reading. Technical Report CHIP-56, University of California, La Jolla, La Jolla, CA.

Sang, T., P. Beame, and H. Kautz (2005). Solving Bayesian networks by weighted model counting. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, Volume 1, pp. 475–482. AAAI Press.

Wainwright, M. J., T. Jaakkola, and A. S. Willsky (2003). Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory 49*(5), 1120–1146.

Wainwright, M. J., T. Jaakkola, and A. S. Willsky (2005). Map estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory 51*(11), 3697–3717.

Yanover, C., T. Meltzer, and Y. Weiss (2006). Linear programming relaxations and belief propagation — an empirical study. *Journal of Machine Learning Research 7*, 1887–1907.

Yedidia, J. S., W. T. Freeman, and Y. Weiss (2000). Generalized belief propagation. In *NIPS*, pp. 689–695.

Yedidia, J. S., W. T. Freeman, and Y. Weiss (2003). Understanding belief propagation and its generalizations. In G. Lakemeyer and B. Nebel (Eds.), *Exploring Artificial Intelligence in the New Millennium*, Chapter 8, pp. 239–269. Morgan Kaufmann.

Zhang, N. L. and D. Poole (1994). A simple approach to Bayesian network computations. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 171–178.

Zhang, N. L. and D. Poole (1996). Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research 5*, 301–328.